Edward (Jed) Frees, University of Wisconsin - Madison, Australian National University

Online Supplement to Constructing Insurable Risk Portfolios

Contents

Preface

Date: 17 January 2025

This is a supplement to the book **Constructing Insurable Risk Portfolios** and will be available online. It contains:

- *Illustrative Statistical Code*,
- *Selected Proofs and Theory Development*, and
- *Selected Exercise Solutions.*

To implement the code, always first run the preliminary set-up code in Section 15.1.

You will be able to test code from each chapter independent of the others. Make sure that you have downloaded the data sets. The code was developed on my machines, so you will need to change the folder/directory set-up to something that works for you.

Expect bugs. You can report them to me at jfrees@bus.wisc.edu.

15

Illustrative Statistical Code

15.1 Preliminary Code

15.1.1 Packages

Here is a list of packages used in this book. You will want to install them prior to testing out the code in this book.

```
library("actuar")
library("alabama")
library("boot")
library("copula")
library("cubature")
library("CVXR")
library("doBy")
library("epitools")
library("forecast")
library("GB2")
library("goftest")
library("gifski")
library("ggplot2")
library("gridExtra")
library("grid")
library("Hmisc")
library("invgamma")
library("kableExtra")
library("knitr")
library("ks")
library("lattice")
library("MASS")
library("metR")
library("mvtnorm")
library("pander")
library("PerformanceAnalytics")
library("psych")
library("quantmod")
library("rootSolve")
library("rgl")
library("rpart")
library("rpart.plot")
library("RColorBrewer")
library("scales")
```

```
library("splines")
library("statmod")
library("survival")
library("tweedie")
library("VGAM")
library("VineCopula")
library("xts")
```

15.1.2 Table Generation

Here is the code that I use to generate most of the tables in the book.

```
#HtmlEval <- TRUE
#  Default Widths for Latex
ColWidth4  <- "1.8cm"
ColWidth5  <- "1.6cm"
ColWidth6  <- "1.4cm"
ColWidth7  <- "1.3cm"
ColWidth8  <- "1.1cm"
ColWidth9  <- "1.0cm"
ColWidth10 <- "0.9cm"
ColWidth14 <- "0.5cm"
#  Function to Create a Style for Generating Tables
TableGen1 <-function(TableData,TextTitle,Align=`r`, Digits=0,
                     ColumnSpec=1, ColumnSpec0=1, ColWidth0= "1.5cm",
                     BorderRight=1, ColWidth= "1.5cm",
                     latexFont= 10){

    if (HtmlEval){ColumnSpec <- ColumnSpec+1
     TextTitle1 <- paste0('**', TextTitle,'**', collapse='')
     kableExtra::kbl(TableData, caption=TextTitle1, align = Align,
                     booktabs = T, digits=Digits) %>%
     kableExtra::kable_classic(full_width = F, font = 12,
                               html_font = "Cambria") %>%
     kable_styling(bootstrap_options = c("striped", "condensed"))  %>%
     kableExtra::column_spec(ColumnSpec, width =  ColWidth) %>%
     kableExtra::column_spec(BorderRight, border_right = TRUE, width =  ColWidth) %>%
     kableExtra::column_spec(ColumnSpec0, width =  ColWidth0)
   } else
   { ColumnSpec <- ColumnSpec+1
     TextTitle1 <- paste0('\\textbf{',TextTitle,'}', collapse='')
     kableExtra::kbl(TableData, caption = TextTitle1, align = Align,
                     booktabs = T, digits=Digits,escape = FALSE) %>%
    kableExtra::kable_styling(latex_options = c("striped", "condensed") ,
                              font = latexFont, protect_latex= TRUE) %>%
       kableExtra::column_spec(ColumnSpec, width =  ColWidth) %>%
    kableExtra::column_spec(BorderRight, border_right = TRUE, width =  ColWidth)  %>%
         kableExtra::column_spec(ColumnSpec0, width =  ColWidth0)
   }
}
```

15.1.3 Basic Summary Functions for the Gamma and Pareto Distributions

```
# These functions are used throughout the book, beginning in Chap 1
# Basic Summary Functions for a gamma Distribution

f1 <- function(x){dgamma(x=x,shape = risk1shape, scale = risk1scale)}
F1 <- function(x){pgamma(q=x,shape = risk1shape, scale = risk1scale)}
q1 <- function(x){qgamma(p=x,shape = risk1shape, scale = risk1scale)}
ERisk1fun  <- function(){actuar::mgamma(order=1, shape = risk1shape,
                                        scale = risk1scale)}
sdRisk1fun <- function(){
       sqrt(actuar::mgamma(order=2, shape = risk1shape,
                           scale = risk1scale) - ERisk1fun()**2)}
RC1 <- function(u){
       ERisk1fun() - actuar::levgamma(limit = u, shape = risk1shape,
                                      scale = risk1scale)}
# Basic Summary Functions for a Pareto Distribution
f2 <- function(x){actuar::dpareto(x=x,shape = risk2shape, scale = risk2scale)}
F2 <- function(x){actuar::ppareto(q=x,shape = risk2shape, scale = risk2scale)}
q2 <- function(x){actuar::qpareto(p=x,shape = risk2shape, scale = risk2scale)}
ERisk2fun  <- function(){actuar::mpareto(order=1, shape = risk2shape,
                                         scale = risk2scale)}
sdRisk2fun <- function(){
      sqrt(actuar::mpareto(order=2, shape = risk2shape,
                           scale = risk2scale) - ERisk2fun()**2)}
RC2 <- function(u){
      ERisk2fun() - actuar::levpareto(limit = u,shape = risk2shape,
                                      scale = risk2scale)}
#  Additive Risk Transfer Cost
RTC <- function(u1,u2){ RC1(u1)+RC2(u2) }
```

15.2 Chapter One Code

15.2.1 Example 1.1. Portfolio of Insurance Stock Returns

```
# Example 1.1
# download data from Yahoo Finance
# Pick some insurance stocks...
# https://companiesmarketcap.com/insurance/largest-insurance-companies-by-market-cap/
# https://finance.yahoo.com/
library(quantmod)  # to download stock data
stock_namelist <- c("UNH", "1299.HK","2318.HK",  "CB",
                    "ALV.DE", "AON",  "CS.PA",  "8766.T")
datesToFrom <- c("2015-01-01", "2023-12-31")
prices <- xts()
for (i in 1:length(stock_namelist)) {
  tmp <- Ad(getSymbols(stock_namelist[i], from = datesToFrom[1],
                       to = datesToFrom[2], auto.assign = FALSE))
  tmp <- na.approx(tmp, na.rm = FALSE)  # interpolate NAs
  prices <- cbind(prices, tmp)
  }
for (i in 1:ncol(prices)) {
  prices[,i] <- na.approx(prices[,i], na.rm = FALSE)  # interpolate NAs
  }

Short_stock_namelist <- c("United Health", "AIA","Ping An",  "Chubb",
                          "Allianz", "Aon",  "AXA",  "Tokio Marine")
colnames(prices) <- Short_stock_namelist
indexClass(prices) <- "Date"
prices <- prices[-nrow(prices),]
prices <- prices[-1,]
# I 'save' and 'load' intermediate steps while developing.
# If you are re-running code to check, this is not necessary.

save(prices, file= "../ChapTablesData/Chap1/InsurSectorPrices2023.Rdata")
```

```
# Example 1.1,  Create Returns
# save(prices, file= "../ChapTablesData/Chap1/InsurSectorPrices2023.Rdata")
load(file= "../ChapTablesData/Chap1/InsurSectorPrices2023.Rdata")

Return <- (prices/stats::lag(prices) - 1)[-1]
p.num  <- ncol(Return)  # number of stocks

#We now divide the data into a training set and test set:
# Training 2015-01-05 to 2019-12-31
# TestingOLD 2020-06-01 to 2022-05-31
# Testing 2020-06-01 to 2023-08-31

Return_trn <- Return[1:1299, ]
```

```
Return_tst <- Return[1427:2188, ]

T_trn <- nrow(Return_trn)
mu    <- colMeans(Return_trn, na.rm = TRUE)
Sigma <- cov(Return_trn, use = "pairwise.complete.obs")

save(Return, p.num, Return_trn, Return_tst,
     T_trn, mu, Sigma,
     file= "../ChapTablesData/Chap1/InsurSectorReturns2023.Rdata")
```

```
#  Plot Figure 1.1
#save(Return, p.num, Return_trn, Return_tst,
#     T_trn, mu, Sigma,
#     file= "../ChapTablesData/Chap1/InsurSectorReturns2023.Rdata")
load(file= "ChapTablesData/Chap1/InsurSectorPrices2023.Rdata")
load(file= "ChapTablesData/Chap1/InsurSectorReturns2023.Rdata")
mydata <- prices/rep(prices[1, ], each = nrow(prices))

library(RColorBrewer)
line_types <- c("solid", "dashed", "dotted","solid",
                "dashed", "dotted", "solid", "dashed")
grey_types <- c("grey50", "grey80","black", "grey50",
                "grey80","black", "grey50", "grey80")

{ if (HtmlEval) {  plot(mydata,   main = "", grid.col = "lightgray", lwd = 1)} else {
 plot(mydata,   main = "", grid.col = "lightgray", #lwd = 1, #col = "black"
      col = grey_types, lty = line_types) }
addLegend("topleft", pch=0, cex=0.8,lty = line_types)
addEventLines(xts("                          ",
                  index(Return["2015-01-06"])), srt=00,
                        pos=1, lwd = 4, col = FigDBlue)
addEventLines(xts("training                ",
                  index(Return["2019-12-31"])), srt=00,
                        pos=1, lwd = 4, col = FigDBlue)
addEventLines(xts("                testing",
                  index(Return["2020-06-01"])),  srt=0,
                        pos=1, lwd = 4, col = FigDGreen)
addEventLines(xts("",
                  index(Return["2023-05-31"])),  srt=0,
                        pos=1, lwd = 4, col = FigDGreen)
}
```

```
#  Create Table 1.4
temp1 <- t(table.AnnualizedReturns(Return_trn))
temp2 <- t(table.AnnualizedReturns(Return_tst))
RetSumm <- cbind(as.matrix(temp1), as.matrix(temp2))
colnames(RetSumm) <- c("Train Ann Return",    "Train Std Dev",
                       "Train Sharpe (Rf=0)", "Test Ann Return",
                       "Test Std Dev", "Test Sharpe (Rf=0)" )
```

```
TableGen1(TableData=RetSumm,
         TextTitle='Insurance Sector Performance Measures',
         Align='r', Digits=3, ColumnSpec=1:6,
         ColWidth0= "3.5cm", ColumnSpec0 = 1,
         BorderRight= c(1,4), ColWidth= "1.2cm")
```

15.2.2 Section 1.4.1 Minimal Scenario 1. Effects of Dependence

```
# Section 1.4.1 Illustrative Code
set.seed(2017)
nsim      <- 100000
Q.def     <- 1.3
uparam    <- seq(0, 1, length.out = 11)
u.compare <- 7
u.def     <- uparam[u.compare]
rhoparam  <- (-9:9)/10
rho.ind   <- 10; rho.lower <- 10-3; rho.upper <- 10+3
nrhoparms <- length(rhoparam)
QuanS     <- matrix(0,length(uparam),length(rhoparam))
dfS       <- matrix(0,length(uparam),length(rhoparam))
# Generate Dist Function at Q.def and Quantiles
# for several dependence parameters
for (j in 1:length(rhoparam)) {
  FrankParam <- copula::iRho(frankCopula(param=1, dim = 2), rho=rhoparam[j])
  nc <- copula::frankCopula(param = FrankParam, dim = 2)
  X  <- copula::rCopula(nsim, nc)
  #  Loop over several upper limits
  for (i in 1:length(uparam)) {
    u1 <- uparam[i]
  Sretained <- pmin(X[,1]) + pmin(X[,2],u1)
  if ((j == rho.ind)  *(i==u.compare))    {Sret.ind   <- Sretained }
  if ((j == rho.lower)*(i==u.compare))    {Sret.lower <- Sretained }
  if ((j == rho.upper)*(i==u.compare))    {Sret.upper <- Sretained }
  df         <- ecdf(Sretained)
  dfS[i,j]   <- df(Q.def)
  QuanS[i,j] <- quantile(Sretained, probs = c(0.95))
  }
}
Qaxis <- seq(1.3, 1.65, length.out = 11)
df <- ecdf(Sret.ind);   dfQ.ind   <- df(Qaxis)
df <- ecdf(Sret.lower); dfQ.lower <- df(Qaxis)
df <- ecdf(Sret.upper); dfQ.upper <- df(Qaxis)

Quanaxis <- seq(0.84, 1, 0.01)
QuanS.ind   <- quantile(Sret.ind,   probs = Quanaxis)
QuanS.lower <- quantile(Sret.lower, probs = Quanaxis)
QuanS.upper <- quantile(Sret.upper, probs = Quanaxis)
save(uparam, dfS, QuanS, Sret.ind, dfQ.ind, Sret.lower, dfQ.lower,
     Sret.upper,dfQ.upper, QuanS.ind, QuanS.lower, QuanS.upper,
```

```
     rho.ind, rho.lower, rho.upper, rhoparam,
     file = "../ChapTablesData/Chap1/MinScenario1.RData")
```

```
# Section 1.4.1 Illustrative Code,  Plot Figure 1.2
#save(uparam, dfS, QuanS, Sret.ind, dfQ.ind, Sret.lower, dfQ.lower,
#     Sret.upper,dfQ.upper, QuanS.ind, QuanS.lower, QuanS.upper,
#     rho.ind, rho.lower, rho.upper, rhoparam,
 #    file = "../ChapTablesData/Chap1/MinScenario1.RData")
load(file = "ChapTablesData/Chap1/MinScenario1.RData")
Qaxis <- seq(1.3,1.65, length.out = 11)
Quanaxis <- seq(0.84, 1, 0.01)
par(mfrow = c(2,2))
plot(Qaxis,dfQ.ind, type = "l", ylab="Distribution Function",
     xlab = "Capital", main="u=0.6")
lines(Qaxis,dfQ.lower,  col=FigBlue, lty=2)
lines(Qaxis,dfQ.upper,  col=FigRed, lty=3)

plot(Quanaxis,QuanS.ind, type = "l", ylab="VaR",
     xlab = "Probability", main="u=0.6")
lines(Quanaxis,QuanS.lower,  col=FigBlue, lty=2)
lines(Quanaxis,QuanS.upper,  col=FigRed, lty=3)

plot(uparam,dfS[,rho.ind], type = "l", ylab="Dist Fct at Q=1.3",
     xlab = "Limit Parameter u")
lines(uparam,dfS[,rho.lower],  col=FigBlue, lty=2)
lines(uparam,dfS[,rho.upper],  col=FigRed, lty=3)

ICost <- (1+uparam**2)/2 - uparam
plot(ICost,QuanS[,rho.ind], type = "l", ylab="VaR",
     xlab = "Risk Transfer Cost")
lines(ICost,QuanS[,rho.lower],  col=FigBlue, lty=2)
lines(ICost,QuanS[,rho.upper],  col=FigRed, lty=3)
```

```
# Section 1.4.1 Illustrative Code,  Plot Figure 1.3
par(mfrow = c(1,2))
plot(rhoparam,dfS[6,], ylab="Dist Fct at Q=1.3", xlim = c(-1,1) , cex=0.8,
     xlab="Spearman's Rho", type = "b", main="u=0.6")
plot(rhoparam,QuanS[6,], ylab="VaR", xlim = c(-1,1) , cex=0.8,
     xlab="Spearman's Rho", type = "b", main="u=0.6")
```

15.2.3 Section 1.4.2 Minimal Scenario 2. Form of Risk Transfer

```
# Section 1.4.2 Illustrative Code
nsim <- 100000
set.seed(2017)
Q.def = 1.1
IC0.compare <- 7
```

```
IC0 <- seq(0, 0.5, length.out = 11)
uVec <- 1-sqrt(2*IC0)
cVec <- 1- 2* IC0
dVec <- 1-sqrt(1-2*IC0)
rhoparam<- (-9:9)/10
rho.ind <- 10; rho.lower <- 10-3;rho.upper <- 10+3
nrhoparms <- length(rhoparam)
QuanS <- matrix(0,length(IC0),length(rhoparam))
dfS1 <- matrix(0,length(IC0),length(rhoparam)) -> dfS2 -> dfS3
Sretained <- matrix(0,nsim,3)
Sret.ind.mat <- matrix(0,nsim,3) -> Sret.lower.mat -> Sret.upper.mat
for (j in 1:length(rhoparam)) {
  FrankParam <- iRho(frankCopula(param=1, dim = 2), rho=rhoparam[j])
  nc <- frankCopula(param = FrankParam, dim = 2)
  X <- rCopula(nsim, nc)
  for (i in 1:length(IC0)) {
  Sretained[,1] <- X[,1] + pmin(X[,2],uVec[i])
  Sretained[,2] <- X[,1] + cVec[i]*X[,2]
  Sretained[,3] <- X[,1] + X[,2] - pmin(X[,2],dVec[i])
  if ((j == rho.ind)  *(i==IC0.compare))    {Sret.ind.mat   <- Sretained }
  if ((j == rho.lower)*(i==IC0.compare))    {Sret.lower.mat <- Sretained }
  if ((j == rho.upper)*(i==IC0.compare))    {Sret.upper.mat <- Sretained }
  if ((j == rho.ind))    {Sret.ind.mat   <- Sretained }
  df <- ecdf(Sretained[,1]);   dfS1[i,j] <- df(Q.def)
  df <- ecdf(Sretained[,2]);   dfS2[i,j] <- df(Q.def)
  df <- ecdf(Sretained[,3]);   dfS3[i,j] <- df(Q.def)
  }
}
save(dfS1, dfS2,dfS3, IC0, file = "../ChapTablesData/Chap1/MinScenario2.RData")
```

```
# Section 1.4.2 Illustrative Code,  Plot Figure 1.4
load( file = "ChapTablesData/Chap1/MinScenario2.RData")
rho.ind <- 10; rho.lower <- 10-3;rho.upper <- 10+3
par(mfrow = c(1,3))
plot(IC0,dfS1[,rho.ind], type = "l", ylab="Distribution Function",
                         xlab = expression(RTC[0]),main = "Upper")
lines(IC0,dfS1[,rho.lower],  col=FigBlue, lty=2)
lines(IC0,dfS1[,rho.upper],  col=FigRed, lty=3)
abline(0.9,0,  col="grey")
abline(0.8,0,  col="grey")
abline(0.7,0,  col="grey")
abline(0.6,0,  col="grey")
abline(v = 0.2,  col="grey")
abline(v = 0.3,  col="grey")
abline(v = 0.4,  col="grey")

plot(IC0,dfS2[,rho.ind], type = "l", ylab="Distribution Function",
                         xlab = expression(RTC[0]),main = "Coinsure")
lines(IC0,dfS2[,rho.lower],  col=FigBlue, lty=2)
lines(IC0,dfS2[,rho.upper],  col=FigRed, lty=3)
```

```
abline(0.9,0,  col="grey")
abline(0.8,0,  col="grey")
abline(0.7,0,  col="grey")
abline(0.6,0,  col="grey")
abline(v = 0.2,  col="grey")
abline(v = 0.3,  col="grey")
abline(v = 0.4,  col="grey")

plot(IC0,dfS3[,rho.ind], type = "l", ylab="Distribution Function",
                         xlab = expression(RTC[0]),main = "Deduct")
lines(IC0,dfS3[,rho.lower],  col=FigBlue, lty=2)
lines(IC0,dfS3[,rho.upper],  col=FigRed, lty=3)
abline(0.9,0,  col="grey")
abline(0.8,0,  col="grey")
abline(0.7,0,  col="grey")
abline(0.6,0,  col="grey")
abline(v = 0.2,  col="grey")
abline(v = 0.3,  col="grey")
abline(v = 0.4,  col="grey")
```

```
# Section 1.4.2 Illustrative Code,  Plot Figure 1.5
par(mfrow = c(1,3))
plot(IC0,dfS1[,rho.lower], type = "l", ylab="Distribution Function",
                           xlab = expression(RTC[0]), main="Neg Depend")
lines(IC0,dfS2[,rho.lower],  col=FigGreen, lty=2)
lines(IC0,dfS3[,rho.lower],  col=FigOrange, lty=3)
abline(0.9,0,  col="grey")
abline(0.8,0,  col="grey")
abline(0.7,0,  col="grey")
abline(0.6,0,  col="grey")
abline(v = 0.2,  col="grey")
abline(v = 0.3,  col="grey")
abline(v = 0.4,  col="grey")

plot(IC0,dfS1[,rho.ind], type = "l", ylab="Distribution Function",
                         xlab = expression(RTC[0]), main="Independ")
lines(IC0,dfS2[,rho.ind],  col=FigGreen, lty=2)
lines(IC0,dfS3[,rho.ind],  col=FigOrange, lty=3)
abline(0.9,0,  col="grey")
abline(0.8,0,  col="grey")
abline(0.7,0,  col="grey")
abline(0.6,0,  col="grey")
abline(v = 0.2,  col="grey")
abline(v = 0.3,  col="grey")
abline(v = 0.4,  col="grey")

plot(IC0,dfS1[,rho.upper], type = "l", ylab="Distribution Function",
                           xlab = expression(RTC[0]), main="Pos Depend")
lines(IC0,dfS2[,rho.upper],  col=FigGreen, lty=2)
lines(IC0,dfS3[,rho.upper],  col=FigOrange, lty=3)
```

```
abline(0.9,0,  col="grey")
abline(0.8,0,  col="grey")
abline(0.7,0,  col="grey")
abline(0.6,0,  col="grey")
abline(v = 0.2,  col="grey")
abline(v = 0.3,  col="grey")
abline(v = 0.4,  col="grey")
```

15.3 Chapter Two Code

15.3.1 Example 2.1. Property Fund Claims Distribution

```
# Example 2.1 Illustrative Code
ClaimLev  <- read.csv("Data/CLAIMLEVEL.csv", header=TRUE)
ClaimData <- subset(ClaimLev,Year==2010);     #2010 subset
ClaimsBC  <- ClaimData$Claim
prob.seq  <- seq(0.50, 0.99, by=0.001)
VaR <- quantile(ClaimsBC, prob.seq)
CTE <- VaR
for (i in (1:length(prob.seq)))
   {CTE[i] <- sum(ClaimsBC*(ClaimsBC>VaR[i]))/sum(ClaimsBC>VaR[i])
   }
options(scipen=10)
par(mfrow=c(1, 3))
  plot(prob.seq, VaR, log = "y", xlab = 'Confidence Level')
    abline(v = 0.8, col = FigBlue, lty = 2)
  plot(prob.seq, CTE, log = "y", xlab = 'Confidence Level', ylab = 'ES')
    abline(v = 0.8, col = FigBlue, lty = 2)
  plot(VaR, CTE, log = "xy", ylab = 'ES')
    abline(0,1)
```

15.3.2 Example 2.2. Property Fund and the Pareto Distribution

```
# Example 2.2 Illustrative Code
#  Inference assuming a Pareto Distribution
fit.pareto <- VGAM::vglm(Claim ~ 1, paretoII, loc=0, data = ClaimData)

gamma.P <- as.numeric(exp(coef(fit.pareto))[1])
eta.P   <- as.numeric(exp(coef(fit.pareto))[2])
alpha   <- 0.80
VaR     <- gamma.P*(1-alpha)**(-1/eta.P) - gamma.P
beta.seq <- seq(0.01, 0.199, by=0.001)
RVaR    <- gamma.P/beta.seq*
             ((1-alpha)**(1-1/eta.P) -(1-alpha-beta.seq)**(1-1/eta.P)
               )/(1-1/eta.P) -  gamma.P
betaCTE <- 0.2 - 10^(-16)
CTE <- gamma.P/betaCTE*
             ((1-alpha)**(1-1/eta.P) -(1-alpha-betaCTE)**(1-1/eta.P)
               )/(1-1/eta.P) -  gamma.P
```

```
# Example 2.2 Illustrative Code,  Plot Figure 2.2
beta.seqplot <- c(0, beta.seq, 0.2)
RVaRplot     <- c(VaR, RVaR, CTE)
plot(beta.seqplot, RVaRplot, log = "y", ylab='RVaR',
```

```
     xlab = expression(beta), type = 'l')
  abline(h=VaR,  col = FigBlue, lty = 2)
  abline(h=CTE,  col = FigGreen, lty = 4)
  text(0.02, 380000, "ES", cex = .8)
  text(0.18, 10000, "VaR", cex = .8)
```

15.3.3 Section 2.2.2. Distribution Function (of Retained Losses)

```
# Plot Figure 2.3
x <- seq(-30, 130, by = .02)
y <- (.2+(x/100)^2)*(x<60)*(x>0) + 1*(x>=60)
plot(x,y, xlim=c(-50, 130), ylim=c(0,1.2), type="l", xaxt="n", yaxt="n",
     xlab="z", ylab=expression(F[g](z)))
axis(1, c(0, 60), labels=c("0", "c(u-d)"), cex=1.2)
axis(2, c(.2, 0.57, .8,1), labels=c("F(d)", "F(u)",expression(alpha),"1"), cex=1.2)
points(0, .2, pch=19)
points(60, 1, pch=19)
```

15.3.4 Example 2.3. Property Fund Claims Distribution - Continued

```
# Example 2.3 Set Up
ClaimsBC  <- ClaimData$Claim
alpha = 0.99
VaR.alpha = quantile(ClaimsBC, alpha)
```

```
# Example 2.3 Illustrative Code
maxClaim <- max(ClaimsBC)
sortClaims <- sort(ClaimsBC, decreasing = TRUE)
maxClaimN <- sortClaims[1]
maxClaimN.1 <- sortClaims[2]
u.seq = seq(0.01*maxClaimN.1, 0.25*maxClaimN.1, length.out = 110)
VaR.u <- pmin(VaR.alpha, u.seq)

LEV <- mean(pmin(ClaimsBC,VaR.alpha))
CTE.u <- VaR.u
for (i in (1:length(u.seq)))
   {
  LEVi <- mean(pmin(ClaimsBC,u.seq[i]))
  CTE.u[i] <-  ((LEVi - LEV + (1-alpha)*VaR.alpha)/(1-alpha))*
                (u.seq[i] >= VaR.alpha) +
                 u.seq[i]*(u.seq[i] < VaR.alpha)
}

options(scipen=10)
par(mfrow=c(1, 2))
  plot(u.seq, VaR.u, xlab = 'Upper Limit',
```

```
     ylab = "VaR", type = 'l', ylim = c(40000,600000))
    abline(v = VaR.alpha, col = FigBlue, lty = 2)
  plot(u.seq, CTE.u, xlab = 'Upper Limit',
     ylab = "ES", type = 'l', ylim = c(40000,600000))
    abline(v = VaR.alpha, col = FigBlue, lty = 2)
```

15.3.5 Example 2.4. Changes of Risk Retention Summary Measures Using a Pareto Distribution

Some Preliminary Functions

```
ParetoRTC <-function(d,c,u,risk2scale=1000,risk2shape=3){
  d <- d*(d>0)
  u <- u*(u>0)
  c <- c*(c>=0)*(c<=1) + 1*(c>1)
  Prem <- c*risk2scale/(risk2shape-1)*(
          (risk2scale/(d+risk2scale))**(risk2shape-1) -
          (risk2scale/(u+risk2scale))**(risk2shape-1) )
  RTC <- ERisk2fun() - Prem
  return(RTC)
  }
ParetoQuantile <-
  function(d,c,u,risk2scale=1000,risk2shape=3,alpha=0.98){
    d <- d*(d>0); u <- u*(u>0); c <- c*(c>=0)*(c<=1) + 1*(c>1)
    Quan <- c*(q2(alpha)-d)*(F2(d) <= alpha)*(alpha <= F2(u)) +
          c*(u-d)*(alpha >= F2(u))
    if (u > 10e23){  Quan <- c*(q2(alpha)-d)*(F2(d) <= alpha)}
    return(Quan)
  }

ParetoCTE <-function(d,c,u,risk2scale=1000,risk2shape=3,alpha=0.98){
  d <- d*(d>0); u <- u*(u>0); u <- pmin(u,1e10); c <- c*(c>=0)*(c<=1) + 1*(c>1)
  Quan <- ParetoQuantile(d,c,u,risk2scale,risk2shape,alpha)
  EX.u    <- levpareto(limit = u,   shape = risk2shape, scale = risk2scale)
  EX.d    <- levpareto(limit = d,   shape = risk2shape, scale = risk2scale)
  EX.Quan <- levpareto(limit = Quan,shape = risk2shape, scale = risk2scale)
  k       <- c/(1-alpha)
  Case1   <- c/(1-alpha)*(EX.u - EX.d)
  Case2   <- c*(Quan - d) + c/(1-alpha)*(EX.u - EX.Quan)
  Case3   <- c*(u - d)
  CTE     <- Case1 *( alpha < F2(d)) + Case2 *(F2(d) <  alpha ) *( alpha < F2(u)) +  Case3 *( alpha >
  return(CTE)
}
```

Evaluating Discrete Changes

```
# Example 2.4 Illustrative Code
risk2scale <- 1000
```

```
risk2shape <- 3
alpha      <- 0.98

OutMat <- matrix(NaN,4,8)
colnames(OutMat) <- c(" Deduct $d$"," Coins $c$"," Upper Limit $u$",
                      "$RTC$","$VaR$",
                      "$\\frac{\\Delta VaR}{\\Delta RTC}$",
                      "$ES$",
                      "$\\frac{\\Delta ES}{\\Delta RTC}$")
OutMat[,1]   <- c(0,100, 0, 0)
OutMat[,2] <- c(1, 1, 0.9, 1)
OutMat[,3] <- c(Inf, Inf, Inf, 2000)
OutMat[,4]  <- c(ParetoRTC(d=OutMat[1,1],c=OutMat[1,2],u=Inf),
                 ParetoRTC(d=OutMat[2,1],c=OutMat[2,2],u=Inf),
                 ParetoRTC(d=OutMat[3,1],c=OutMat[3,2],u=Inf),
                 ParetoRTC(d=OutMat[4,1],c=OutMat[4,2],u=OutMat[4,3]))
OutMat[,5]  <- c(ParetoQuantile(d=OutMat[1,1],c=OutMat[1,2],u=Inf),
                 ParetoQuantile(d=OutMat[2,1],c=OutMat[2,2],u=Inf),
                 ParetoQuantile(d=OutMat[3,1],c=OutMat[3,2],u=Inf),
                 ParetoQuantile(d=OutMat[4,1],c=OutMat[4,2],u=OutMat[4,3]))
OutMat[2,6] <- (OutMat[1,5]-OutMat[2,5])/(OutMat[1,4]-OutMat[2,4])
OutMat[3,6] <- (OutMat[1,5]-OutMat[3,5])/(OutMat[1,4]-OutMat[3,4])
OutMat[4,6] <- (OutMat[1,5]-OutMat[4,5])/(OutMat[1,4]-OutMat[4,4])
OutMat[,7]  <- c(ParetoCTE(d=OutMat[1,1],c=OutMat[1,2],u=Inf),
                 ParetoCTE(d=OutMat[2,1],c=OutMat[2,2],u=Inf),
                 ParetoCTE(d=OutMat[3,1],c=OutMat[3,2],u=Inf),
                 ParetoCTE(d=OutMat[4,1],c=OutMat[4,2],u=OutMat[4,3]))
OutMat[2,8] <- (OutMat[1,7]-OutMat[2,7])/(OutMat[1,4]-OutMat[2,4])
OutMat[3,8] <- (OutMat[1,7]-OutMat[3,7])/(OutMat[1,4]-OutMat[3,4])
OutMat[4,8] <- (OutMat[1,7]-OutMat[4,7])/(OutMat[1,4]-OutMat[4,4])
footnote1 <- "As with the statistical package `R` conventions, `Inf` means 'infinity'"
footnote2 <- "and `NaN` means 'not a number.'"
footnote3 <- "The symbol $\\\\Delta$ means 'change in'."
if (HtmlEval == TRUE){
     footnote3 <- "The symbol $\\Delta$ means 'change in'." }

#  Table 2.1
if (knitr::is_html_output()){
kableExtra::kbl(OutMat,
caption='**Risk Transfer Costs and $VaR$s for Selected Risk Retention Parameters**',
              align = 'ccccccc',
              booktabs = T, digits=2) %>%
  kableExtra::kable_classic(full_width = F, font = 12,
                            html_font = "Cambria") %>%
  kable_styling(bootstrap_options = c("striped", "condensed")) %>%
                footnote( general = c(footnote1, footnote2, footnote3) )
  }   else
  { TextTitle <- "Risk Transfer Costs and $VaR$s for Selected Risk Retention Parameters"
    TextTitle1 <- paste0('\\textbf{',TextTitle,'}', collapse='')
    kableExtra::kbl(OutMat, escape = FALSE,
                caption=TextTitle1,
```

```
              align = 'ccccccc',
              booktabs = T, digits=2) %>%
    kableExtra::kable_styling(
              latex_options = c("striped", "condensed") ,
              font = 10)   %>%
              column_spec(1:3, width = "0.8cm") %>%
              column_spec(4:8, width = "1.2cm") %>%
              column_spec(3, border_right = TRUE)  %>%
               footnote( general = c(footnote1, footnote2, footnote3) , escape = FALSE)
  }
```

15.3.6 Example 2.5. Changes of Risk Retention Summary Measures Using a Pareto Distribution - Continued

```
# Example 2.5 Illustrative Code, Produces Table 2.3
OutMat <- matrix(NaN,4,8)

OutMat[,1]  <- c(0,100, 500, 1000)
OutMat[,2]  <- c(1, 1, 0.9, 0.9)
OutMat[,3]  <- c(Inf, 10000, 2000, 1500)
OutMat[,4]  <- c(ParetoRTC(d=OutMat[1,1],c=OutMat[1,2],u=Inf),
                 ParetoRTC(d=OutMat[2,1],c=OutMat[2,2],u=Inf),
                 ParetoRTC(d=OutMat[3,1],c=OutMat[3,2],u=Inf),
                 ParetoRTC(d=OutMat[4,1],c=OutMat[4,2],u=OutMat[4,3]))
OutMat[,5]  <- c(ParetoQuantile(d=OutMat[1,1],c=OutMat[1,2],u=Inf),
                 ParetoQuantile(d=OutMat[2,1],c=OutMat[2,2],u=Inf),
                 ParetoQuantile(d=OutMat[3,1],c=OutMat[3,2],u=Inf),
                 ParetoQuantile(d=OutMat[4,1],c=OutMat[4,2],u=OutMat[4,3]))
RM.2 <-function(d,c,u,risk2scale=1000,risk2shape=3,alpha=0.98){
  RM2.c <- -( (q2(alpha)-d)/(ParetoRTC(d,c,u)/c) ) *
              (F2(d) <= alpha)*(alpha <= F2(u)) +
          - ( (u-d)/(ParetoRTC(d,c,u)/c))*(alpha >= F2(u) )
  if (u > 10e23){ RM2.c <-
       - ( (q2(alpha)-d)/(ParetoRTC(d,c,u=Inf)/c) ) * (F2(d) <= alpha)}
  RM2.d <- -1/(1-F2(d))*(alpha >= F2(d))
  RM2.u <- -1/(1-F2(u))*(alpha >= F2(u))
  return(c(RM2.d,RM2.c,RM2.u))
  }

OutMat[1,6:8] <- RM.2(d=OutMat[1,1],c=OutMat[1,2],u=OutMat[1,3])
OutMat[2,6:8] <- RM.2(d=OutMat[2,1],c=OutMat[2,2],u=OutMat[2,3])
OutMat[3,6:8] <- RM.2(d=OutMat[3,1],c=OutMat[3,2],u=OutMat[3,3])
OutMat[4,6:8] <- RM.2(d=OutMat[4,1],c=OutMat[4,2],u=OutMat[4,3])

colnames(OutMat) <- c("$d$","$c$","$u$","$RTC$","$VaR$",
                      "$\\frac{\\partial_d VaR}{\\partial_d RTC}$",
                      "$\\frac{\\partial_c VaR}{\\partial_c RTC}$",
                      "$\\frac{\\partial_u VaR}{\\partial_u RTC}$")
```

```
TableGen1(TableData=OutMat,
         TextTitle='Differential Relative Changes of $VaR$',
         Align='r', Digits=2, ColumnSpec=1:7,
         BorderRight=3, ColWidth = ColWidth7)
```

15.3.7 Section 2.4. Convexity Figures

```
# Figure 2.5 Illustrative Code
risk2scale <- 2000
Q          <- 1.5*ERisk2fun()
alpha      <- 0.8
uparam     <- seq(from = 0, to = 2*ERisk2fun(), length.out = 110)
df         <- F2(Q)*(uparam<Q)  + 1*(uparam >= Q)
Quan       <- pmin(q2(alpha),uparam)
Eg.X       <- levpareto(limit = uparam,
                        shape = risk2shape, scale = risk2scale)
Eg.X.Quan  <- levpareto(limit = pmin(uparam,Quan),
                        shape = risk2shape, scale = risk2scale)
CTE        <- Quan + (1/(1-alpha))*(Eg.X-Eg.X.Quan)

par(mfrow = c(1,3))
  plot (uparam,df, ylim=c(0.6,1), type = "l", xlab = "u")
  plot (uparam,Quan, type = "l", xlab = "u", ylab = "Value at Risk")
  plot (uparam,CTE, type = "l", xlab = "u", ylab = 'ES')
```

15.3.8 Example 2.6. Pareto Distribution and Convexity Demo

```
# Example 2.6 Illustrative Code
cparam <- seq(from = 0, to = 1, length.out = 110)
d1     <- 0
u1     <- 1000
d2     <- 1000
u2     <- 2500
dparam <- d1*cparam + d2*(1-cparam)
uparam <- u1*cparam + u2*(1-cparam)
CTE    <- 0*cparam
for (kindex in 1:length(CTE)) {
  CTE[kindex] <- ParetoCTE(d=dparam[kindex],c=1,u=uparam[kindex],
                           risk2shape = 3, risk2scale = 2000, alpha=0.85)
  }

plot (cparam, CTE, type = "l", ylab = "ES", xlab = "c", ylim=c(500,2000) )
  text(0.05, 1700, expression(d[1]==0),    cex = .8)
  text(0.05, 1600, expression(u[1]==1000), cex = .8)
  text(0.92,  800, expression(d[2]==1000), cex = .8)
  text(0.92,  700, expression(u[2]==2500), cex = .8)
  abline(a=CTE[1], b=CTE[110]-CTE[1], lty = "dashed", col =FigRed)
```

```
  arrows(0.01, 1700, 0, 1900, length = 0.1, angle = 20)
  arrows(0.98,  750, 1,  900, length = 0.1, angle = 20)
```

15.4 Chapter Three Code

15.4.1 Example 3.1. Total Cost for a Pareto Distribution

```
# Example 3.1 Illustrative Code
risk2scale <- 2000
secLoad    <- 0.2
alpha      <- 0.98
uparam     <- seq(from = 0, to = 6*ERisk2fun(), length.out = 1100)
Quan       <- pmin(q2(alpha),uparam)
Eg.X       <- actuar::levpareto(limit = uparam,shape = risk2shape,
                                scale = risk2scale)
TC         <- Quan + (1+secLoad)*(ERisk2fun() - Eg.X)
opt        <- q2(secLoad/(1+secLoad))

plot (uparam,TC,  type = "l", xlab = "u", ylim = c(0, 6000))
  abline(v=opt, col=FigBlue, lty = 2)
  abline(v=q2(alpha), col=FigGreen, lty = 3)
```

15.4.2 Example 3.2. Optimal Upper Limits Using a Pareto Distribution

```
# Example 3.2 Illustrative Code
alpha            <- 0.90
u                <- seq(100,10000,10)
xi.u             <- function(u){ParetoQuantile(d=0,c=1,u,risk2scale,
                                               risk2shape,alpha)}
premConstraint.u <- function(u){-(ParetoRTC(d=0,c=1,u) -RTC.max)}
RTCmax         <- seq(60,480,10)
Result           <- matrix(0,length(RTCmax),2)
for (i in 1:length(RTCmax)) {
  RTC.max <-  RTCmax[i]
  optsol2   <- alabama::auglag(par=200, fn=xi.u, hin=premConstraint.u,
                               control.outer=list(method="nlminb",trace=FALSE))
  Result[i,1] <- optsol2$par
  Result[i,2] <- xi.u(optsol2$par)
}
par(mfrow=c(1,2))
  plot(RTCmax,Result[,1],xlab="Maximal Risk Transfer Cost",
                         ylab="Maximum Upper Limit", type = "l")
  plot(RTCmax,Result[,2],xlab="Maximal Risk Transfer Cost",
                         ylab="Optimal Value at Risk", type = "l")
```

15.4.3 Example 3.3. Portfolio of Insurance Stock Returns

Figure 3.3

```
# Example 3.3 Illustrative Code, Figure 3.3
load(file="ChapTablesData/Chap3/Frontier.Rdata")
xy.HiMean <- c(RetSumm[1,2],RetSumm[1,1])
xy.LowStd <- c(RetSumm[4,2],RetSumm[4,1])
xy.equal  <- c(RetSumm[3,2],RetSumm[3,1])
plot(PortPerf[2,],PortPerf[1,], type="b", ylab="Portfolio Mean",
     xlab="Portfolio Std Dev",
     xlim=c(0.10, 0.27), ylim=c(0.08, 0.30))
  points(x=xy.equal[1],y=xy.equal[2], col=FigGreen)
  points(x=xy.HiMean[1],y=xy.HiMean[2],col=FigBlue)
  points(x=xy.LowStd[1],y=xy.LowStd[2], col=FigRed)
  text(x=xy.HiMean[1]+.02,y=xy.HiMean[2]-.02,
     "United Health - Highest Mean",col=FigBlue, cex=0.8)
  text(x=xy.LowStd[1]+.035,y=xy.LowStd[2],
     "Chubb - Lowest Std Dev", col=FigRed, cex=0.8)
  text(x=xy.equal[1]-.035,y=xy.equal[2]-.035,
     "Equally weighted Portfolio", col=FigGreen, cex=0.8)
  text(x=0.13,0.28, "Portfolio Frontier")
  arrows(x0=0.13, y0=0.26, x1 = 0.14, y1 = 0.24,
       length = 0.15, lwd=2, angle = 30)
  text(x=0.16,0.15, "Minimum Variance", cex=0.8)
  arrows(x0=0.136, y0=0.15, x1 = 0.122, y1 = 0.162,
       length = 0.145, lwd=1, angle = 30)
```

Create Markowitz and Naive Portfolios

```
# Example 3.3 Illustrative Code

load(file= "../ChapTablesData/Chap1/InsurSectorReturns2023.Rdata")
portolioMinVar <- function(Sigma) {
  c.prop <- Variable(nrow(Sigma))
  prob   <- Problem(Minimize(quad_form(c.prop, Sigma)),
                  constraints = list(c.prop >= 0, sum(c.prop) == 1))
  result <- solve(prob)
  c.prop <- as.vector(result$getValue(c.prop))
  names(c.prop) <- colnames(Sigma)
  return(c.prop)
}
portolioMarkowitz <- function(mu, Sigma, lmd = 10) {
  c.prop <- Variable(nrow(Sigma))
  prob   <- Problem(Maximize(t(mu) %*% c.prop -
                    lmd*quad_form(c.prop, Sigma)),
                    constraints = list(c.prop >= 0, sum(c.prop) == 1))
  result <- solve(prob)
  c.prop <- as.vector(result$getValue(c.prop))
  names(c.prop) <- colnames(Sigma)
```

```
  return(c.prop)
}
c_Markowitz <- portolioMarkowitz(mu, Sigma)
c_MinVar <- portolioMinVar(Sigma)
c_Equal <- rep(1/8,8)
c_all <- cbind("MinVar"      = c_MinVar,
               "Markowitz"   = c_Markowitz,
               "Equal"       = c_Equal)

# compute returns of all portfolios
ret_all     <- xts(Return     %*% c_all, index(Return    ))
ret_all_trn <- xts(Return_trn %*% c_all, index(Return_trn))
ret_all_tst <- xts(Return_tst %*% c_all, index(Return_tst))

temp1 <- t(table.AnnualizedReturns(Return_trn))
temp2 <- t(table.AnnualizedReturns(Return_tst))
RetSumm <- cbind(as.matrix(temp1), as.matrix(temp2))
colnames(RetSumm) <- c("Train Ann Return", "Train Std Dev",
                       "Train Sharpe (Rf=0)", "Test Ann Return",
                       "Test Std Dev", "Test Sharpe (Rf=0)")
Lambda.vec <- c(seq(from=0, to = 3, length.out = 5),
                seq(from=3.5, to = 12, length.out = 5),
                seq(from=12.5, to = 100, length.out = 5))
Portmu     <- rep(0,length(Lambda.vec)) -> Portstd
PortPerf   <- matrix(0,2,length(Lambda.vec))
for (iVec in 1:length(Lambda.vec)) {
   MarkPort        <- portolioMarkowitz(mu, Sigma, lmd = Lambda.vec[iVec])
   ret_MarkPort    <- xts(Return_trn %*% MarkPort, index(Return_trn))
   PortPerf[,iVec] <- t(table.AnnualizedReturns(ret_MarkPort)[1:2,1])
   }
temp1    <- t(table.AnnualizedReturns(ret_all_trn))
temp2    <- t(table.AnnualizedReturns(ret_all_tst))
PortSumm <- cbind(as.matrix(temp1), as.matrix(temp2))
save(PortPerf, c_all, ret_all, ret_all_trn,
     ret_all_tst, PortSumm, RetSumm,
     file="../ChapTablesData/Chap3/Frontier.Rdata")
```

Display Figure 3.4 Portfolio Allocations

```
# Example 3.3 Illustrative Code, Figure 3.4
if (HtmlEval) {barplot(t(c_all), col = rainbow10equal[1:3], legend = colnames(c_all),
        cex=0.8, args.legend = list(x = "topright", ncol = 1, cex=0.6),
        cex.names=0.6, beside = TRUE, cex.lab=0.8, main = "",
        xlab = "Insurance Stocks", ylab = "Proportions")
} else {
barplot(t(c_all),  legend = colnames(c_all),
        cex=0.8, args.legend = list(x = "topright", ncol = 1, cex=0.6),
        cex.names=0.6, beside = TRUE, cex.lab=0.8, main = "",
```

```
        xlab = "Insurance Stocks", ylab = "Proportions")
}
```

Table 3.1 Summarize Portfolio Performance Measures

```
temp1 <- t(table.AnnualizedReturns(ret_all_trn))
temp2 <- t(table.AnnualizedReturns(ret_all_tst))
PortSumm <- cbind(as.matrix(temp1), as.matrix(temp2))
colnames(PortSumm) <- c("Train Ann Return", "Train Std Dev",
                        "Train Sharpe (Rf=0)", "Test Ann Return",
                        "Test Std Dev", "Test Sharpe (Rf=0)" )
TableGen1(TableData=PortSumm,
          TextTitle='Portfolio Performance Measures',
          Align='ccccccc', Digits=3, ColumnSpec=1:6,
          BorderRight= 1, ColWidth = ColWidth6)
```

Figure 3.5

```
line_types <- c("solid", "dashed", "dotted")
grey_types <- c("grey50", "grey80","black")
if (HtmlEval)
{chart.CumReturns(ret_all_tst, main = "",  lwd = 1,
                  wealth.index = TRUE,  colorset = rainbow10equal[1:3])
addLegend("topleft", pch=0, cex=0.8,bty = "o")
} else {
  chart.CumReturns(ret_all_tst, main = "",  lwd = 1,
                  wealth.index = TRUE, col = grey_types, lty = line_types)
addLegend("topleft", pch=0, cex=0.8,lty = line_types)}
```

15.4.4 Example 3.4. Optimal Upper Limits with ES Using a Pareto Distribution

```
# Example 3.4 Illustrative Code

alpha <- 0.90
u     <- seq(100,10000,10)
xi.u  <- function(u){ParetoQuantile(d=0,c=1,u,risk2scale,risk2shape,alpha)}
premConstraint.u <- function(u){-(ParetoRTC(d=0,c=1,u) -RTC.max)}
CTE.u <- function(u){
  Quan      <- ParetoQuantile(d=0,c=1,u,risk2scale,risk2shape,alpha)
  Eg.X      <- levpareto(limit = u,shape = risk2shape, scale = risk2scale)
  Eg.X.Quan <- levpareto(limit = pmin(u,Quan),shape = risk2shape,
                         scale = risk2scale)
  CTE       <- Quan + (1/(1-alpha))*(Eg.X-Eg.X.Quan)
  return(CTE)
}
```

```
RTCmax <- seq(60,480,10)
Result   <- matrix(0,length(RTCmax),6)
for (i in 1:length(RTCmax)) {
  RTC.max <-  RTCmax[i]
  optsol2   <- alabama::auglag(par=200, fn=CTE.u, hin=premConstraint.u,
                               control.outer=list(method="nlminb",trace=FALSE))
  Result[i,1] <- optsol2$par
  Result[i,2] <- xi.u(optsol2$par)
  Result[i,3] <- CTE.u(optsol2$par)
  optsol3 <- alabama::auglag(par=200, fn=xi.u, hin=premConstraint.u,
                             control.outer=list(method="nlminb",trace=FALSE))
  Result[i,4] <- optsol3$par
  Result[i,5] <- xi.u(optsol3$par)
  Result[i,6] <- CTE.u(optsol3$par)
}
plot(RTCmax,Result[,3],xlab="Maximal Risk Transfer Cost",
                         ylab="Optimal $ES$", type = "l" )
  abline(h=q2(alpha), col=FigGreen, lty = 3, lwd=4.0)
```

15.4.5 Section 3.5.1. Comparing VaR to ES Using a Pareto Distribution

```
#  Figure 3.7
par(mfrow=c(1,3))
  plot(Result[,1],Result[,4], xlab="ES optimal - u",
                              ylab="VaR optimal - u")
    abline(0,1)
    abline(v=q2(alpha), col=FigRed, lty = 2)
  plot(Result[,2],Result[,5], xlab="ES optimal - VaR",
                              ylab="VaR optimal - VaR")
    abline(0,1)
    abline(v=q2(alpha), col=FigRed, lty = 2)
  plot(Result[,3],Result[,6], xlab="ES optimal - ES",
                              ylab="VaR optimal - ES")
    abline(0,1)
    abline(v=q2(alpha), col=FigRed, lty = 2)
```

15.4.6 Section 3.5.2. Deductibles and Upper Limits

```
#  Figure 3.8
# Determine optimal u in terms of d
RTC.max1   <- 400
up.bdd.d.fct <- function(d){levpareto(limit = d,   shape = risk2shape,
                                      scale = risk2scale)-RTC.max1}
d.max <- pmin(q2(alpha),uniroot(up.bdd.d.fct, lower = 0,
                                              upper = 10e23)$root)
dVec  <- seq(0, d.max, length.out = 21)
uVec  <- 0 * dVec
```

```
CTEMatrix <- matrix(0,nrow = length(dVec),ncol = 4)
for (id in 1:length(dVec)) {
  CTEMatrix[id,1] <- dVec[id]
  CTEMatrix[id,2] <-  RTC.max1
  uVec.fct <- function(u){ParetoRTC(d=dVec[id],c=1,u,risk2scale=1000,
                                    risk2shape=3)-RTC.max1}
   if ( id < 21) {temp <- uniroot(uVec.fct, lower = 0, upper = 10e23)$root }
  else {temp <- 10e23}
  CTEMatrix[id,3] <- temp
  CTEMatrix[id,4] <- ParetoCTE(d=dVec[id],c=1,u=CTEMatrix[id,3],
                               risk2scale=1000, risk2shape=3,alpha = 0.90)
 }

RTCmaxVec <- seq(10, 410, length.out = 21)
CTEmaxMatrix <- matrix(0,nrow = length(RTCmaxVec),ncol = 3)
for (iv in 1:length(RTCmaxVec)) {
  RTCmax <- RTCmaxVec[iv] -> CTEmaxMatrix[iv,1]
  uVec.fct <- function(u){ParetoRTC(d=0,c=1,u,risk2scale=1000,
                                    risk2shape=3)-RTCmax}
  if ( iv < 21) {temp <- uniroot(uVec.fct, lower = 0,
                                           upper = 10e23)$root }
 else {temp <- 10e23}
CTEmaxMatrix[iv,2] <- temp
CTEmaxMatrix[iv,3] = ParetoCTE(d=0,c=1,u=CTEmaxMatrix[iv,2],risk2scale=1000,
                               risk2shape=3,alpha = 0.90)
}

par(mfrow=c(2,2))
  plot(head(CTEMatrix[,1],-2),head(CTEMatrix[,3],-2),
       ylab = expression(u^'*'),
       xlab = expression(d^'*'), type = "b")
  plot(head(CTEMatrix[,1],-2),head(CTEMatrix[,4],-2), ylab = "ES",
       xlab = expression(d^'*'), type = "b")
  plot(head(CTEmaxMatrix[,1],-2),head(CTEmaxMatrix[,2],-2),
       ylab = expression(u^'*'),
       xlab = expression(RTC['max']), type = "b")
  plot(head(CTEmaxMatrix[,1],-2),head(CTEmaxMatrix[,3],-2),
       ylab = expression(ES^'*'),
       xlab = expression(RTC['max']), type = "b")
```

15.5 Chapter Four Code

15.5.1 Example 4.1. Quota Sharing of Three Pareto Risks

Figure 4.1

```
# Example 4.1 Illustrative Code
theta1 <- 1000; theta2 <- 2000; theta3 <- 3000;
alpha1 <- 3; alpha2 = 3; alpha3 <- 4;
mu.vec  <-  c(actuar::mpareto(shape=alpha1, scale=theta1, order=1),
              actuar::mpareto(shape=alpha2, scale=theta2, order=1),
              actuar::mpareto(shape=alpha3, scale=theta3, order=1))
var.vec <-  c(actuar::mpareto(shape=alpha1, scale=theta1, order=2),
              actuar::mpareto(shape=alpha2, scale=theta2, order=2),
              actuar::mpareto(shape=alpha3, scale=theta3, order=2)) - mu.vec**2
sum.msvar <- mu.vec[1]**2/var.vec[1] +
             mu.vec[2]**2/var.vec[2]+ mu.vec[3]**2/var.vec[3]

KVec      <- seq(100, 2500, length.out=25)
Lambdavec <- 2*KVec/sum.msvar
c1Vec     <- c2Vec <- c3Vec <- 0*KVec
for (j in 1:25) {
  c1Vec[j] <- (Lambdavec[j]/2) * mu.vec[1]/var.vec[1]
  c2Vec[j] <- (Lambdavec[j]/2) * mu.vec[2]/var.vec[2]
  c3Vec[j] <- (Lambdavec[j]/2) * mu.vec[3]/var.vec[3]
  }
RCostMax <- sum(mu.vec) - KVec
plot(RCostMax , c1Vec, type="l", ylab="proportion",
     xlab="Maximal Risk Transfer Cost", ylim=c(0,1.2))
  lines(RCostMax , c2Vec)
  lines(RCostMax , c3Vec)
  text(1200,0.85, expression(c[1]))
  text(500,0.75, expression(c[3]))
  text(1200,0.30, expression(c[2]))
```

Follow Up to Example 4.1 - Determining Optimal Proportions

```
# Example 4.1 Illustrative Code, Follow Up
library(CVXR)
KVec       <- seq(2000, 2000, length.out=1)
RCostMax   <- sum(mu.vec) - KVec
C.vec      <- Variable(3)
Exp.ins    <-  C.vec[1]*mu.vec[1] + C.vec[2]*mu.vec[2] + C.vec[3]*mu.vec[3]
objective  <-  (C.vec[1]**2)*var.vec[1] +
               (C.vec[2]**2)*var.vec[2] + (C.vec[3]**2)*var.vec[3]
constraints <- list(sum(mu.vec) - Exp.ins == RCostMax[1])
prob       <- Problem(Minimize(objective), constraints)
result     <- solve(prob)
```

```
value1     <- result$value

# Check the hand formula
#result$getValue(C.vec)
#c1Vec[20];c2Vec[20];c3Vec[20]
#  They match ...

# Constrain coefficients to be between 0 and 1
C.vec       <- Variable(3)
Exp.ins     <-  C.vec[1]*mu.vec[1] + C.vec[2]*mu.vec[2] + C.vec[3]*mu.vec[3]
objective   <-  (C.vec[1]**2)*var.vec[1] +
               (C.vec[2]**2)*var.vec[2] + (C.vec[3]**2)*var.vec[3]
constraints <- list(sum(mu.vec) - Exp.ins == RCostMax[1],
                    C.vec >= 0, C.vec <= 1)
prob        <- Problem(Minimize(objective), constraints)
result      <- solve(prob)
cvec.Constrained <- result$getValue(C.vec)
value2      <- result$value
```

15.5.2 Example 4.2. Surplus Share Retention for the Property Fund

```
# Example 4.2 Illustrative Code

load(file = "Data/MultiFreqSevExample/dataout.RData")
retainedLine <- 100
MaxRetain    <- 6*retainedLine
InsAmt       <- subset(dataout$CoverageBC, dataout$CoverageBC > 0)
InsAmt       <- InsAmt[order(InsAmt)]
NumCov       <- as.numeric(length(InsAmt))
NumFullyRet  <- as.numeric(sum(InsAmt<retainedLine))
LargeCov     <- sum(InsAmt>=MaxRetain)
NumInt       <- NumCov - NumFullyRet- LargeCov

reins.prop   <- pmax((InsAmt - retainedLine)/InsAmt,0)
reins.prop   <- pmin(reins.prop, (MaxRetain - retainedLine)/InsAmt)
ins.prop     <- 1 - reins.prop
plot(InsAmt,reins.prop, ylab = "c(IA)", xlab="Insurance Amount (IA)",
     ylim=c(0,1), type = "b", lty = 3)
```

15.5.3 Example 4.3. Excess of Loss for Three Pareto Risks

Determine the Optimal Upper Limits

```
# Example 4.3 Illustrative Code
theta1 <- 1000;theta2 <- 2000;theta3 <- 3000;
alpha1 <- 3;   alpha2 <- 3;   alpha3 <- 4;
Pmin   <- 2000
```

```
VarFct <- function(M){
  M1=M[1];M2=M[2];M3=M[3]
  mu1    <- actuar::levpareto(limit=M1,shape=alpha1, scale=theta1, order=1)
  var1   <- actuar::levpareto(limit=M1,shape=alpha1, scale=theta1, order=2) - mu1^2
  mu2    <- actuar::levpareto(limit=M2,shape=alpha2, scale=theta2, order=1)
  var2   <- actuar::levpareto(limit=M2,shape=alpha2, scale=theta2, order=2) - mu2^2
  mu3    <- actuar::levpareto(limit=M3,shape=alpha3, scale=theta3, order=1)
  var3   <- actuar::levpareto(limit=M3,shape=alpha3, scale=theta3, order=2) - mu3^2
  varFct <- var1 + var2 + var3
  meanFct <- mu1 + mu2 + mu3
  c(meanFct,varFct)
  }
f    <- function(M){VarFct(M)[2]}
h    <- function(M){VarFct(M)[1] - Pmin}
par0 <- rep(1000,3)
op   <- alabama::auglag(par=par0,fn=f,hin=h,control.outer=list(trace=FALSE))
```

Check that the Excess is the Same for all Three Risks

```
M1star <- op$par[1];M2star <- op$par[2];M3star <- op$par[3]
#M1star - actuar::levpareto(M1star,shape=alpha1, scale=theta1,order=1)
# [1] 1344.135
#M2star - actuar::levpareto(M2star,shape=alpha2, scale=theta2,order=1)
# [1] 1344.135
#M3star - actuar::levpareto(M3star,shape=alpha3, scale=theta3,order=1)
# [1] 1344.135
```

Figure 4.3

```
# Example 4.3 Illustrative Code, Figure 4.3
set.seed(2018)
nSim     <- 10000
M1star   <- op$par[1];M2star <- op$par[2];M3star <- op$par[3]
Y1       <- actuar::rpareto(nSim, shape = alpha1, scale = theta1)
Y2       <- actuar::rpareto(nSim, shape = alpha2, scale = theta2)
Y3       <- actuar::rpareto(nSim, shape = alpha3, scale = theta3)
YTotal   <- Y1 + Y2 + Y3
Yinsur   <-  pmin(Y1,M1star)+pmin(Y2,M2star)+pmin(Y3,M3star)
Yreinsur <- YTotal - Yinsur

par(mfrow=c(1,3))
  plot(density(YTotal),   xlim=c(0,10000), main="Total Loss", xlab="Losses")
  plot(density(Yinsur),   xlim=c(0,10000), main="Insurer",    xlab="Losses")
  plot(density(Yreinsur), xlim=c(0,10000), main="Reinsurer",  xlab="Losses")
```

15.5.4 Example 4.6. Standard Deviation of an Excess of Loss Policy with Two Risks

Simulate and Determine the Variance

```
# Example 4.6 Illustrative Code
#  Set Simulation Parameters
nsim <- 20000
set.seed(202020)
#  Normal, or Gaussian, Copula
norm.cop <- copula::normalCopula(param=0.3, dim = 2,'dispstr' ="un")
UCop     <- copula::rCopula(nsim, norm.cop)
#  Marginal Distributions
X1       <- qgamma(p=UCop[,1],shape = risk1shape, scale = risk1scale)
X2       <- actuar::qpareto(p=(UCop[,2]),shape = risk2shape, scale = risk2scale)
#  Objective Function
f0 <- function(u1,u2){
      S <- pmin(X1,u1) +  pmin(X2,u2)
      return( var(S) )
      }
```

Figure 4.4

```
# Example 4.6 Illustrative Code, Figure 4.4
#  Plot the Functions
u1Vec <- seq(10, qgamma(p=0.90,shape = risk1shape, scale = risk1scale), length.out=20)
u2Vec <- seq(10, actuar::qpareto(p=0.90,shape = risk2shape,
                                 scale = risk2scale), length.out=20)
f0Mat <- matrix(0,20,20)
for (i in 1:20)
  {for (j in 1:20) {
  f0Mat[i,j] <- f0(u1=u1Vec[i],u2=u2Vec[j])
  }}
par(mfrow=c(1,2))
  plot(u1Vec, sqrt(f0Mat[,1]), xlab = expression(u[1]),
       ylab = "std dev", type = "l")
    for (j in 2:20) {lines(u1Vec, sqrt(f0Mat[,j]))}
  plot(u2Vec, sqrt(f0Mat[1,]), xlab = expression(u[2]),
       ylab = "std dev", ylim=c(0,6000), type = "l")
    for (j in 2:20) {lines(u2Vec, sqrt(f0Mat[j,]))}
```

15.5.5 Example 4.7. Effects of Dependence on Portfolio Summary Measures

```
# Example 4.7 Illustrative Code
# For the gamma distributions, use
alpha1 <- 2;      theta1 <- 100
alpha2 <- 2;      theta2 <- 200
```

```
# For the Pareto distributions, use
alpha3 <- 2;      theta3 <- 1000
alpha4 <- 3;      theta4 <- 2000
# Deductibles
d1     <- 100
d2     <- 200

# Simulate the risks
nSim <- 500000  #number of simulations
set.seed(2017) #set seed to reproduce work
X1 <- rgamma(nSim,alpha1,scale = theta1)
X2 <- rgamma(nSim,alpha2,scale = theta2)
X3 <- actuar::rpareto(nSim,scale=theta3,shape=alpha3)
X4 <- actuar::rpareto(nSim,scale=theta4,shape=alpha4)
# Portfolio Risks
S             <- X1 + X2 + X3 + X4
Sretained     <- pmin(X1,d1) + pmin(X2,d2)
Sreinsurer    <- S - Sretained
OutMat        <- matrix(0, nrow = 6, ncol = 5)
OutMat[1:3,1] <- t(as.matrix(c(mean(Sretained),mean(Sreinsurer),mean(S))))
OutMat[1:3,2] <- t(as.matrix(c(sd(Sretained),sd(Sreinsurer),sd(S))))
OutMat[1,3:5] <- quantile(Sretained,  probs=c( 0.90, 0.95, 0.99))
OutMat[2,3:5] <- quantile(Sreinsurer, probs=c( 0.90, 0.95, 0.99))
OutMat[3,3:5] <- quantile(S        ,  probs=c( 0.90, 0.95, 0.99))

### Normal Copula ##
set.seed(2017)
parm <- c(0.5,0.5,0.5,0.5,0.5,0.5)*0
nc   <- copula::normalCopula(parm, dim = 4, dispstr = "un")
mcc  <- copula::mvdc(nc, margins = c("gamma", "gamma","paretoII","paretoII"),
                     paramMargins = list(list(scale = theta1, shape=alpha1),
                                         list(scale = theta2, shape=alpha2),
                                         list(scale = theta3, shape=alpha3),
                                         list(scale = theta4, shape=alpha4) ) )
X <- copula::rMvdc(nSim, mvdc = mcc)  # Another way to simulate
X1<-X[,1]
X2<-X[,2]
X3<-X[,3]
X4<-X[,4]

# Portfolio Risks
S            <- X1 + X2 + X3 + X4
Sretained   <- pmin(X1,d1) + pmin(X2,d2)
Sreinsurer  <- S - Sretained

OutMat[4:6,1] <- t(as.matrix(c(mean(Sretained),mean(Sreinsurer),mean(S))))
OutMat[4:6,2] <- t(as.matrix(c(sd(Sretained),sd(Sreinsurer),sd(S))))
OutMat[4,3:5] <- quantile(Sretained,  probs=c( 0.90, 0.95, 0.99))
OutMat[5,3:5] <- quantile(Sreinsurer, probs=c( 0.90, 0.95, 0.99))
OutMat[6,3:5] <- quantile(S        ,  probs=c( 0.90, 0.95, 0.99))
```

```
rownames(OutMat) <- c("Retained (Ind)", "Reinsurer (Ind)","Total (Ind)",
                      "Retained (Dep)", "Reinsurer (Dep)","Total (Dep)")
colnames(OutMat) <- c("Mean", "Std Dev","$VaR_{0.90}$", "$VaR_{0.95}$", "$VaR_{0.99}$")

TableGen1(TableData=OutMat,
         TextTitle='Portfolio Expected Values,
                    Standard Deviations, and Quantiles ($VaR_{\\alpha}$)',
         Align='r', Digits=0, ColumnSpec=1:5,
         BorderRight=1, ColWidth = ColWidth5,
         ColWidth0 = "4cm")
```

15.5.6 Example 4.8. Quota Share Example with Three Risks

```
# Example 4.8 Illustrative Code, More Set Up
# Treat the correlations as Pearson correlations, Exercise....
param      <- c(-0.2, 0.8, -0.3)
param      <- c(0,0,0)
risk2shape <- 3
risk2scale <- 2000
ExpMat     <- cbind(ERisk1fun(),ERisk2fun(),ERisk3fun())
stdevVec   <- c(sdRisk1fun(),sdRisk2fun(),sdRisk3fun())
#  Dependency structure
rho12  <- param[1]
rho13  <- param[2]
rho23  <- param[3]
BigRho <- matrix(c(1, rho12, rho13, rho12, 1,
                     rho23, rho13, rho23, 1),nrow = 3, ncol = 3)
BigSigma    <- diag(stdevVec) %*% BigRho %*% diag(stdevVec)
InvSigMat   <- solve(BigSigma)
XSigX       <- ExpMat %*% InvSigMat %*% t(ExpMat)
ELoss.Quota <- ERisk1fun()+ERisk2fun()+ERisk3fun()
RCOSTmax    <- 0.2*ELoss.Quota
LM.Quota    <- 2*(ELoss.Quota-RCOSTmax)/XSigX
c           <- (as.numeric(LM.Quota)/2)*InvSigMat %*% t(ExpMat)
c(LM.Quota,c)
```

```
# Example 4.8 Illustrative Code
risk2shape  <- 3
risk2scale  <- 2000
#  Dependency structure
rho12    <- param[1]
rho13    <- param[2]
rho23    <- param[3]
BigSigma <- matrix(c(1, rho12, rho13, rho12, 1,
                     rho23, rho13, rho23, 1),nrow = 3, ncol = 3)
# Generating Dependent Lapses and Claims
nsim      <- 20000
set.seed(202020)
```

```
norm.cop  <- copula::normalCopula(param=param, dim = 3,'dispstr' ="un")
UCop      <- copula::rCopula(nsim, norm.cop)
X1        <- q1(UCop[,1])
X2 <- actuar::qpareto(p=UCop[,2],shape = risk2shape, scale = 10000)
X3        <- q3(UCop[,3])
Xmat      <- cbind(X1,X2,X3)
CorMat    <- cor(Xmat)
SigMat    <- cov(Xmat)
InvSigMat <- solve(SigMat)
ExpMat    <- cbind(ERisk1fun(),ERisk2fun(),ERisk3fun())
stdevVec  <- c(sdRisk1fun(),sdRisk2fun(),sdRisk3fun())
#round(sqrt(diag(SigMat)),digits=0)
ELoss.Quota <- ERisk1fun()+ERisk2fun()+ERisk3fun()
RCOSTmax <- 0.2*ELoss.Quota

# K = ELoss.Quota - RCOSTmax
XSigX    <- ExpMat %*% InvSigMat %*% t(ExpMat)
LM.Quota <- 2*(ELoss.Quota-RCOSTmax)/XSigX
c        <- (as.numeric(LM.Quota)/2)*InvSigMat %*% t(ExpMat)
#  Collect Terms for Output
outEx48       <- matrix(0, nrow=3, ncol=6)
outEx48[,1]   <- ExpMat
outEx48[,2]   <- round(stdevVec,digits=0)
outEx48[,3:5] <- round(BigSigma ,digits=3)
outEx48[,6]   <- round(c,digits=3)
rownames(outEx48) <- c("Gamma1", "Pareto", "Gamma2")
colnames(outEx48) <- c("Mean", "Std Dev", "Corr 1","Corr 2","Corr 3", "c")
```

15.5.7 Example 4.9. Optimal Upper Limits with Excess of Loss for Three Dependent Risks

```
# Example 4.9 Illustrative Code
param      <- c(0.95, 0.0, 0.0)
risk2shape <- 3
risk2scale <- 2000
ELoss.XL  <- ERisk1fun()+ERisk2fun()+ERisk3fun()
RCOSTmax  <- 0.2*ELoss.XL
K         <- ELoss.XL - RCOSTmax
paramM    <- c(1.1*ERisk1fun(),1.2*ERisk2fun(),1.1*ERisk3fun())
H11       <- function(u){integrate(F1, lower = 0, upper = u)$value }
H12       <- function(u){integrate(F2, lower = 0, upper = u)$value }
H13       <- function(u){integrate(F3, lower = 0, upper = u)$value }
H1        <- function(u){u-H11(u)}
H2        <- function(u){u-H12(u)}
H3        <- function(u){u-H13(u)}
# Inverse Functions
H11inv <- function(lambda){H11star <- function(x){H11(x)-lambda}
          return(uniroot(H11star, lower = 0, upper = UpRangeRisk1)$root)}
H12inv <- function(lambda){H12star <- function(x){H12(x)-lambda}
```

```
          return(uniroot(H12star, lower = 0, upper = UpRangeRisk2)$root)}
H13inv <- function(lambda){H13star <- function(x){H13(x)-lambda}
          return(uniroot(H13star, lower = 0, upper = UpRangeRisk3)$root)}
IndExLoss <- function(lambda){
             H11inv(lambda/2) + H12inv(lambda/2)  +
             H13inv(lambda/2) - 3*lambda/2 - K
             }
IndExLoss <- function(lambda){
             H11inv(lambda/2) + H12inv(lambda/2)  +
             H13inv(lambda/2) - 3*lambda/2 - (ELoss.XL-RCOSTmax)
             }
lambda.opt <- uniroot(IndExLoss, lower = 0, upper = 9e5)$root
#  Gradient Functions
g.1 <- function(MParms){
       M1 <- MParms[1];M2 <- MParms[2];M3 <- MParms[3];lambda <- MParms[4]
       return(2*H11(M1) - lambda)
       }
g.2 <- function(MParms){
       M1 <- MParms[1];M2 <- MParms[2];M3 <- MParms[3];lambda <- MParms[4]
       return(2*H12(M2) - lambda)
       }
g.3 <- function(MParms){
       M1 <- MParms[1];M2 <- MParms[2];M3 <- MParms[3];lambda <- MParms[4]
       return(2*H13(M3) - lambda)
       }
g.4 <- function(MParms){
       M1 <- MParms[1];M2 <- MParms[2];M3 <- MParms[3];lambda <- MParms[4]
       return(H1(M1)+H2(M2)+H3(M3) - (ELoss.XL-RCOSTmax) )
       }

model <- function(MParms) {
         c(F1=g.1(MParms),F2=g.2(MParms),
         F3=g.3(MParms),F4=g.4(MParms))}
XL    <- rootSolve::multiroot(f = model, start = c(1000,1000,1000,1000))
sol.ind.params <- round(XL$root, digits=0)

#########################################################################
# Now try Dependent case
#param <- c(-0.2, 0.3, -0.3)
#param <- c(0,0,0)
norm.cop   <- copula::normalCopula(param=param, dim = 3,'dispstr' ="un")
norm.cop12 <- copula::normalCopula(param=as.numeric(param[1]), dim = 2)
norm.cop13 <- copula::normalCopula(param=as.numeric(param[2]), dim = 2)
norm.cop23 <- copula::normalCopula(param=as.numeric(param[3]), dim = 2)

F.MX <- function(arg){
  MParms <- arg[1:4];i <- arg[5]; j<- arg[6]
  M1 <- MParms[1];M2 <- MParms[2];M3 <- MParms[3];lambda <- MParms[4]
  if ( (i==1)*(j==2) + (i==2)*(j==1) ) {norm.cop<- norm.cop12}
  if ( (i==1)*(j==3) + (i==3)*(j==1) ) {norm.cop<- norm.cop13}
  if ( (i==2)*(j==3) + (i==3)*(j==2) ) {norm.cop<- norm.cop23}
```

```
  FiMi <- F1(M1)*(i==1) + F2(M2)*(i==2) + F3(M3)*(i==3)
  JointF_MX <- function(z){
    F1jz <- F1(z)*(j==1) + F2(z)*(j==2) + F3(z)*(j==3)
    return(FiMi-pCopula(c(FiMi,F1jz), norm.cop))
    }
  VecJointF_MX <- Vectorize(JointF_MX)
  return(integrate(VecJointF_MX, lower = 0, upper = MParms[j])$value)
  }
#  Gradient Functions
g.1.dep <- function(MParms){
  M1 <- MParms[1];M2 <- MParms[2];M3 <- MParms[3];lambda <- MParms[4]
  eqeval <- 2*( ELoss.XL - RCOSTmax-H1(M1) ) -
    2*( F.MX(c(MParms,1,2)) + F.MX(c(MParms,1,3)) ) +
    ( 2*M1-2*(ELoss.XL-RCOSTmax)-lambda)*(1-F1(M1) )
  return(eqeval)
  }
g.2.dep <- function(MParms){
  M1 <- MParms[1];M2 <- MParms[2];M3 <- MParms[3];lambda <- MParms[4]
  eqeval <- 2*(ELoss.XL-RCOSTmax-H2(M2)) -
            2*(F.MX(c(MParms,2,1))+F.MX(c(MParms,2,3))) +
            (2*M2-2*(ELoss.XL-RCOSTmax)-lambda)*(1-F2(M2))
  return(eqeval)
  }
g.3.dep <- function(MParms){
  M1 <- MParms[1];M2 <- MParms[2];M3 <- MParms[3];lambda <- MParms[4]
  eqeval <- 2*(ELoss.XL-RCOSTmax-H3(M3)) -
            2*(F.MX(c(MParms,3,1))+F.MX(c(MParms,3,2))) +
            (2*M3-2*(ELoss.XL-RCOSTmax)-lambda)*(1-F3(M3))
  return(eqeval)
  }
g.4 <- function(MParms){
  M1 <- MParms[1];M2 <- MParms[2];M3 <- MParms[3];lambda <- MParms[4]
  eqeval <- H1(M1)+H2(M2)+H3(M3) - (ELoss.XL-RCOSTmax )
  return(eqeval)
  }
model <- function(MParms) {c(F1=g.1.dep(MParms),F2=g.2.dep(MParms),
                             F3=g.3.dep(MParms),F4=g.4(MParms))}
XL.dep <- multiroot(f = model, start = sol.ind.params)
sol.dep.params <- round(XL.dep$root, digits=0)
ERiskVec <- c(ERisk1fun(),ERisk2fun(),ERisk3fun(), "")
ind.Frac <- c(round(c(F1(sol.ind.params[1]),F2(sol.ind.params[2]),
                      F3(sol.ind.params[3])), digits=3), "")
dep.Frac <- c(round(c(F1(sol.dep.params[1]),F2(sol.dep.params[2]),
                      F3(sol.dep.params[3])), digits=3), "")
ExLossDepOut <- cbind(ERiskVec,sol.ind.params,
                      ind.Frac,sol.dep.params,dep.Frac)
rownames(ExLossDepOut) <- c("Gamma1", "Pareto", "Gamma2","LME")
colnames(ExLossDepOut) <- c("Mean", "Optimal Ind u","Ind u Prop",
                            "Optimal Dep u","Dep u Prop")
save(sol.ind.params, sol.dep.params, ExLossDepOut,
     file="../ChapTablesData/Chap4/Example433.Rdata")
```

15.5.8 Example 4.10. Optimizing Portfolio of Insurance Stock Returns

Optimizing the Portfolio

```
# Example 4.10 Illustrative Code
load(file= "ChapTablesData/Chap1/InsurSectorReturns2023.Rdata")
load(file="ChapTablesData/Chap3/Frontier.Rdata")
portolioCTE <- function(X, lmd = 0.5, alpha = 0.95) {
  T <- nrow(X)
  p.num <- ncol(X)
  X <- as.matrix(X)
  mu <- colMeans(X, na.rm = FALSE)
  # variables
  w <- Variable(p.num)
  z <- Variable(T)
  zeta <- Variable(1)
  # problem
  prob <- Problem(Maximize(t(w) %*% mu - lmd*zeta - (lmd/(T*(1-alpha))) * sum(z)),
                  constraints = list(z >= 0, z >= -X %*% w - zeta,
                                     w >= 0, sum(w) == 1))
  result <- solve(prob)
  return(as.vector(result$getValue(w)))
  }

c_CTE095 <- portolioCTE(Return_trn, alpha = 0.95)
c_CTE099 <- portolioCTE(Return_trn, alpha = 0.99)

# combine portfolios
c_all2 <- cbind(c_all,
                "ES-$\\alpha = 0.95$" = c_CTE095,
                "ES-$\\alpha = 0.99$" = c_CTE099)

ret_all     <- xts(Return     %*% c_all2, index(Return    ))
ret_all_trn <- xts(Return_trn %*% c_all2, index(Return_trn))
ret_all_tst <- xts(Return_tst %*% c_all2, index(Return_tst))
```

Table 4.6

```
# Example 4.10 Illustrative Code, Table 4.6
temp1 <- t(table.AnnualizedReturns(ret_all_trn))
temp2 <- t(table.AnnualizedReturns(ret_all_tst))
PortSumm2 <- cbind(as.matrix(temp1), as.matrix(temp2))
colnames(PortSumm2) <- c("Train Ann Return", "Train Std Dev",
                         "Train Sharpe (Rf=0)", "Test Ann Return",
                         "Test Std Dev", "Test Sharpe (Rf=0)")
```

```
TableGen1(TableData=PortSumm2,
         TextTitle='Portfolio Performance Measures, with $ES$',
         Align='ccccccc', Digits=3, ColumnSpec=1:6,
         ColWidth0= "2.5cm",
         BorderRight= c(1,4), ColWidth = ColWidth6)
```

15.5.9 Example 4.11. Constrained Optimization for Excess of Loss with Two Risks

Evaluate the Constraint Function

```
# Example 4.11 Illustrative Code,  Constraint Function
f1in <- function(u1,u2){
  TotalCost <- actuar::mgamma(order=1, shape = risk1shape, scale = risk1scale) +
               actuar::mpareto(order=1, shape = risk2shape, scale = risk2scale)
  TotalRetained <-
           actuar::levgamma(limit = u1, shape = risk1shape, scale = risk1scale) +
           actuar::levpareto(limit = u2,shape = risk2shape, scale = risk2scale)
  TransCost <- TotalCost - TotalRetained
  return(TransCost)
  }
```

Plot Figure 4.6

```
# Example 4.11 Illustrative Code,  Plot the Functions
u1Vec <- seq(10, qgamma(p=0.90,shape = risk1shape, scale = risk1scale),
             length.out=20)
u2Vec <- seq(10, actuar::qpareto(p=0.90,shape = risk2shape,
                                 scale = risk2scale), length.out=20)
f0Mat <- matrix(0,20,20) -> f1inMat
for (i in 1:20)
  {for (j in 1:20) {
  f0Mat[i,j] <- f0(u1=u1Vec[i],u2=u2Vec[j])
  f1inMat[i,j] <- f1in(u1=u1Vec[i],u2=u2Vec[j])
  }}
par(mfrow=c(1,3))
  plot(u1Vec, f1inMat[,1], xlab = expression(u[1]),ylab = "RTC")
    for (j in 2:20) {lines(u1Vec, f1inMat[,j])}
  plot(u2Vec, f1inMat[1,], xlab = expression(u[2]),ylab = "RTC", ylim = c(0,11000))
    for (j in 2:20) {lines(u2Vec, f1inMat[j,])}
  plot(f1inMat[,1], sqrt(f0Mat[,1]),ylab = "std dev",xlab = "RTC")
    for (j in 2:20) {lines(f1inMat[,j], sqrt(f0Mat[,j]))}
```

Optimization without Derivatives

```
# Example 4.11 Illustrative Code, Without Derivatives A
RTCUpper <- actuar::mgamma(order=1, shape = risk1shape, scale = risk1scale) +
            actuar::mpareto(order=1, shape = risk2shape, scale = risk2scale) -1
#  Start at mean values
starter = c(actuar::mgamma(order=1, shape = risk1shape, scale = risk1scale),
            actuar::mpareto(order=1, shape = risk2shape, scale = risk2scale))
f0.ala <- function(par){f0(u1=par[1], u2=par[2])}

OptMat <- matrix(0,20,4)
RTCVec <- seq(10, RTCUpper, length.out=20)
```

```
# Example 4.11 Illustrative Code, Without Derivatives B
for (kindex in 1:length(RTCVec)) {
  f1.ala <- function(par){-(f1in(u1=par[1],u2=par[2])-RTCVec[kindex])}
  hin <- function(par) {h <- NA
    h[1] <- f1.ala(par)
    h[2] <- par[1]  # non-negativity constraint on first parameter
    h[3] <- par[2]
    return(h)}
  var.opt <- alabama::auglag(par=starter,     # initial value
             fn=f0.ala,                       # objective function
             hin=hin,                         # inequality constraint
             control.outer=list(method="nlminb",trace=FALSE))
  OptMat[kindex,1:2] <- var.opt$par
  OptMat[kindex,3]   <- f0(var.opt$par[1],var.opt$par[2])
  OptMat[kindex,4]   <- f1in(var.opt$par[1],var.opt$par[2])
  }

save(RTCVec,OptMat, file = "../ChapTablesData/Chap4/Ex431.RData")
```

Optimization with Derivatives

```
# Example 4.11 Illustrative Code, With Derivatives
library(alabama)
F1 <- function(x){pgamma(q=x,shape = risk1shape, scale = risk1scale)}
F2 <- function(x){actuar::ppareto(q=x,shape = risk2shape, scale = risk2scale)}

f0.grad1 <- function(u1,u2){
  Fu1.x2 <- function(x2){F2(x2)-pCopula(c(F1(u1),F2(x2)), norm.cop)}
  Fu1.x2.vec <- Vectorize(Fu1.x2)
  H11 <- function(u2){integrate(Fu1.x2.vec, lower = 0, upper = u2)$value }
  part <- u1+u2 - (actuar::levgamma(limit = u1, shape = risk1shape,
                                    scale = risk1scale) +
            actuar::levpareto(limit = u2,shape = risk2shape,
                              scale = risk2scale))
  partial1 <- 2*(part * (1-F1(u1))  - H11(u2))
  return(partial1)
  }
```

```
f0.grad2 <- function(u1,u2){
  Fx1.u2 <- function(x1){F1(x1)-pCopula(c(F1(x1),F2(u2)), norm.cop)}
  Fx1.u2.vec <- Vectorize(Fx1.u2)
  H22 <- function(u1){integrate(Fx1.u2.vec , lower = 0, upper = u1)$value }
  part <- u1+u2 -
    (actuar::levgamma(limit = u1, shape = risk1shape, scale = risk1scale) +
     actuar::levpareto(limit = u2,shape = risk2shape,scale = risk2scale))
  partial2 <- 2*( part * (1-F2(u2))  - H22(u1) )
  return(partial2)
}
  gr <- function(par) {gr <- NA
  gr[1] <- f0.grad1(u1=par[1],u2=par[2])
# non-negativity constraint on first parameter
  gr[2] <- f0.grad2(u1=par[1],u2=par[2])
  return(gr)
  }

OptMat <- matrix(0,20,4)
RTCVec <- seq(10, RTCUpper, length.out=20)
for (kindex in 1:length(RTCVec)) {
  f1.ala <- function(par){-(f1in(u1=par[1],u2=par[2])-RTCVec[kindex])}
  hin <- function(par) {h <- NA
    h[1] <- f1.ala(par)
    h[2] <- par[1]  # non-negativity constraint on first parameter
    h[3] <- par[2]
    return(h)}
  var.opt <- alabama::auglag(par=starter, # initial value
            fn=f0.ala,           # objective function
            gr=gr,               # gradient of the objective function
            hin=hin,             # inequality constraint
            control.outer=list(method="nlminb",trace=FALSE))
  OptMat[kindex,1:2] <- var.opt$par
  OptMat[kindex,3]   <- f0(var.opt$par[1],var.opt$par[2])
  OptMat[kindex,4]   <- f1in(var.opt$par[1],var.opt$par[2])
}
round(OptMat, digits=2)
```

15.5.10 Section 4.4.4 Root-Finding Methods (with Derivatives)

```
# Example 4.11 Illustrative Code, Root Finding
library(rootSolve)

param      <- c(0.95, 0.0, 0.0)
norm.cop   <- copula::normalCopula(param=param, dim = 3,'dispstr' ="un")
norm.cop12 <- copula::normalCopula(param=as.numeric(param[1]), dim = 2)
f0.grad1   <- function(u1,u2,LM){
  Fu1.x2   <- function(x2){F2(x2)-pCopula(c(F1(u1),F2(x2)), norm.cop12)}
  Fu1.x2.vec <- Vectorize(Fu1.x2)
  H11      <- function(u2){integrate(Fu1.x2.vec, lower = 0,
```

```
                                upper = u2)$value }
  part     <- u1+u2 -
    (actuar::levgamma(limit = u1,  shape = risk1shape, scale = risk1scale) +
     actuar::levpareto(limit = u2, shape = risk2shape, scale = risk2scale) )
  partial1 <- 2*(part * (1-F1(u1))  - H11(u2))
  return(partial1)
}

f0.grad1(u1=start[1],u2=start[2],LM=start[3])
f0.grad2 <- function(u1,u2,LM){
  Fx1.u2 <- function(x1){F1(x1)-pCopula(c(F1(x1),F2(u2)), norm.cop12)}
  Fx1.u2.vec <- Vectorize(Fx1.u2)
  H22 <- function(u1){integrate(Fx1.u2.vec , lower = 0,
                                upper = u1)$value }
  part <- u1+u2 -
    (actuar::levgamma(limit = u1,  shape = risk1shape, scale = risk1scale) +
     actuar::levpareto(limit = u2, shape = risk2shape,scale = risk2scale) )
  partial2 <- 2*(part * (1-F2(u2))  - H22(u1) )
  return(partial2)
  }

f1in.grad1 <- function(u1,u2,LM){-1*(1-F1(u1))}
f1in.grad2 <- function(u1,u2,LM){-1*(1-F2(u2))}
LA.grad1 <- function(u1,u2,LM){f0.grad1(u1,u2) + LM*f1in.grad1(u1,u2) }
LA.grad2 <- function(u1,u2,LM){f0.grad2(u1,u2) + LM*f1in.grad2(u1,u2) }

RTCUpper <- actuar::mgamma(order=1,  shape = risk1shape, scale = risk1scale) +
            actuar::mpareto(order=1, shape = risk2shape, scale = risk2scale) - 1
#  Start at mean values
starter = c(actuar::mgamma(order=1, shape = risk1shape, scale = risk1scale),
            actuar::mpareto(order=1, shape = risk2shape, scale = risk2scale) )

OptMatMulti <- matrix(0,20,6)
RTCVec <- seq(10, RTCUpper, length.out=20)
for (kindex in 1:(length(RTCVec)-1)) {
 LA.grad3 <- function(u1,u2,LM) {f1in(u1,u2)-RTCVec[kindex]}
 model <- function(MParms) {
   c(F1=LA.grad1(MParms[1],MParms[2],MParms[3]),
     F2=LA.grad2(MParms[1],MParms[2],MParms[3]),
     F3=LA.grad3(MParms[1],MParms[2],MParms[3]))}
 starterA <- starter/kindex
 model(start)
 LA.grad1(start[1],start[2],start[3])

ss <- multiroot(f = model, start = c(starterA,10000))
  OptMatMulti[kindex,1:3] <- ss$root
  OptMatMulti[kindex,4] <- f0(ss$root[1],ss$root[2])
  OptMatMulti[kindex,5] <- f1in(ss$root[1],ss$root[2])
  OptMatMulti[kindex,6] <- RTCVec[kindex]
}
colnames(OptMatMulti) <- c("u1","u2","LME", "f0", "f1in", "RTCmax")
```

```
round(OptMatMulti,digits = 2)
```

15.5.11 Example 4.12 Linear Sharing Among Three Agents

Optimal Sharing Results and Table 4.7

```
# Example 4.12 Illustrative Code
library(CVXR)
num.row     <- 3
ones.vec    <- as.vector(rep(1,num.row))
ones.mat    <- matrix (rep(1,num.row^2),nrow = num.row)
X.sigma     <- matrix(c(10, -4 , -1 , -4 , 8 , 1, -1 ,  1 , 1), nrow = num.row)
X.sigma.inv <- solve(X.sigma)
mu = as.vector(c( 20, 2.5, 10))
Results.mat <- matrix(0,4,5)
Results.mat[1:3,1] <- diag(X.sigma)
Results.mat[4,1]   <- sum(Results.mat[1:3,1])

## Exchange Clearing
C.mat       <- Variable(3,3)
objective   <- quad_form(t(C.mat[1,]), X.sigma) +
               quad_form(t(C.mat[2,]), X.sigma) + quad_form(t(C.mat[3,]), X.sigma)
constraints <- list(t(C.mat) %*% ones.vec == ones.vec)
prob        <- Problem(Minimize(objective), constraints)
result      <- solve(prob)

C.opt.1     <- result$getValue(C.mat)
Summarize.Result <-function(Condnum,C.opt){
  Y.sigma <- C.opt %*% X.sigma %*% t(C.opt)
  Results.mat[1:3,Condnum+1] <- diag(Y.sigma)
  Results.mat[4,Condnum+1] <- sum(Results.mat[1:3,Condnum+1])
  return(Results.mat)
  }
Results.mat <- Summarize.Result(Condnum=1, C.opt=C.opt.1)

##  Exchange Clearing  and the No Profits Conditions
C.mat <- Variable(3,3)

objective    <- quad_form(t(C.mat[1,]), X.sigma) + quad_form(t(C.mat[2,]), X.sigma) +
                quad_form(t(C.mat[3,]), X.sigma)
constraints  <- list(t(C.mat) %*% ones.vec == ones.vec, C.mat %*% mu == mu)
prob         <- Problem(Minimize(objective), constraints)
result       <- solve(prob)

C.opt.2      <- result$getValue(C.mat)
Results.mat  <- Summarize.Result(Condnum=2,C.opt=C.opt.2)

## Add Limits
C.mat        <- Variable(3,3)
```

```
objective    <- quad_form(t(C.mat[1,]), X.sigma) + quad_form(t(C.mat[2,]), X.sigma) +
                quad_form(t(C.mat[3,]), X.sigma)
constraints  <- list(t(C.mat) %*% ones.vec == ones.vec,
                     C.mat %*% mu == mu, C.mat >= 0, C.mat <= 1)
prob         <- Problem(Minimize(objective), constraints)
result       <- solve(prob)
C.opt.3      <- result$getValue(C.mat)
Results.mat  <- Summarize.Result(Condnum=3,C.opt=C.opt.3)

## Add  Risk Improvement
C.mat        <- Variable(3,3)
objective    <- quad_form(t(C.mat[1,]), X.sigma) + quad_form(t(C.mat[2,]), X.sigma) +
                quad_form(t(C.mat[3,]), X.sigma)
constraints  <- list(t(C.mat) %*% ones.vec == ones.vec,
                     C.mat %*% mu == mu, C.mat >= 0, C.mat <= 1,
                     quad_form(t(C.mat[1,]), X.sigma) <= X.sigma[1,1],
                     quad_form(t(C.mat[2,]), X.sigma) <= X.sigma[2,2],
                     quad_form(t(C.mat[3,]), X.sigma) <= X.sigma[3,3])
prob         <- Problem(Minimize(objective), constraints)
result       <- solve(prob)
C.opt.4      <- result$getValue(C.mat)
Results.mat  <- Summarize.Result(Condnum=4,C.opt=C.opt.4)

# Check equality constraints
# t(C.opt.4) %*% ones.vec
# C.opt.4 %*% mu

# Check short selling
C.opt.3.mat  <- cbind(C.opt.1[,3],C.opt.2[,3],C.opt.3[,3],C.opt.4[,3])

rownames(Results.mat) <- c("Agent 1", "Agent 2", "Agent 3","Total")
colnames(Results.mat) <- c("Original",  "Clear Exchange",
                           "No Profits","Short Sell", "Risk Improvement")

TableGen1(TableData=Results.mat,
         TextTitle='Risk-Sharing Model Variances',
         Align='rrrrr', Digits=4, ColumnSpec=1:5,
         BorderRight=1 )
```

15.5.12 Example 4.14 Conditional Mean Risk-Sharing Among Three Agents

Determine Optimal Sharing and Plot Figure 4.8

```
# Example 4.14 Illustrative Code
num.row = 3
ones.vec <- as.vector(rep(1,num.row))
ones.mat <- matrix (rep(1,num.row^2),nrow = num.row)
X.sigma = matrix(c(10, -4 , -1 , -4 , 8 , 1, -1 ,  1 , 1), nrow = num.row)
```

```
X.var <- diag(X.sigma)
stdmat <- sqrt(diag(X.var,nrow=num.row, ncol=num.row))
X.corr <- solve(stdmat) %*% X.sigma%*% solve(stdmat)
corrparams <- X.corr[lower.tri(X.corr)]
mu = as.vector(c( 20, 2.5, 10))

library(copula)
library(actuar)
#  Set Simulation Parameters
nsim       <- 20000
set.seed(202020)
#  Normal, or Gaussian, Copula
norm.cop   <- copula::normalCopula(param=corrparams, dim = 3,'dispstr' ="un")
UCop       <- copula::rCopula(nsim, norm.cop)
#  Gamma Marginal Distributions
# scale= Var/Mean, shape = Mean/scale
# Pareto Marginal Distribution
# shape = 0.5*var/(var-mean^2), scale= mean*(shape-1)
risk1scale <- X.var[1]/mu[1]
risk1shape <- mu[1] / risk1scale
risk3scale <- X.var[3]/mu[3]
risk3shape <- mu[3] / risk3scale
risk2shape <- 2 * X.var[2] / (X.var[2]- mu[2]^2)
risk2scale <- mu[2]*(risk2shape - 1)

# Random variables
X1 <- qgamma(p=UCop[,1],shape = risk1shape, scale = risk1scale)
X2 <- actuar::qpareto(p=(UCop[,2]),shape = risk2shape, scale = risk2scale)
X3 <- qgamma(p=UCop[,3],shape = risk3shape, scale = risk3scale)
S  <- X1 + X2 + X3

LowQuan  <- quantile(S, p = 0.05)
HighQuan <- quantile(S, p = 0.95)
sVec     <- seq(from=LowQuan, to=HighQuan, length.out=91)
Results  <- matrix(0, nrow = 3, ncol = length(sVec))
bw       <- 1
for (i in 1:length(sVec) ){
  num <- sum((S>=sVec[i]-bw)*(S < sVec[i]+bw))
  Results[1,i] <- sum(X1*(S>=sVec[i]-bw)*(S < sVec[i]+bw)) / num
  Results[2,i] <- sum(X2*(S>=sVec[i]-bw)*(S < sVec[i]+bw)) / num
  Results[3,i] <- sum(X3*(S>=sVec[i]-bw)*(S < sVec[i]+bw)) / num
  }

densS <- density(S,from =LowQuan, to=HighQuan, n=91)
# Varh1 <- sum(densS$y*Results[1,]^2)/sum(densS$y) - mu[1]^2
# Varh2 <- sum(densS$y*Results[2,]^2)/sum(densS$y) - mu[2]^2
# Varh3 <- sum(densS$y*Results[3,]^2)/sum(densS$y) - mu[3]^2

Meanh1 <- sum(densS$y*Results[1,])/sum(densS$y)
Meanh2 <- sum(densS$y*Results[2,])/sum(densS$y)
Meanh3 <- sum(densS$y*Results[3,])/sum(densS$y)
```

```
VarhA1 <- sum(densS$y*(Results[1,]-Meanh1)^2)/sum(densS$y)
VarhA2 <- sum(densS$y*(Results[2,]-Meanh2)^2)/sum(densS$y)
VarhA3 <- sum(densS$y*(Results[3,]-Meanh3)^2)/sum(densS$y)

par(mfrow=c(1,3))
  plot(density(S), main= "", xlim = c(25, 50),
     xlab="Sum of Risks", yaxt = "n" )
    axis(side=2, las=2)
  plot(sVec, Results[1,] , ylim = c(0,25), col = FigGreen,
     ylab = "Share of Risks", xlab = "Sum of Risks",
     yaxt = "n", type = "l",lty = 3 , lwd = 2.0)
    axis(side=2, las=2)
    lines(sVec, Results[2,], col = FigRed, lty = 2)
    lines(sVec, Results[3,], col = "black",  lty = 1)
  plot(sVec, Results[1,]/sVec , ylim = c(0,0.7), col = FigGreen,
     ylab = "Proportion of Risks", xlab = "Sum of Risks",
     yaxt = "n", type = "l",lty = 3 , lwd = 2.0)
    axis(side=2, las=2)
    lines(sVec, Results[2,]/sVec, col = FigRed, lty = 2)
    lines(sVec, Results[3,]/sVec, col = "black",  lty = 1)
```

Table 4.9

```
# Example 4.14 Illustrative Code, Table 4.9
SumVar <- VarhA1 + VarhA2 + VarhA3
P2PVar <- c(VarhA1,VarhA2,VarhA3,SumVar)
Results1.mat <- cbind(Results.mat,P2PVar)
rownames(Results1.mat) <- c("Agent\u00A01", "Agent 2",
                            "Agent 3","Total")
colnames(Results1.mat) <- c("Original",  "Clear Exchange",
                            "No Profits","Short Sell",
                            "Risk Improve", "Conditional Mean")

TableGen1(TableData=Results1.mat,
         TextTitle='Risk Exchange Model Variances',
         Align='r', Digits=4, ColumnSpec=1:6,
         BorderRight=1, ColWidth = ColWidth6)
```

15.6 Chapter Five Code

15.6.1 Example 5.1. Lack of Convexity of Value at Risk for Excess of Loss

Set-Up

```
# Example 5.1 Set Up
risk1shape <- 2;           risk1scale <- 5000
risk2shape <- 5         ;  risk2scale <- 25000
rho  <-  -0.3
```

Evaluate Value at Risk

```
# Example 5.1 Illustrative Code
nsim <- 200000
set.seed(202020)
#  Normal, or Gaussian, Copula
norm.cop <- copula::normalCopula(param=-0.3, dim = 2,'dispstr' ="un")
UCop     <- copula::rCopula(nsim, norm.cop)
#  Marginal Distributions
X1 <- qgamma(p=UCop[,1],shape = risk1shape, scale = risk1scale)
X2 <- actuar::qpareto(p=(UCop[,2]),shape = risk2shape, scale = risk2scale)
cVec <- seq(0, 1, length.out=200)
u1.0 <- qgamma(p=0.70,shape = risk1shape, scale = risk1scale)
u1.1 <- qgamma(p=0.90,shape = risk1shape, scale = risk1scale)
u2.0 <- actuar::qpareto(p=0.70,shape = risk2shape, scale = risk2scale)
u2.1 <- actuar::qpareto(p=0.90,shape = risk2shape, scale = risk2scale)

u1Vec.a <- u1.0*(1-cVec) + u1.1*cVec
u1Vec.b <- u1.1*(1-cVec) + u1.0*cVec
u2Vec   <- u2.0*(1-cVec) + u2.1*cVec

#  Objective Function
f0 <- function(u1,u2){
      S <- pmin(X1,u1) +  pmin(X2,u2)
      return(quantile(S, probs=0.9))
      }
VarVec.a <- 0* cVec -> VarVec.b
for (kindex in 1:length(cVec)) {
  VarVec.a[kindex] <- f0(u1Vec.a[kindex],u2Vec[kindex])
  VarVec.b[kindex] <- f0(u1Vec.b[kindex],u2Vec[kindex])
}

save(cVec,VarVec.a, VarVec.b,
    file = "../ChapTablesData/Chap5/Example51.Rdata")
```

Figure 5.1

```
# Example 5.1 Illustrative Code, Figure 5.1
# save(cVec,VarVec.a, VarVec.b,
#     file = "../ChapTablesData/Chap5/Example51.Rdata")
load(file = "ChapTablesData/Chap5/Example51.Rdata")

par(mfrow=c(1,2))
  plot(cVec,VarVec.a, type = "l", ylab = "Value at Risk",
     xlab = "c", ylim = c(16500,23500), cex.lab=1.1)
    text(0.35, 17500,  expression(u[a1]==F[1]^-1~(0.7)), cex = 0.7)
    text(0.35, 17000,  expression(u[a2]==F[2]^-1~(0.7)), cex = 0.7)
    text(0.80, 23000,  expression(u[b1]==F[1]^-1~(0.9)), cex = 0.7)
    text(0.80, 22500,  expression(u[b2]==F[2]^-1~(0.9)), cex = 0.7)
    abline(a=VarVec.a[1],b=VarVec.a[length(cVec)]-VarVec.a[1],
       lty = "dashed", col =FigRed)
  plot(cVec,VarVec.b, type = "l", ylab = "Value at Risk",
     xlab = "c", ylim = c(16500,23500), cex.lab=1.1)
    text(0.20, 19500,  expression(u[a1]==F[1]^-1~(0.9)), cex = 0.7)
    text(0.20, 19000,  expression(u[a2]==F[2]^-1~(0.7)), cex = 0.7)
    text(0.80, 21500,  expression(u[b1]==F[1]^-1~(0.7)), cex = 0.7)
    text(0.80, 21000,  expression(u[b2]==F[2]^-1~(0.9)), cex = 0.7)
    abline(a=VarVec.b[1],b=VarVec.b[length(cVec)]-VarVec.b[1],
       lty = "dashed", col =FigRed)
```

15.6.2 Example 5.2. Three Measures of Uncertainty

Set-Up

```
# Example 5.2 Set Up
risk1shape <- 2;          risk1scale <- 5000
risk2shape <- 5         ;  risk2scale <- 5000
```

Evaluate Three Measures of Uncertainty

```
# Example 5.2 Illustrative Code
nsim <- 200000
# set.seed(2018)
Q          <- 18000
U1Vec      <- c(8000, 14000)
U2Vec      <- c(16000,  7000)
theta      <- seq(from = 0, to = 1, length.out = 15)
uparam     <- matrix(0,nrow=2,ncol=length(theta))
uparam[1,] <- U1Vec[1]*theta + U2Vec[1]*(1-theta)
uparam[2,] <- U1Vec[2]*theta + U2Vec[2]*(1-theta)
#uparam
alpha1     <- 0.75
```

```
rhoparam   <- c(-0.9,-.3,0,0.3,0.9)
matrix(0,length(theta),length(rhoparam)) ->
              yQ2 -> QuanS1 -> QuanS2
for (j in 1:length(rhoparam)) {
  set.seed(2018)
  copulaParam <- copula::iRho(normalCopula(param=1, dim = 2),
                              rho=rhoparam[j])
  nc    <- copula::normalCopula(param=copulaParam, dim = 2)
  U     <- copula::rCopula(nsim, nc)
  risk1 <- q1(U[,1])
  risk2 <- q1(U[,2])

  for (i in 1:length(theta)) {
    Retain <- pmin(risk1,uparam[1,i]) + pmin(risk2,uparam[2,i])
    df.2        <- ecdf(Retain)
    yQ2[i,j]    <- df.2(Q)
    quant       <- quantile(Retain, probs = c(alpha1))
    excess      <- pmax(Retain-quant,0)
    CTE         <- quant + mean(excess)/(1-alpha1)
    QuanS1[i,j] <- quant
    QuanS2[i,j] <- CTE
    }
}

save(theta,yQ2,QuanS1,QuanS2,Q,alpha1,
    file = "../ChapTablesData/Chap5/Example52.Rdata")
```

Figure 5.2

```
# Example 5.2 Illustrative Code, Figure 5.2
# save(theta,yQ2,QuanS1,QuanS2,
#     file = "../ChapTablesData/Chap5/Example52.Rdata")
load(file = "ChapTablesData/Chap5/Example52.Rdata")

par(mfrow = c(1,3))
  plot(theta,yQ2[,3], type = "l", ylab=paste("Dist Fct at ",Q),
       ylim=c(0.5,0.9), xlab = "c")
    lines(theta,yQ2[,4],  col=FigRed, lty=3)
        lines(theta,yQ2[,2],  col=FigBlue, lty=2)
lines(theta,yQ2[,5],  col=FigRed, lty=1)
    lines(theta,yQ2[,1],  col=FigBlue, lty=1)

  plot(theta,QuanS1[,3], type = "l", ylab=paste("VaR =",alpha1),
       xlab = "c", ylim = c(16000,24000))
    lines(theta,QuanS1[,4],  col=FigRed, lty=3)
    lines(theta,QuanS1[,2],  col=FigBlue, lty=2)
    lines(theta,QuanS1[,5],  col=FigRed, lty=1)
    lines(theta,QuanS1[,1],  col=FigBlue, lty=1)
```

```
  plot(theta,QuanS2[,3], type = "l", ylab=paste("ES =",alpha1),
       xlab = "c", ylim = c(16000,24000))
    lines(theta,QuanS2[,4],  col=FigRed, lty=3)
    lines(theta,QuanS2[,2],  col=FigBlue, lty=2)
    lines(theta,QuanS2[,5],  col=FigRed, lty=1)
    lines(theta,QuanS2[,1],  col=FigBlue, lty=1)
```

15.6.3 Example 5.3. Bivariate Excess of Loss Distribution

Preliminary Functions

```
# Chapter 5 Gamma Pareto Functions

Prob.fct <- function(u1,u2,rho,y){
  u1       <- u1*(u1>0)
  u2       <- u2*(u2>0)
  norm.cop <- copula::normalCopula(param=rho, dim = 2,'dispstr' ="un")
  ProbA <- 0 -> ProbB -> ProbC -> ProbD1 -> ProbD2
  if (y>u1) { ProbB <- F2(y-u1) - pCopula(c(F1(u1),F2(y-u1)), norm.cop) }
  if (y>u2) { ProbC <- F1(y-u2) - pCopula(c(F1(y-u2),F2(u2)), norm.cop) }
  ProbD1.x <- function(x){VineCopula::BiCopHfunc1(F1(x),F2(y-x),
                                                  family=1, par=rho)*f1(x)}
  ProbD1.x.vec <- Vectorize(ProbD1.x)
  ProbD1   <-  cubature::cubintegrate(ProbD1.x.vec,
                                      lower = max(0,y-u2), upper = min(y,u1),
                                      relTol =1e-4, absTol =1e-8,
                                      method = "hcubature")$integral
  if (y>u2) { ProbD2 <-  copula::pCopula(c(F1(y-u2),F2(u2)), norm.cop) }
  ProbTot <- ProbA + ProbB + ProbC + ProbD1 + ProbD2
  ProbTot <- ProbTot + (1-ProbTot)*(u1+u2<y)
  ProbTot <- ProbTot*(y>0)
  return(ProbTot)
  }

ExLossESGamPareto <- function(u1,u2,rho,alpha){
    Prob.alpha <- function(y){3e5*(Prob.fct(u1,u2,rho,y) - alpha)}
    VaR        <- uniroot(Prob.alpha, lower = 0, upper = 10e6)$root
    Surv.fct.y <- function(y){1-Prob.fct(u1,u2,rho,y)}
    Surv.fct.y.vec  <- Vectorize(Surv.fct.y)
    Sum.limited.Var <- integrate(Surv.fct.y.vec, lower = 0,
                                 upper = VaR)$value
    TotalRetained   <- actuar::levgamma(limit = u1,shape = risk1shape,
                                        scale = risk1scale) +
                       actuar::levpareto(limit = u2,shape = risk2shape,
                                         scale = risk2scale)
    ES        <- VaR + (TotalRetained- Sum.limited.Var)/(1-alpha)
    Output     <- list(Prob.fct(u1,u2,rho, y=4.5*RTCmax), VaR, ES)
    return(Output)
    }
```

Set-Up

```
# Example 5.3 Set Up
risk1shape <- 2;           risk1scale <- 2000
risk2shape <- 3         ;  risk2scale <- 2000
```

Excess of Loss - Distribution Function

```
# Example 5.3 Illustrative Code, Figure 5.3
#  No Constraints

#  Determine the Probability
u1<-5000;u2 <- 1500; rho<-0.5;
Probsumarg <- seq(max(u1,u2)-500, u1+u2+500, length.out=200)
Probsum    <- 0*Probsumarg
for (kindex in 1:length(Probsumarg)) {
    Probsum[kindex] <- Prob.fct(u1,u2,rho, Probsumarg[kindex]) }
plot(Probsumarg,Probsum, type="l", ylab="Probability", xlab="y" )
```

```
# Example 5.3 Illustrative Code
# Check Using Simulation
nsim      <- 1000000
rGaus     <- matrix(rnorm(2*nsim),nrow=nsim,ncol=2)
rho       <- 0.5
Sigma.11  <- matrix(c(1,rho,rho,1),nrow=2,ncol=2)
rGaus.new <- rGaus %*% chol(Sigma.11)
UCop1     <- pnorm(rGaus.new[,1])
UCop2     <- pnorm(rGaus.new[,2])
#  Marginal Distribution Random variables
X1 <- qgamma(p=UCop1,shape = risk1shape, scale = risk1scale)
X2 <- actuar::qpareto(p=UCop2,shape = risk2shape,
                      scale = risk2scale)
S <- pmin(X1,u1) +  pmin(X2,u2)
# Checks out well
# mean((S <= 6000))
# Prob.fct(u1,u2,rho, 6000)
```

Excess of Loss - Quantile

```
# Example 5.3 Illustrative Code
#  Determine the Quantile
u1 <- 5000; u2 <- 1500; rho <- 0.5; alpha <- 0.85;
Prob.alpha <- function(Quan){Prob.fct(u1,u2,rho,Quan) - alpha}
QuanAlpha  <- uniroot(Prob.alpha, lower = 0, upper = 10e6)$root
# Simulation
#QuanAlphasim <- quantile(S, probs = alpha, na.rm = TRUE)
# Constrained Optimization Approach
y.Vec <- function(par){par[1]}
```

```
Prob.y.Vec <- function(par){
  y=par[1]
  return(1e6*(Prob.fct(u1,u2,rho,y) - alpha))
  }
f0Prob.opt <-
  auglag(par=7000,                    # initial value
         fn=y.Vec,                    # objective function
         hin=Prob.y.Vec,              # inequality constraint
         control.outer=list(method="nlminb",trace=FALSE, itmax=20),
         control.optim=list(maxit=20))
#QuanAlpha;QuanAlphasim;f0Prob.opt$value
# Nice
```

```
# Example 5.3 Illustrative Code
# This code generates the VaR and ES over a grid of values of u1 and u2
# Done for corr = 0.5 and corr = -0.5
# Results are stored in and .RData file for later use....
TimeNeg   <- Sys.time()

alpha  <-  0.85
rho    <- 0.50

gridsize <- 50
u1Vec <-
  seq(qgamma(p=0.01,shape = risk1shape, scale = risk1scale),
      qgamma(p=0.99,shape = risk1shape, scale = risk1scale),
      length.out=gridsize)
u2Vec <-
  seq(actuar::qpareto(p=0.01,shape = risk2shape, scale = risk2scale),
      actuar::qpareto(p=0.99,shape = risk2shape, scale = risk2scale),
      length.out=gridsize)

RMGridPos <- matrix(0,gridsize**2,5)
count = 0
for (i in 1:gridsize)
  {for (j in 1:gridsize) {
    count = count + 1
  RMGridPos[count,1] <- u1Vec[i]
  RMGridPos[count,2] <- u2Vec[j]
  T1 <- ExLossESGamPareto(u1=u1Vec[i],u2=u2Vec[j],rho,alpha)
  RMGridPos[count,3] <- T1[[2]]
  RMGridPos[count,4] <- T1[[3]]
  RMGridPos[count,5] <- RC1(u1Vec[i]) + RC2(u2Vec[j])
  }}
colnames(RMGridPos) <- c("u1", "u2", "VaR", "ES", "RTC")
RMGridPos <- data.frame(RMGridPos)
save(RMGridPos,
     file = "../ChapTablesData/Chap5/Example521PosCorr.RData")

#Rerun with negative correlation
```

```
TimeNeg   <- Sys.time()
rho       <- -0.5;
RMGridNeg <- matrix(0,gridsize**2,5)
count     <-  0
for (i in 1:gridsize)
  {for (j in 1:gridsize) {
    count  <-  count + 1
  RMGridNeg[count,1] <- u1Vec[i]
  RMGridNeg[count,2] <- u2Vec[j]
  T1 <- ExLossESGamPareto(u1=u1Vec[i],u2=u2Vec[j],rho,alpha)
  RMGridNeg[count,3] <- T1[[2]]
  RMGridNeg[count,4] <- T1[[3]]
  RMGridNeg[count,5] <- RC1(u1Vec[i]) + RC2(u2Vec[j])
  }}

colnames(RMGridNeg) <- c("u1", "u2", "VaR", "ES", "RTC")
RMGridNeg <- data.frame(RMGridNeg)
save(RMGridNeg,
     file = "../ChapTablesData/Chap5/Example521NegCorr.RData")
Sys.time() - TimeNeg
#Time difference of 3.11391334 hours
```

```
# Example 5.3 Illustrative Code, Figure 5.4
# save(RMGridPos,
#      file = "../ChapTablesData/Chap5/Example521PosCorr.RData")
load( file = "ChapTablesData/Chap5/Example521PosCorr.RData")

angle <- 22+c(0, 90, 180,270)
#angle <- c( 160, 240, 110, 20)
Angle <- shingle(rep(angle, rep(length(RMGridPos[,3]), 4)),angle)

wireframe(rep(RMGridPos[,3], 4) ~
          rep(RMGridPos[,2], 4) *rep(RMGridPos[,1], 4) | Angle,
          groups = Angle,
  panel = function(x, y, subscripts, z, groups,...){
    w <- groups[subscripts][1]
    panel.wireframe(x, y, subscripts, z,
    screen = list(z = w, x = -60, y = 0), ...)},
  strip = FALSE,
  par.settings = list(par.zlab.text=list(cex = 0.6)),
  skip = c(F, F, F, F),
  layout = c(2,2),#shade=TRUE,
  distance = 0.1, col = FigLBlue,
  xlab = expression(u[2]), ylab = expression(u[1]),
  zlab = "VaR")
```

15.6.4 Example 5.4. Partial Derivatives of Bivariate Excess of Loss Distribution Function

```
# Example 5.4 Illustrative Code
#  No Constraints
PDeriv.fct <- function(u1,u2,rho,y){
  u1        <- u1*(u1>0)
  u2        <- u2*(u2>0)
  norm.cop  <- copula::normalCopula(param=rho, dim = 2,'dispstr' ="un")
  PDeriv.u1 <- -((u1<y)*f2(y-u1)*(1 -
    VineCopula::BiCopHfunc2(F1(u1),F2(y-u1), family=1, par=rho)) )*(u1+u2>y)
  PDeriv.u2 <- -((u2<y)*f1(y-u2)*(1 -
    VineCopula::BiCopHfunc2(F2(u2),F1(y-u2), family=1, par=rho)) )*(u1+u2>y)
  return(c(PDeriv.u1,PDeriv.u2))
  }

#  Plot the Partial Derivatives
u1 <- 5000;u2 <- 1500; rho <- 0.5;
PDeriv.arg <- seq(max(u1,u2)-500, u1+u2+500, length.out=20)
PDeriv.u1  <- 0*PDeriv.arg -> PDeriv.u2
for (kindex in 1:length(PDeriv.arg)) {
  PDeriv.u1[kindex] <- PDeriv.fct(u1,u2,rho, PDeriv.arg[kindex])[1]
  PDeriv.u2[kindex] <- PDeriv.fct(u1,u2,rho, PDeriv.arg[kindex])[2]
  }
plot(PDeriv.arg,PDeriv.u1, type="l",
     ylab="Partial Derivatives", xlab="y", cex.lab=1.5 )
  lines(PDeriv.arg, PDeriv.u2, col = FigRed , lty =2)
  text(6500,-0.00010, expression(u[1]), cex=1.5)
  text(5500,-0.00001, expression(u[2]), cex=1.5)

#  Determine the Partial Derivative Using Numerical Approximation
Prob.fct.Vec <- function(par,rho){Prob.fct(par[1],par[2],par[3],par[4])}
#  Agreement in these two approaches
#numDeriv::grad(f=Prob.fct.Test.Vec, x = c(5000,1500))
# numDeriv::grad(f=Prob.fct.Vec, x = c(5000,1500,0.5,6000))
# PDeriv.fct(5000,1500,0.5,6000)
```

15.6.5 Example 5.5. Second-Order Partial Derivatives of Bivariate Excess of Loss Distribution Function

```
# Example 5.5 Illustrative Code
f2prime <- function(y){-f2(y)*(risk2shape+1)/(y+risk2scale) }
f1prime <- function(y){
           ( (risk1shape-1)/y - 1/risk1scale )*f1(y)
           }
C22    <- function(v1,v2,rho){
          z1  <- qnorm(v1)
          z2  <- qnorm(v2)
          arg <- (z1 - rho*z2)/sqrt(1 - rho*rho)
```

```
          C22.v1v2 <- - rho/(sqrt(1 - rho*rho)*dnorm(z2))*dnorm(arg)
          return(C22.v1v2)
          }
PSecDeriv.fct <- function(u1,u2,rho,y){
    u1       <- u1*(u1>0)
    u2       <- u2*(u2>0)
    norm.cop <- copula::normalCopula(param=rho, dim = 2,'dispstr' ="un")
    PSecDeriv.u1 <- f2prime(y-u1)*(1 -
       VineCopula::BiCopHfunc2(F1(u1),F2(y-u1), family=1, par=rho) ) +
       f2(y-u1)*dCopula(c(F1(u1),F2(y-u1)), norm.cop)*f1(u1) -
       f2(y-u1)*C22(F1(u1),F2(y-u1),rho)*f2(y-u1)
    PSecDeriv.u1 <- PSecDeriv.u1*(u1+u2>y)
    if (y<u1) PSecDeriv.u1 <- 0
    PSecDeriv.u2 <- f1prime(y-u2)*(1 -
          VineCopula::BiCopHfunc2(F2(u2),F1(y-u2), family=1, par=rho)) +
       f1(y-u2)*dCopula(c(F2(u2),F1(y-u2)), norm.cop)*f2(u2) -
       f1(y-u2)*C22(F2(u2),F1(y-u2),rho)*f1(y-u2)
    PSecDeriv.u2 <- PSecDeriv.u2*(u1+u2>y)
    if (y<u2) PSecDeriv.u2 <- 0
    return(c(PSecDeriv.u1,PSecDeriv.u2))
  }

#  Plot the Partial Derivatives
#u1=5000;u2 = 1500; y=4000;rho=0.5; PSecDeriv.fct(u1,u2, y)
PSecDeriv.arg <- seq(max(u1,u2)-500, u1+u2+500, length.out=20)
PSecDeriv.u1 <- 0*PSecDeriv.arg -> PSecDeriv.u2
for (kindex in 1:length(PSecDeriv.arg)) {
PSecDeriv.u1[kindex] <-
  PSecDeriv.fct(u1,u2,rho, PSecDeriv.arg[kindex])[1]
PSecDeriv.u2[kindex] <-
  PSecDeriv.fct(u1,u2,rho, PSecDeriv.arg[kindex])[2]
}
plot(PSecDeriv.arg,PSecDeriv.u1, type="l",
     ylab="Partial Second Derivatives", xlab="y" )
lines(PSecDeriv.arg, PSecDeriv.u2, col = FigRed , lty =2)
text(5200, 0.0000010, expression(u[1]))
text(6000, 0.0000001, expression(u[2]))

#  Determine the Partial Second Derivatives Using Numerical Approximations
#  Agreement in these two approaches
# testpoint <- c(5000,1500,0.5,6000)
# PDeriv.fct.Vec.u1 <- function(par){PDeriv.fct(par[1],par[2],0.5,6000)[1]}
# PDeriv.fct.Vec.u2 <- function(par){PDeriv.fct(par[1],par[2],0.5,6000)[2]}
# 1e7*numDeriv::hessian(f=Prob.fct.Test.Vec, x = c(5000,1500))
# 1e7*numDeriv::hessian(f=Prob.fct.Vec, x = testpoint )
#  #  Note that cross-partial derivatives are incorrect numerically.
#  #  This is due to the boundary induced by u1 + u2 > y. Safe to ignore.
# 1e7*numDeriv::grad(f=PDeriv.fct.Vec.u1, x = testpoint)[1]
# 1e7*numDeriv::grad(f=PDeriv.fct.Vec.u2, x = testpoint)[2]
# 1e7*PSecDeriv.fct(5000,1500,0.5,6000)
```

```
# To check using simulation, need to increase the number of simulations,
# testpointA <- c(5000,1500,6000)
# Prob.Sim.Vec <- function(par){
#    S <- pmin(X1,par[1]) +  pmin(X2,par[2])
#    mean((S <= par[3]))}
# Prob.Sim.Vec(testpointA)
# Prob.fct.Vec(testpoint)
# 1e7*numDeriv::jacobian(f=Prob.Sim.Vec, x = testpointA)
# 1e7*numDeriv::jacobian(f=Prob.fct.Vec, x = testpoint)
# 1e7*numDeriv::hessian(f=Prob.Sim.Vec, x = testpointA)
# 1e7*numDeriv::hessian(f=Prob.fct.Vec, x = testpoint)
```

15.6.6 Section 5.3.1. Quantile Derivatives

```
# Section 5.3.1 Illustrative Code, on Quantile Derivatives...
# f2prime <- function(y){ -f2(y)*(risk2shape+1)/(y+risk2scale) }
f1prime <- function(y){ ((risk1shape-1)/y - 1/risk1scale)*f1(y) }

CrossPDeriv.fct <- function(u1,u2,rho,y){
  u1       <- u1*(u1>0)
  u2       <- u2*(u2>0)
  norm.cop <- copula::normalCopula(param=rho, dim = 2,'dispstr' ="un")
  PSecDeriv.u1.y <-
    f2prime(y-u1)*(1 -
      VineCopula::BiCopHfunc2(F1(u1),F2(y-u1), family=1, par=rho)) +
    - f2(y-u1)**2*C22(F1(u1),F2(y-u1),rho)
  PSecDeriv.u1.y <- -PSecDeriv.u1.y*(u1+u2>y)
  if (y<u1) PSecDeriv.u1.y <- 0
  PSecDeriv.u2.y <-
    f1prime(y-u2)*(1 -
      VineCopula::BiCopHfunc2(F2(u2),F1(y-u2), family=1, par=rho)) +
    f1(y-u2)*dCopula(c(F2(u2),F1(y-u2)), norm.cop)*f2(u2) -
    f1(y-u2)**2*C22(F2(u2),F1(y-u2),rho)
  PSecDeriv.u2.y <- -PSecDeriv.u2.y*(u1+u2>y)
  if (y<u2) PSecDeriv.u2.y <- 0
  return(c(PSecDeriv.u1.y,PSecDeriv.u2.y))
  }

ExLossCTE <- function(u1,u2,rho,alpha){
# See Section 5.6 for Theory underpinning ES derivatives
    Prob.alpha    <- function(y){Prob.fct(u1,u2,rho,y) - alpha}
    VaR           <- uniroot(Prob.alpha, lower = 0, upper = 10e6)$root
    Surv.fct.y    <- function(y){1-Prob.fct(u1,u2,rho,y)}
    Surv.fct.y.vec <- Vectorize(Surv.fct.y)
  Sum.limited.Var <- integrate(Surv.fct.y.vec, lower = 0, upper = VaR)$value
  TotalRetained   <- actuar::levgamma(limit = u1,shape = risk1shape,
                                      scale = risk1scale) +
                     actuar::levpareto(limit = u2,shape = risk2shape,
                                       scale = risk2scale)
```

```
  CTE <- VaR + (TotalRetained- Sum.limited.Var)/(1-alpha)

    #  Determine the density at the Quantile
  densVaR <- numDeriv::grad(f=Prob.alpha, x = VaR)
  #  Later, do this analytically
  #  Determine the partial derivatives at the quantile
  Pderivs <- PDeriv.fct(u1,u2,rho,VaR)
  #  Determine the quantile sensitivities (Eqn 5.6)
  VaRSens    <- -Pderivs/densVaR
  # often, close to zero
  ContinuityAdjust <- 1 - (1-Prob.fct(u1,u2,rho,VaR))/(1-alpha)
  norm.cop   <- copula::normalCopula(param=rho, dim = 2,'dispstr' ="un")
  CTESens    <- 0 *VaRSens
  CTESens[1] <- 1-F1(u1) - (F2(VaR-u1) -
                            copula::pCopula(c(F1(u1),F2(VaR-u1)), norm.cop))
  CTESens[1] <- CTESens[1]/(1-alpha) + ContinuityAdjust*VaRSens[1]
  CTESens[2] <- 1-F2(u2) -
           (F1(VaR-u2) - copula::pCopula(c(F2(u2),F1(VaR-u2)), norm.cop))
  CTESens[2] <- CTESens[2]/(1-alpha) + ContinuityAdjust*VaRSens[2]
  #CTESens
  FythetaCross <- CrossPDeriv.fct(u1,u2,rho,VaR)
  temp       <- FythetaCross %*% t(VaRSens)
  HessVaR    <- (1/densVaR)* (diag(PSecDeriv.fct(u1,u2,rho,VaR)) +
       numDeriv::hessian(f=Prob.fct.Vec, x = c(u1,u2,rho,VaR))[3,3] *
       FythetaCross %*% t(VaRSens) +  temp + t(temp) )
  HessCTE    <- ContinuityAdjust*HessVaR
  HessCTE[1,1] <- HessCTE[1,1] + (-f1(u1) - f2(VaR-u1)*(1 -
       VineCopula::BiCopHfunc2(F1(u1),F2(VaR-u1), family=1, par=rho)))/(1-alpha)
  HessCTE[2,2] <- HessCTE[2,2] + (-f2(u2) - f1(VaR-u2)*(1 -
       VineCopula::BiCopHfunc2(F2(u2),F1(VaR-u2), family=1, par=rho)))/(1-alpha)
  Output <- list(VaR,CTE, VaRSens, CTESens, HessVaR, HessCTE)
  return(Output)
  }

#  Initial parameters
u1 <- 5000;u2 <- 1500; rho <- 0.5;alpha <- 0.85

rhoVec      <- seq(-0.2, 0.8, length.out=11)
VaR.Vec1    <- 0* rhoVec -> CTE.Vec1
VaR.Sens1   <- matrix(0,ncol =2, nrow = length(rhoVec)) -> CTE.Sens1
detHessVaR1 <- 0* rhoVec -> detHessCTE1

for (kindex in 1:length(rhoVec)) {
temp               <- ExLossCTE(u1,u2, rho=rhoVec[kindex],alpha)
VaR.Vec1[kindex]    <- temp[[1]]
CTE.Vec1[kindex]    <- temp[[2]]
VaR.Sens1[kindex,]  <- temp[[3]]
CTE.Sens1[kindex,]  <- temp[[4]]
detHessVaR1[kindex] <- det(temp[[5]])
detHessCTE1[kindex] <- det(temp[[6]])
}
```

```
alphaVec    <- seq(0.55, 0.95, length.out=10)
VaR.Vec2    <- 0* alphaVec -> CTE.Vec2
VaR.Sens2   <- matrix(0,ncol =2, nrow = length(alphaVec)) -> CTE.Sens2
detHessVaR2 <- 0* alphaVec -> detHessCTE2

for (kindex in 1:length(alphaVec)) {
temp                <- ExLossCTE(u1,u2, rho,alpha = alphaVec[kindex])
VaR.Vec2[kindex]    <- temp[[1]]
CTE.Vec2[kindex]    <- temp[[2]]
VaR.Sens2[kindex,]  <- temp[[3]]
CTE.Sens2[kindex,]  <- temp[[4]]
detHessVaR2[kindex] <- det(temp[[5]])
detHessCTE2[kindex] <- det(temp[[6]])
}

save(alphaVec,VaR.Vec2,VaR.Sens2,rhoVec,VaR.Vec1,VaR.Sens1,alpha,
     CTE.Vec1,CTE.Sens1,CTE.Vec2,CTE.Sens2,
     file = "../ChapTablesData/Chap5/Sec531QuantileDeriv.RData")
```

15.6.7 Example 5.6. Bivariate Excess of Loss VaR Sensitivities

```
# Example 5.6, Figure 5.7
#save(alphaVec,VaR.Vec2,VaR.Sens2,rhoVec,VaR.Vec1,VaR.Sens1,alpha,
#     CTE.Vec1,CTE.Sens1,CTE.Vec2,CTE.Sens2,
#     file = "../ChapTablesData/Chap5/Sec531QuantileDeriv.RData")
load(file = "ChapTablesData/Chap5/Sec531QuantileDeriv.RData")

par(mfrow=c(2,2))
  plot(alphaVec,VaR.Vec2, xlab = expression('confidence level' ~ alpha),
       ylab = "VaR", type="l")
  plot(alphaVec,VaR.Sens2[,1], ylim = c(0,1.7), type="l",
       xlab = expression('confidence level' ~ alpha), ylab = "VaR Sensitivity")
    lines(alphaVec,VaR.Sens2[,2], col = FigRed , lty =2)
    text(0.85, 1.3, expression(u[1]))
    text(0.7, 0.4, expression(u[2]))
  plot(rhoVec,VaR.Vec1, xlab = expression('dependence' ~ rho),
       ylab = "VaR", type="l")
  plot(rhoVec,VaR.Sens1[,1], ylim = c(0,1.5), type="l",
       xlab = expression('dependence' ~ rho), ylab = "VaR Sensitivity")
    lines(rhoVec,VaR.Sens1[,2], col = FigRed , lty =2)
    text(0, 1.3, expression(u[1]))
    text(0, 0.3, expression(u[2]))
```

15.6.8 Example 5.7. Bivariate Excess of Loss - ES Sensitivities

```
# Example 5.7 Illustrative Code
load(file = "ChapTablesData/Chap5/Sec531QuantileDeriv.RData")
par(mfrow=c(2,2))
  plot(alphaVec,CTE.Vec2, xlab = expression('confidence level' ~ alpha),
       ylab = "ES", type="l")
  plot(alphaVec,CTE.Sens2[,1], ylim = c(0,2), type="l",
       xlab = expression('confidence level' ~ alpha),
       ylab = "ES Sensitivity")
    lines(alphaVec,CTE.Sens2[,2], col = FigRed , lty =2)
    text(0.85, 1.3, expression(u[1]))
    text(0.7, 0.2, expression(u[2]))

plot(rhoVec,CTE.Vec1, xlab = expression('dependence' ~ rho),
     ylab = "ES", type="l")
plot(rhoVec,CTE.Sens1[,1], ylim = c(0,2), type="l",
     xlab = expression('dependence' ~ rho),
     ylab = "ES Sensitivity")
lines(rhoVec,CTE.Sens1[,2], col = FigRed , lty =2)
text(0, 1.3, expression(u[1]))
text(0, 0.3, expression(u[2]))
```

15.6.9 Example 5.8. Visualizing Excess of Loss Constrained Optimization - ES

```
# Example 5.8 Illustrative Code, Figure 5.9
# save(RMGridPos,
#      file = "../ChapTablesData/Chap5/Example521PosCorr.RData")
load( file = "ChapTablesData/Chap5/Example521PosCorr.RData")

ggRTC <- ggplot2::ggplot(RMGridPos, aes(x = u1, y = u2)) +
  geom_raster(aes(fill = RTC)) +
  geom_contour(aes(z = RTC), colour = FigBlue,
               size = 0.1, alpha = 0.5, bins = 6) +
  geom_text_contour(aes(z = RTC),  colour = FigBlue, size = 3) +
  labs(x = expression(u[1]),  y = expression(u[2]), fill = "RTC") +
  theme(legend.title = element_text(size = 10, face = "bold"),
  legend.position = "top", panel.background = element_blank(),
  axis.text = element_text(colour = "black", size = 10),
  axis.title = element_text(size = 12, face = "italic"),
  legend.text = element_text(size = 7),
  legend.key = element_blank() ) +
  scale_fill_continuous(low = "white", high = "grey90") +
  scale_y_continuous(expand = c(0,0) ) +
  scale_x_continuous(expand = c(0,0) )
ggPosCTE2 = ggRTC +
  geom_contour(aes(z = ES), colour = FigRed, linetype = 18) +
  geom_text_contour(aes(z = ES),  colour = FigRed , size = 3)
gridExtra::grid.arrange(ggRTC,ggPosCTE2,nrow=1)
```

15.6.10 Example 5.9. Visualizing Excess of Loss Constrained Optimization - VaR

```
# Example 5.9 Illustrative Code, Figure 5.10
ggPosVar2 = ggRTC +
  geom_contour(aes(z = VaR), colour = FigRed, linetype = 18) +
  geom_text_contour(aes(z = VaR),  colour = FigRed , size = 3)
ggPosVar2
```

15.6.11 Section 5.4.2. Distribution Function Contour Plot

```
risk1shape <- 2;           risk1scale <- 2000
risk2shape <- 3         ;  risk2scale <- 2000
```

```
# Section 5.4.2, Figure 5.11

load( file = "ChapTablesData/Chap5/Sec542RMGridNoDFCorrB.RData")
lengthGrid = length(RMGridNoDF[,"u1"])
midpoint = RMGridNoDF[round(lengthGrid/2),]

RMGridNoDFmidu1A <- subset(RMGridNoDF, u1==as.numeric(midpoint["u1"]))
RMGridNoDFmidu1B <- subset(RMGridNoDF, u1==5375)
RMGridNoDFmidu2A <- subset(RMGridNoDF, u2==35000)
RMGridNoDFmidu2B <- subset(RMGridNoDF, u2==50000)
# str(RMGridNoDF)
# table(RMGridNoDF$u1)
# table(RMGridNoDF$u2)
breaksProb <- seq(0.80, 0.96, by = 0.04)
breaksRTC <- seq(200, 1200, by = 6)
ggNoDFConu1A = ggplot(RMGridNoDFmidu1A, aes(x = u2, y = x)) +
  geom_raster(aes(fill = x)) +
  geom_contour(aes(z = Prob), colour = FigGreen, size = 0.2, alpha = 0.5) +
  geom_text_contour(aes(z = Prob),  colour = FigGreen ,breaks = breaksProb) +
  geom_contour(aes(z = RTC), colour = FigBlue, linetype = 18) +
  geom_text_contour(aes(z = RTC),  colour = FigBlue ,breaks = breaksRTC) +
  theme(legend.position = "none") +
  scale_fill_continuous(low = "white", high = "grey90") +
  scale_y_continuous(expand = c(0,0)) +
  scale_x_continuous(expand = c(0,0)) +
  labs(x = expression(u[2]), title =expression(paste('Constant ', u[1], '=4500')))  #

ggNoDFConu1B = ggplot(RMGridNoDFmidu1B, aes(x = u2, y = x)) +
  geom_raster(aes(fill = x)) +
  geom_contour(aes(z = Prob), colour = FigGreen, size = 0.2, alpha = 0.5) +
  geom_text_contour(aes(z = Prob),  colour = FigGreen ,breaks = breaksProb) +
  geom_contour(aes(z = RTC), colour = FigBlue, linetype = 18) +
  geom_text_contour(aes(z = RTC),  colour = FigBlue ,breaks = breaksRTC) +
  theme(legend.position = "none") +
```

```
  scale_fill_continuous(low = "white", high = "grey90") +
  scale_y_continuous(expand = c(0,0)) +
  scale_x_continuous(expand = c(0,0)) +
  labs(x = expression(u[2]), title =expression(paste('Constant ', u[1], '=5375')))

ggNoDFConu2A = ggplot(RMGridNoDFmidu2A, aes(x = u1, y = x)) +
  geom_raster(aes(fill = x)) +
  geom_contour(aes(z = Prob), colour = FigGreen, size = 0.2, alpha = 0.5) +
  geom_text_contour(aes(z = Prob),  colour = FigGreen ,breaks = breaksProb) +
  geom_contour(aes(z = RTC), colour = FigBlue,linetype = 18) +
  geom_text_contour(aes(z = RTC),  colour = FigBlue ) +
  theme(legend.position = "none") +
  scale_fill_continuous(low = "white", high = "grey90") +
  scale_y_continuous(expand = c(0,0)) +
  scale_x_continuous(expand = c(0,0)) +
 labs(x = expression(u[1]), title =expression(paste('Constant ', u[2], '=35000')))

ggNoDFConu2B = ggplot(RMGridNoDFmidu2B, aes(x = u1, y = x)) +
  geom_raster(aes(fill = x)) +
  geom_contour(aes(z = Prob), colour = FigGreen, size = 0.2, alpha = 0.5) +
  geom_text_contour(aes(z = Prob),  colour = FigGreen ,breaks = breaksProb) +
  geom_contour(aes(z = RTC), colour = FigBlue, linetype = 18) +
  geom_text_contour(aes(z = RTC),  colour = FigBlue ) +
  theme(legend.position = "none") +
  scale_fill_continuous(low = "white", high = "grey90") +
  scale_y_continuous(expand = c(0,0)) +
  scale_x_continuous(expand = c(0,0)) +
  labs(x = expression(u[1]), title =expression(paste('Constant ', u[2], '=50000')))

gridExtra::grid.arrange(ggNoDFConu2A,ggNoDFConu2B,
             ggNoDFConu1A,ggNoDFConu1B, nrow=2)
```

15.6.12 Example 5.10. Parameter Limits for Gamma and Pareto Distributions

```
# Example 5.10 Illustrative Code, Part A
RTCmax <- 0.30*(ERisk1fun() + ERisk2fun())
alpha <- 0.85
# Lower/Upper Bounds for Active Trade-off Function AT

Ubdd.fct <- function(j, RTCmax){
  RCj          <- function(u){RC1(u)*(j==1) + RC2(u)*(j==2)}
  Excess       <- RTCmax - (RTC(0,0) - RCj(0))
  Ubdd.AT.fctj <- function(u){Excess - RCj(u)}
  Upperbdd     <- 1.0e8
  if (Excess > 0) {Upperbdd <-
          uniroot(Ubdd.AT.fctj, lower = 0, upper =5e4)$root}
  return(Upperbdd)
```

```
  }
Lbdd.fct <- function(j, RTCmax){
  RCj          <- function(u){RC1(u)*(j==1) + RC2(u)*(j==2)}
  Lbdd.AT.fctj <- function(u){RTCmax - RCj(u)}
  Lowerbdd     <- 0
  if (RTCmax < RCj(0) ) {Lowerbdd <-
          uniroot(Lbdd.AT.fctj, lower = 0, upper = 5e4)$root}
  return(Lowerbdd)
  }
ATj.fct  <- function(j,u, RTCmax){
  RCj    <- function(u){RC1(u)*(j==1) + RC2(u)*(j==2)}
  RCnej  <- function(u){RC1(u)*(j != 1) + RC2(u)*(j != 2)}
  RCnejx <- function(x){RTCmax - RCj(u) - RCnej(x)}
  ustar  <- 1.0e8
  if (RTCmax - RCj(u) >= RCnej(0) ) {ustar <- 0}
  if ( (RTCmax - RCj(u) > 0)*(RTCmax - RCj(u) < RCnej(0)) ) {
           ustar <- uniroot(RCnejx, lower =  0, upper =1e9)$root}
  return(ustar)
  }
AT1.fct <- function(u1,RTCmax){
  RC2x   <- function(x){RTCmax - RC1(u1) - RC2(x)}
  u2star <- 1.0e8 #1.2e4
  if ( RTCmax - RC1(u1) >= RC2(0) ) {u2star <- 0}
  if ( (RTCmax - RC1(u1) > 0)*(RTCmax - RC1(u1) < RC2(0)) ) {
          u2star <- uniroot(RC2x, lower =  0, upper =1e9)$root}
  return(u2star)
}
```

```
# Example 5.10 Illustrative Code, Part B
OutMat.U    <- matrix(0,nrow = 8, ncol = 2)
u1.u1Lower  <- Lbdd.fct(1, RTCmax)+0.001-> OutMat.U[1,1]
u2.u1Lower  <- ATj.fct(1,u1.u1Lower, RTCmax) -> OutMat.U[2,1]
tempLower   <- ExLossESGamPareto(u1.u1Lower,u2.u1Lower,rho,alpha)
VaR.u1Lower <- tempLower[[2]]          -> OutMat.U[3,1]
CTE.u1Lower <- tempLower[[3]]          -> OutMat.U[4,1]
u1.u1Upper  <- Ubdd.fct(1, RTCmax)     -> OutMat.U[5,1]
u2.u1Upper  <- ATj.fct(1,u1.u1Upper, RTCmax) -> OutMat.U[6,1]
tempUpper   <- ExLossESGamPareto(u1.u1Upper,u2.u1Upper,rho,alpha)
VaR.u1Upper <- tempUpper[[2]]          -> OutMat.U[7,1]
CTE.u1Upper <- tempUpper[[3]]          -> OutMat.U[8,1]

u2.u2Lower  <- Lbdd.fct(2, RTCmax)+0.00-> OutMat.U[2,2]
u1.u2Lower  <- ATj.fct(2,u2.u2Lower, RTCmax) -> OutMat.U[1,2]
tempLower   <- ExLossESGamPareto(u1.u2Lower,u2.u2Lower,rho,alpha)
VaR.u2Lower <- tempLower[[2]]           -> OutMat.U[3,2]
CTE.u2Lower <- tempLower[[3]]           -> OutMat.U[4,2]
u2.u2Upper  <- Ubdd.fct(2, RTCmax)      -> OutMat.U[6,2]
u1.u2Upper  <- ATj.fct(2,u2.u2Upper, RTCmax) -> OutMat.U[5,2]
tempUpper   <- ExLossESGamPareto(u1.u2Upper,u2.u2Upper,rho,alpha)
VaR.u2Upper <- tempUpper[[2]]           -> OutMat.U[7,2]
```

```
CTE.u2Upper <- tempUpper[[3]]          -> OutMat.U[8,2]

save(OutMat.U,
     file = "../ChapTablesData/Chap5/Ex551ActiveConstraints.RData")
```

15.6.13 Example 5.11. Fair Transfer Costs, Uniform Distributions

```
# Example 5.11 Illustrative Code, Figure 5.12
ATUnif <- function(u,RTCmax){
  sqroot = 2*RTCmax-(1-u)**2
  Gu <- 1-sqrt(sqroot*(sqroot>0))
  Gu <- Gu*(Gu>0)
  Gu <- pmin(1,Gu)
  return(Gu)
  }
uparambase<- seq(0, 1, length.out = 110)
u.0.4 <- ATUnif(uparambase, RTCmax = 0.4)
u.0.5 <- ATUnif(uparambase, RTCmax = 0.5)
u.0.2 <- ATUnif(uparambase, RTCmax = 0.2)
u.0.8 <- ATUnif(uparambase, RTCmax = 0.8)
RTCmax = 0.8
plot(uparambase,u.0.8, type = "l", lty = 1,
     xlab = "u", ylab = "AT(u)", ylim=c(0,1) )
  lines(uparambase,u.0.5, lty = 2, col = FigRed)
  lines(uparambase,u.0.4, lty = 3, col = FigBlue)
  lines(uparambase,u.0.2, lty = 4, col = FigGreen)
```

15.6.14 Example 5.13 Visualizing the Objective Function, Uniform Distributions

```
# Example 5.13 Illustrative Code
RCUnif  <- function(u){ 0.5 - (1-u)**2/2 }
RTCstar <- function(u1,u2,RTCmax){ RCUnif(u1) + RCUnif(u2) - RTCmax }

ATUnif <- function(u,RTCmax){
  sqroot <- 2*RTCmax-(1-u)**2
  Gu <- 1-sqrt(sqroot*(sqroot>0))
  Gu <- Gu*(Gu>0)
  Gu <- pmin(1,Gu)
  return(Gu)
  }
ExLossCTEUnif <- function(u1,u2,rho,alpha){
    Prob.alpha      <- function(y){Prob.fct(u1,u2,rho,y) - alpha}
    VaR             <- uniroot(Prob.alpha, lower = 0, upper = 10e6)$root
    Surv.fct.y      <- function(y){1-Prob.fct(u1,u2,rho,y)}
    Surv.fct.y.vec  <- Vectorize(Surv.fct.y)
```

```
    Sum.limited.Var <-  integrate(Surv.fct.y.vec, lower = 0, upper = VaR)$value
    TotalRetained   <- u1-u1**2/2 + u2-u2**2/2
    CTE             <- VaR + (TotalRetained- Sum.limited.Var)/(1-alpha)
    Output          <- list(Prob.fct(u1,u2,rho,y=1.1),VaR,CTE)
    return(Output)
    }
RTCmax    <- 0.2
alpha     <- 0.7
u1base    <- seq(0.4, 0.9, length.out = 10) -> u2gen
for (iparam in 1:length(u1base)) {
       u2gen[iparam] = ATUnif(u1base[iparam], RTCmax) }
Prob.mat <- matrix(0,3,length(u1base))
VaR.mat  <- matrix(0,3,length(u1base))
CTE.mat  <- matrix(0,3,length(u1base))

for (iparam in 1:length(u1base)) {
  T1   <- ExLossCTEUnif(u1=u1base[iparam],u2=u2gen[iparam], rho=-0.5,alpha)
  Prob.mat[1,iparam] <- T1[[1]]
  VaR.mat[1,iparam]  <- T1[[2]]
  CTE.mat[1,iparam]  <- T1[[3]]
  T1 <- ExLossCTEUnif(u1=u1base[iparam],u2=u2gen[iparam], rho= 0.000001,alpha)
  Prob.mat[2,iparam] <- T1[[1]]
  VaR.mat[2,iparam]  <- T1[[2]]
  CTE.mat[2,iparam]  <- T1[[3]]
  T1 <- ExLossCTEUnif(u1=u1base[iparam],u2=u2gen[iparam], rho= 0.5,alpha)
  Prob.mat[3,iparam] <- T1[[1]]
  VaR.mat[3,iparam]  <- T1[[2]]
  CTE.mat[3,iparam]  <- T1[[3]]
  }

save(u1base,VaR.mat,CTE.mat,
     file = "../ChapTablesData/Chap5/Ex552FairCostsUniform.RData")
```

```
# Example 5.13 Illustrative Code, Figure 5.13
# save(u1base,VaR.mat,CTE.mat,
#      file = "../ChapTablesData/Chap5/Ex552FairCostsUniform.RData")
load( file = "ChapTablesData/Chap5/Ex552FairCostsUniform.RData")

par(mfrow = c(1,2))
  plot(u1base,VaR.mat[2,], type = "l", lty = 1,
       xlab = expression(u[1]), ylab = "VaR", ylim=c(0.90,1.1))
    lines(u1base,VaR.mat[1,],  col=FigRed, lty=3)
    lines(u1base,VaR.mat[3,],  col=FigBlue, lty=2)
  plot(u1base,CTE.mat[2,], type = "l", lty = 1,
       xlab = expression(u[1]), ylab = "ES", ylim=c(1,1.2))
    lines(u1base,CTE.mat[1,],  col=FigRed, lty=3)
    lines(u1base,CTE.mat[3,],  col=FigBlue, lty=2)
```

15.6.15 Example 5.14. Visualizing the Objective Function with an Active Constraint

Lower/Upper Bounds for Active Trade-off Function

```
# Example 5.14 Illustrative Code
# Lower/Upper Bounds for Active Trade-off Function AT

AT.fct <- function(u1,RTCmax){
  RC2x <- function(x){RTCmax - RC1(u1) - RC2(x)}
  u2star <- 1.0e8 #1.2e4
  if (RTCmax - RC1(u1) >= RC2(0)) {u2star <- 0}
  if ((RTCmax - RC1(u1) > 0)*(RTCmax - RC1(u1) < RC2(0))) {
       u2star <- uniroot(RC2x, lower =  0, upper =1e9)$root}
  return(u2star)
}

ATu.fct <- function(u1){AT.fct(u1,RTCmax=RTCmax)}

f0.VaRu <- function(u1) {
  u2=ATu.fct(u1)
  Prob.alpha <- function(y){3e5*(Prob.fct(u1,u2,rho,y) - alpha)}
  VaR <- uniroot(Prob.alpha, lower = 0, upper = 5e4)$root
  return(VaR)
  }
f0.CTEu <- function(u1) {
  u2=ATu.fct(u1)
  Prob.alpha <- function(y){Prob.fct(u1,u2,rho,y) - alpha}
  VaR <- uniroot(Prob.alpha, lower = 0, upper = 5e4)$root
  Surv.fct.y <- function(y){1-Prob.fct(u1,u2,rho,y)}
  Surv.fct.y.vec <- Vectorize(Surv.fct.y)
  Sum.limited.Var <- integrate(Surv.fct.y.vec, lower = 0, upper = VaR)$value
  TotalRetained <-
      actuar::levgamma(limit = u1,shape = risk1shape, scale = risk1scale) +
      actuar::levpareto(limit = u2,shape = risk2shape, scale = risk2scale)
    CTE <- VaR + (TotalRetained- Sum.limited.Var)/(1-alpha)
  return(CTE)
  }
```

Single Parameter Optimization

```
# Example514A
Ubdd.AT.fct <- function(u){RTCmax - RC2(0) - RC1(u)}
Upperbdd    <- 1.0e5
if (RTCmax - RC2(0) > 0) {
       Upperbdd <- uniroot(Ubdd.AT.fct, lower = 0, upper =1e6)$root}
Lbdd.AT.fct <- function(u){RTCmax - RC1(u)}
Lowerbdd    <- uniroot(Lbdd.AT.fct, lower = 0, upper =1e6)$root
u1opt       <- optimize(f0.VaRu, interval = c(Lowerbdd, 4000), tol = 0.1)
u1opt2      <- optimize(f0.CTEu, interval = c(Lowerbdd, Upperbdd), tol = 0.1)
```

```
u1VaRopt.1    <- round(u1opt$minimum,digits=2)
u2VaRopt.1    <- round(AT.fct(u1=u1VaRopt.1+.001,RTCmax),digits=2)
VaRu1VaRopt.1 <- round(ExLossESGamPareto(u1=u1VaRopt.1,u2=u2VaRopt.1,
                                    rho=0.5, alpha = 0.85)[[2]],digits=2)
CTEu1VaRopt.1 <- round(ExLossESGamPareto(u1=u1VaRopt.1,u2=u2VaRopt.1,
                                    rho=0.5, alpha = 0.85)[[3]],digits=2)
u1CTEopt.1    <- round(u1opt2$minimum,digits=2)
u2CTEopt.1    <- round(AT.fct(u1=u1CTEopt.1+.001,RTCmax),digits=2)
VaRu1CTEopt.1 <- round(ExLossESGamPareto(u1=u1CTEopt.1,u2=u2CTEopt.1,
                                    rho=0.5, alpha = 0.85)[[2]],digits=2)
CTEu1CTEopt.1 <- round(ExLossESGamPareto(u1=u1CTEopt.1,u2=u2CTEopt.1,
                                    rho=0.5, alpha = 0.85)[[3]],digits=2)

save(u1VaRopt.1, u2VaRopt.1, VaRu1VaRopt.1,CTEu1VaRopt.1,
     u1CTEopt.1, u2CTEopt.1, VaRu1CTEopt.1,CTEu1CTEopt.1,
     file = "../ChapTablesData/Chap5/Ex555GamParetoVisualB.RData")
```

Figure 5.15

```
# Fig515Plot
RCostMax <- seq(from = 4000, to=500, by = -500)
OutMat.Rmax <- matrix(0,nrow = length(RCostMax), ncol = 3)
colnames(OutMat.Rmax) <- c( "RCostMax",  "u1LowerBdd", "u1UpBdd")

for (jCost in 1:length(RCostMax)){
  RTCmax       <- RCostMax[jCost] -> OutMat.Rmax[jCost,]
  # Lower/Upper Bounds for Active Trade-off Function AT
  Ubdd.AT.fct1 <- function(u){RTCmax - RC2(0) - RC1(u)}
  Upperbdd     <- 1.0e8
  if (RTCmax - RC2(0) > 0) {
       Upperbdd <- uniroot(Ubdd.AT.fct1, lower = 0, upper =5e4)$root}
  Lbdd.AT.fct1 <- function(u){RTCmax - RC1(u)}
  Lowerbdd     <- uniroot(Lbdd.AT.fct1, lower = 0, upper =5e4)$root
  OutMat.Rmax[jCost,2] <- Lowerbdd+.001
  OutMat.Rmax[jCost,3] <- min(Upperbdd, 2.2e4)
  }

f0.VaRuRTCmax <- function(u1,RTCmax) {
  u2         <- AT1.fct(u1,RTCmax)
  Prob.alpha <- function(y){3e5*(Prob.fct(u1,u2,rho,y) - alpha)}
  VaR        <- uniroot(Prob.alpha, lower = 0, upper = 5e4)$root
  return(VaR) }

NumUs = 10
Var.mat <- matrix(0,nrow = length(RCostMax), ncol = NumUs) -> U.mat
for (jCost in 1:length(RCostMax)){
  U.mat[jCost,] <-
    seq(from=OutMat.Rmax[jCost,2], to=OutMat.Rmax[jCost,3], length.out = NumUs)
  for (kCost in 1:NumUs){
```

```
      Var.mat[jCost,kCost] <-
        f0.VaRuRTCmax(u1=U.mat[jCost,kCost],RTCmax=RCostMax[jCost])
      }}
uVec1 <- seq(from=OutMat.Rmax[1,2], to=OutMat.Rmax[1,3], length.out = NumUs)

f0.CTEuRTCmax <- function(u1,RTCmax) {
    u2              <- AT1.fct(u1,RTCmax)
    Prob.alpha      <- function(y){Prob.fct(u1,u2,rho,y) - alpha}
    VaR             <- uniroot(Prob.alpha, lower = 0, upper = 5e4)$root
    Surv.fct.y      <- function(y){1-Prob.fct(u1,u2,rho,y)}
    Surv.fct.y.vec  <- Vectorize(Surv.fct.y)
    Sum.limited.Var <- integrate(Surv.fct.y.vec, lower = 0, upper = VaR)$value
    TotalRetained   <-
      actuar::levgamma(limit = u1,shape = risk1shape, scale = risk1scale) +
      actuar::levpareto(limit = u2,shape = risk2shape, scale = risk2scale)
    CTE             <- VaR + (TotalRetained- Sum.limited.Var)/(1-alpha)
    return(CTE)
    }

CTE.mat <- matrix(0,nrow = length(RCostMax), ncol = NumUs) -> U.mat
for (jCost in 1:length(RCostMax)){
  U.mat[jCost,] <-
    seq(from=OutMat.Rmax[jCost,2], to=OutMat.Rmax[jCost,3], length.out = NumUs)
  for (kCost in 1:NumUs){
      CTE.mat[jCost,kCost] <-
        f0.CTEuRTCmax(u1=U.mat[jCost,kCost],RTCmax=RCostMax[jCost])
      }}
uVec1 <- seq(from=OutMat.Rmax[1,2], to=OutMat.Rmax[1,3], length.out = NumUs)

save(U.mat, Var.mat,CTE.mat, RCostMax,
        file = "../ChapTablesData/Chap5/Ex555Fig515VaRCTEActive.RData")
```

15.7 Chapter Seven Code

15.7.1 Example 7.1. Optimal Retention Policies for ANU Risks

Retrieve Data and Set-up Objective and Constraint Functions

```
# Example 7.1 Illustrative Code
#  Retrieve ANU Simulated Data
load(file = "Data/SimData/AggClaimsMay100Kw0.7.Rdata")


Xsim   <- Claims[,-1]
alpha <- 0.8
nrisk  <- ncol(Xsim)
RTCmax <- 433

QSRisks <- c(6, 7, 12)
XLRisks <- c(1:5, 8:11, 13:nrisk)

starter1.param <- rep(100, 14)
for (j in 1:nrisk){ starter1.param[j] <- quantile(p=0.90, Xsim[,j]) }
starter2.param <- rep(0.8, nrisk)


ANUStdRTC.fct <- function(param, Xsim = Xsim, Type = 1){
  RetClaims <- rep(0, nrow(Xsim) )

  if (Type == 1){ # Excess of Loss
    for (kidx in 1:nrisk){RetClaims <- RetClaims +
                          pmin(Xsim[,kidx], param[kidx]) }
    }
  if (Type == 2) {# Quota Share
       RetClaims <- as.vector(Xsim %*% param)
    }
  if (Type == 3){ #  Combo
      for (kidx in XLRisks){ RetClaims <- RetClaims +
                             pmin(Xsim[,kidx], param[kidx]) }
      for (kidx in QSRisks){ RetClaims <- RetClaims +
                             Xsim[,kidx] * param[kidx] }
      #  RetClaims <- pmin( (param[1]*Xsim[,1]+
      #                param[2]*Xsim[,2]) , param[3]) Umbrella
    }
  CeededClaims <- rowSums(Xsim) - RetClaims
  RTC.R     <- mean( CeededClaims )
  stddev    <- sd(RetClaims)
  VaR       <- as.numeric(quantile(RetClaims, prob = alpha, na.rm = TRUE))
  ES        <- as.numeric(VaR   +
                          mean(pmax(RetClaims - VaR,   0)) / (1-alpha) )
  return( c(RTC.R, VaR, ES, stddev) )
```

```
}
#  Objective Functions
f01.fct <- function(param){ ANUStdRTC.fct(param, Xsim, Type = 1)[4] }
f02.fct <- function(param){ ANUStdRTC.fct(param, Xsim, Type = 2)[4] }
f03.fct <- function(param){ ANUStdRTC.fct(param, Xsim, Type = 3)[4] }
#  Constraint Functions
Pmat.fct <- function(param, Type) {
  h <- NA
  h[1] <- RTCmax - ANUStdRTC.fct(param, Xsim, Type)[1]
  for (j in 1:nrisk){ h[j+1] <- 1e6*param[j] }
  if (Type == 2) {
     for (j in 1:nrisk){h[nrisk + 1 + j] <- 1e6 * (1.0 - param[j]) }
     }
  if (Type == 3) {
     QSRisks <- c(6, 7, 12)
     for (j in 1:length(QSRisks)){h[nrisk + 1 + j] <-
                             1e6 * (1.0 - param[ QSRisks[j] ]) }
     }
  return(h)
}

Pmat1.fct <- function(param) { Pmat.fct(param, Type = 1) }
Pmat2.fct <- function(param) { Pmat.fct(param, Type = 2) }
Pmat3.fct <- function(param) { Pmat.fct(param, Type = 3) }
```

Optimize Portfolios

```
# Example 7.1 Illustrative Code

#  Optimization using 'alabama' package
TimeCheckA <- Sys.time()
fXL.opt <- alabama::auglag( par = starter1.param,
                            fn  = f01.fct,  hin = Pmat1.fct,
           control.outer=list(method="nlminb",trace=FALSE) )
Sys.time() - TimeCheckA #Time difference of 8.602036 mins

TimeCheckA <- Sys.time()
fQS.opt <- alabama::auglag( par = starter2.param,
                            fn  = f02.fct,  hin = Pmat2.fct,
           control.outer=list(method="nlminb",trace=FALSE) )
Sys.time() - TimeCheckA #Time difference 6.897557 mins

TimeCheckA <- Sys.time()
starter3.param <- fXL.opt$par
for (kidx in QSRisks){ starter3.param[kidx] <- fQS.opt$par[kidx] }
fcombo.opt <- alabama::auglag( par = starter3.param,
                               fn  = f03.fct,  hin = Pmat3.fct,
           control.outer=list(method="nlminb",trace=FALSE) )
Sys.time() - TimeCheckA #Time difference of 3.871779 mins
```

```
save(fXL.opt, fQS.opt, fcombo.opt,
     file= "../ChapTablesData/Chap7/Example711Feb2024.Rdata")
```

15.7.2 Example 7.2. Bivariate Excess of Loss Using VaR Optimization

Simulate Data

```
# Example 7.2 Illustrative Code
# Generate Bivariate Gamma-Pareto Simulated Data

#  Set Simulation Parameters
nsim <- 50000
set.seed(202020)
q1 <- function(x){qgamma(p=x,shape = risk1shape, scale = risk1scale)}
q2 <- function(x){actuar::qpareto(p=x,shape = risk2shape,
                                  scale = risk2scale)}
ERisk1 <- actuar::mgamma(order=1, shape = risk1shape,
                         scale = risk1scale)
RC1    <- function(u){ ERisk1 -
    actuar::levgamma(limit = u,shape = risk1shape, scale = risk1scale)}
ERisk2 <- actuar::mpareto(order=1, shape = risk2shape, scale = risk2scale)
RC2    <- function(u){ ERisk2 -
    actuar::levpareto(limit = u,shape = risk2shape, scale = risk2scale)}
RTC    <- function(u1,u2){RC1(u1)+RC2(u2)}

#  Normal, or Gaussian, Copula
corrparams <- c(0.5)
norm.cop <- copula::normalCopula(param=corrparams, dim = 2,'dispstr' ="un")
UCop     <- copula::rCopula(nsim, norm.cop)

X1 <-  q1(UCop[,1])
X2 <-  q2(UCop[,2])

RTCmax <- 0.30*(ERisk1 + ERisk2)
```

Basic Simulation VaR Optimization

```
# Example 7.2 Illustrative Code
#Basic VaR Simulation Approach

f0.RVaR <- function(par) {
  u1=par[1];u2=par[2]
  sumYj <- pmin(u1, X1) + pmin(u2, X2)
  VaR   <- quantile(sumYj, prob = alpha)
  return(VaR) }

fineq <- function(par) {
```

```
  u1=par[1]; u2 = par[2]
  h    <- NA
  h[1] <- RTCmax - RTC(u1,u2)
  h[2] <- par[1]
  h[3] <- par[2]
  return(h)}

starter <- c(3500 , q2(0.999))  # c(4000 , q2(0.999)) did not work well

ala.auglag.opt <-
  alabama::auglag(par=starter,             # initial value
                  fn= f0.RVaR,             # objective function
                  hin=fineq,               # inequality constraint
                  control.outer=list(method="nlminb",trace=FALSE,itmax=20),
                  control.optim=list(maxit=20) )
```

Basic Simulation Optimization Using the Distribution Function

```
# Example 7.2 Illustrative Code
# Distribution Function Approach

h1.in <- function(coeff) {
  u1=coeff[1];u2=coeff[2]; y = coeff[3]
  sumYj <- pmin(u1, X1) + pmin(u2, X2)
  return(1e6*(mean( sumYj <= y) - alpha)) }

fineq.df <- function(coeff) {
  u1=coeff[1];u2=coeff[2]; y = coeff[3]
  h    <- NA
  h[1] <- h1.in(coeff)
  h[2] <- RTCmax - RTC(u1,u2)
  h[3] <- u1
  h[4] <- u2
  h[5] <- y
  return(h)}

x.arg <- function(coeff){coeff[3]}
startera <- c(3500 , q2(0.999), 6500)

ala.auglag1.opt <-
  alabama::auglag(par=startera, # initial value
  fn= x.arg,                    # objective function
  hin=fineq.df,                 # inequality constraint
  control.outer=list(method="nlminb",trace=FALSE, itmax=20),
  control.optim=list(maxit=20) )
```

15.7.3 Example 7.3. Bivariate Excess of Loss Using *ES* Optimization

Optimize Using ES

```
# Example 7.3 Illustrative Code

#Basic ES simulation Approach

f0.RCTE <- function(par) {
  u1=par[1];u2=par[2]
  sumYj <- pmin(u1, X1) + pmin(u2, X2)
  VaR      <- quantile(sumYj, prob = alpha)
  CTE      <- VaR + mean(pmax(sumYj - VaR, 0))/(1-alpha)
  return(CTE) }

fineq <- function(par) {
  u1=par[1]; u2 = par[2]
  h <- NA
  h[1] <- RTCmax - RTC(u1,u2)
  h[2] <- par[1]
  h[3] <- par[2]
  return(h)}

starter <- c(4200 , 600)  # c(4000 , q2(0.999)) did not work well

ala.auglag.opt <-
  alabama::auglag(par=starter,    # initial value
                  fn= f0.RCTE,    # objective function
                  hin=fineq,      # inequality constraint
                  control.outer=list(method="nlminb",
                                     trace=FALSE, itmax=20),
                  control.optim=list(maxit=20))
```

Optimize Using Display (7.4)

```
# Example 7.3 Illustrative Code
# Minimizer of a RM Approach

f0.dfRCTE <- function(coeff){
  u1 <- coeff[1]; u2 <- coeff[2]; VaR.x <- coeff[3]
  sumYj <- pmin(u1, X1) + pmin(u2, X2)
  CTE1     <- VaR.x + mean(pmax(sumYj- VaR.x, 0)) /(1-alpha)
  c(CTE1)
  }
fineq.df <- function(coeff) {
  u1=coeff[1];u2=coeff[2]; VaR.x = coeff[3]
  h <- NA
  h[1] <- RTCmax - RTC(u1,u2)
  h[2] <- u1
  h[3] <- u2
```

```
  h[4] <- VaR.x
  return(h)
  }
startera <- starter <- c(4200 , 800, 5000)
CTEMinEx711.opt <- alabama::auglag(par=startera, # initial value
                  fn= f0.dfRCTE,               # objective function
                  hin=fineq.df,                # inequality constraint
                  control.outer=list(method="nlminb",trace=FALSE, itmax=20),
                  control.optim=list(maxit=20))
```

Optimize Using Deterministic Approach

```
# Example73Det
# Illustrative Code Using Deterministic minimizer of a RM
#  Not used going forward
timeDet <- Sys.time()
f0Deter.CTE <- function(coeff) {
  u1 <- coeff[1]; u2 <- coeff[2]; VaR.x <- coeff[3]
    Surv.fct.y      <- function(y){1-Prob.fct(u1,u2,rho,y)}
    Surv.fct.y.vec  <- Vectorize(Surv.fct.y)
    Sum.limited.Var <- integrate(Surv.fct.y.vec,lower = 0,upper = VaR.x)$value
    TotalRetained   <-
      actuar::levgamma(limit = u1,shape = risk1shape, scale = risk1scale) +
      actuar::levpareto(limit = u2,shape = risk2shape, scale = risk2scale)
    CTE <- VaR.x + (TotalRetained- Sum.limited.Var)/(1-alpha)
  return(CTE) }

fineq.df <- function(coeff) {
  u1=coeff[1];u2=coeff[2]; VaR.x = coeff[3]
  h <- NA
  h[1] <- RTCmax - RTC(u1,u2)
  h[2] <- u1
  h[3] <- u2
  h[4] <- VaR.x
  return(h)
  }
startera <- CTEMinEx711.opt$par #starter <- c(4200 , 800, 5000)

CTEMinEx711Det.opt <-
  alabama::auglag(par=startera,           # initial value
                  fn= f0Deter.CTE,        # objective function
                  hin=fineq.df,           # inequality constraint
                  control.outer=list(method="nlminb",trace=FALSE, itmax=20),
                  control.optim=list(maxit=20))

save(CTEMinEx711Det.opt,
     file = "../ChapTablesData/Chap7/CTEDeterEx711.Rdata")

# CTEMinEx711Det.opt$par
```

```
# CTEMinEx711Det.opt$value
# CTEMinEx711Det.opt$outer.iterations
# CTEMinEx711Det.opt$convergence
# fineq.df(CTEMinEx711Det.opt$par)

#Sys.time() - timeDet
# > Sys.time() - timeDet
# Time difference of 20.1229366 mins
```

15.7.4 Example 7.4. Quota Sharing of ANU Risks Using CVXR

```
# Example 7.4 Illustrative Code
library(CVXR)
load(file = "../Data/SimData/AggClaimsMay100Kw0.7.Rdata")
Xsim    <- Claims[,-1]
ANU.mu  <- colMeans(Xsim, na.rm = FALSE)
numVars <- ncol(Xsim)
nsim    <- nrow(Xsim)
TotCost <- sum(ANU.mu)
Sigma   <- cov(Xsim)
RTCmax  <- 433
alpha   <- 0.90

pos_part <- function(z) { 0.5 * ( z + abs(z) ) }
# Summarize Results
SumCVXR <- function(theta){
  RTC      <- TotCost - theta %*% ANU.mu
  Variance <- t(theta) %*% Sigma %*% theta
  Claims.retained <- Xsim %*% theta
  VaR      <- quantile(Claims.retained, prob = alpha)
  ES      <- VaR + mean(pmax(Claims.retained - VaR, 0))/(1-alpha)
  output   <- c(RTC, VaR, ES, sqrt(Variance) )
  return(output)
}

### Minimize Expected Shortfall (1)

coeff         <- Variable(numVars+1)
VaR.z0        <- coeff[1]
theta.vec     <- coeff[-1]
Claims.retained <- Xsim %*% theta.vec
RTC           <- TotCost - t(theta.vec) %*% ANU.mu
constraints   <- list(RTC <= RTCmax,
                      coeff >= 0, theta.vec <= 1)

objES         <- VaR.z0 +
                 mean( pos_part(Claims.retained  - VaR.z0) ) / (1-alpha)
probES        <- Problem(Minimize(objES), constraints)
CVXR.ESresult <- solve(probES, solver = "ECOS")
```

```
coeffES.opt   <- as.vector(CVXR.ESresult$getValue(coeff))

### Minimize the variance

theta.vec     <- Variable(numVars)
Claims.retained <- Xsim %*% theta.vec
RTC           <- TotCost - t(theta.vec) %*% ANU.mu
constraints   <- list(RTC <= RTCmax,
                      theta.vec >= 0, theta.vec <= 1)

objVar         <- quad_form(theta.vec, Sigma)
probVar        <- Problem(Minimize(objVar), constraints)
CVXR.Varresult <- solve(probVar, solver = "ECOS")
thetaVar.opt   <- as.vector( CVXR.Varresult$getValue(theta.vec) )
#Results Summary
OutMatCVXR <- rbind(
   c(SumCVXR(thetaVar.opt),  thetaVar.opt ),
   c(SumCVXR(coeffES.opt[-1]), coeffES.opt[-1])
  )

save(OutMatCVXR,
     file = "../ChapTablesData/Chap7/Example723CVXR.Rdata")
```

15.7.5 Example 7.5. Bivariate Excess of Loss Using VaR Optimization and Smoothing

```
# Example 7.5 Illustrative Code
# Distribution Function Approach
# Try a smooth estimate of the dist function

x.arg <- function(coeff){coeff[3]}
startera <- c(3500 , q2(0.999), 6500)

library(ks)
h2.in <- function(coeff) {
  u1=coeff[1];u2=coeff[2]; y = coeff[3]
  sumYj <- pmin(u1, X1) + pmin(u2, X2)
  EstProb <- ks::kcde(sumYj, eval.points = y)$estimate
  return(1e6*(EstProb - alpha)) }

fineq2.df <- function(coeff) {
  u1=coeff[1];u2=coeff[2]; y = coeff[3]
  h    <- NA
  h[1] <- h2.in(coeff)
  h[2] <- RTCmax - RTC(u1,u2)
  h[3] <- u1
  h[4] <- u2
  h[5] <- y
```

```
  return(h)}

KernSmoothEx711.opt <-
  alabama::auglag(par=startera,
                  fn= x.arg,
                  hin=fineq2.df,
                  control.outer=list(method="nlminb",trace=FALSE,
                                     itmax=20),
                  control.optim=list(maxit=20))
save(KernSmoothEx711.opt,
     file = "../ChapTablesData/Chap7/CTEKernSmoothEx711.Rdata")
```

15.7.6 Section 7.4. Constructing a Portfolio Frontier

Remark. This is a set of code that can be used for either excess of loss or quota share, utilizing kernel smoothing and gradients to speed convergence.

```
# ESOptimization
# These functions are for ES minimization of
#  Excess of Loss (TRUE) and Quota Share (ExLoss = FALSE)

# Function to Compute Retained Claims
RetClaims.fct <- function(coeff, Xsim, ExLoss = TRUE){
  VaR.z0    <- coeff[1]
  theta.vec <- coeff[-1]
  p.num     <- length(theta.vec)
  RetClaims <- rep(0, nrow(Xsim) )
  if (ExLoss){  # Excess of Loss
     for (kidx in 1:p.num){RetClaims <- RetClaims +
                             pmin(Xsim[,kidx], theta.vec[kidx]) }
     } else {  # Quota Share
            RetClaims  <- as.vector(Xsim %*% theta.vec)   }
  return( RetClaims )
}  # end RetClaims.fct function

#  ESKernmin.fct evaluates the ES objective function
#    using kernel smoothing
ESKernmin.fct <- function(coeff, RetClaims){
  VaR.z0        <- coeff[1]
  theta.vec     <- coeff[-1]
  ESKern1.fct   <- function(ytidle) { as.numeric( mean(
                       pmax(RetClaims + bw * ytidle - VaR.z0, 0) ) ) *
                              dnorm(ytidle)  }
  ESKernmin     <- VaR.z0 + integrate(Vectorize(ESKern1.fct),
                                      lower = -6, upper = 6)$value / (1-alpha)
  return( ESKernmin )
}  # end ESKernmin.fct function

#Function to Compute Simulated RTC
RTCR.fct <- function(Xsim, RetClaims){
```

```
  CeededClaims <- rowSums(Xsim) - RetClaims
  RTC.R     <- mean( CeededClaims )
  return( RTC.R )
}  # end RTCR.fct function

#Function to Compute Simulated RTC with Kernel Smoothing
RTCKernR.fct <- function(jVar, coeff, Xsim, RetClaims){
  theta.vec     <- coeff[-1]
  RTCKern1.fct  <- function(ytidle) { as.numeric( mean(
                       pmax(Xsim[,jVar] - theta.vec[jVar] + bw * ytidle, 0) ) ) *
                              dnorm(ytidle)  }
  RTCKern      <- integrate(Vectorize(RTCKern1.fct),
                                    lower = -6, upper = 6)$value
  return( RTCKern )
}  # end RTCKernR.fct function

# Function to Summarize Risk Measures
ESSummary.fct <- function(coeff, RetClaims){
  VaR.z0    <- coeff[1]
  theta.vec <- coeff[-1]
  ones      <- rep(1, length(RetClaims) )
  stddev    <- sd(RetClaims)
  VaR       <- as.numeric(quantile(RetClaims, prob = alpha, na.rm = TRUE))
  Kern.df   <- function(y){ yarg <- (y*ones - RetClaims)/bw
                            return(mean(pnorm(yarg)) - alpha)}
  Kern.VaR  <- uniroot(f=Kern.df,lower = VaR - 100, upper = VaR + 100)$root
  ESmin     <- as.numeric(VaR.z0 +
                            mean(pmax(RetClaims - VaR.z0, 0)) /(1-alpha) )
  ES        <- as.numeric(VaR   +
                            mean(pmax(RetClaims - VaR,   0)) /(1-alpha) )
  ESKern1.fct <- function(y) { as.numeric ( mean( pmax(
                           RetClaims -  VaR.z0 +  bw * y, 0) ) ) * dnorm(y) }
  ESKernmin   <- VaR.z0 + integrate(Vectorize(ESKern1.fct),
                                    lower = -6, upper = 6)$value / (1-alpha)
  ESKern2.fct <- function(y) { as.numeric ( mean( pmax(
                           RetClaims - Kern.VaR + bw * y, 0) ) ) * dnorm(y) }
  ESKern      <- Kern.VaR + integrate(Vectorize(ESKern2.fct), lower = -6,
                                      upper = 6)$value / (1-alpha)
  return( c(VaR, Kern.VaR, ESmin, ES,  ESKernmin, ESKern, stddev) )
}  # end ESSummary.fct function

# Gradient of the ES Objective Function
f0z.fct <- function(coeff, Xsim, RetClaims, ExLoss = TRUE){
  VaR.z0        <- coeff[1]
  theta.vec     <- coeff[-1]
  numLoss       <- nrow(Xsim)
  ones          <- rep(1, numLoss )
  VaR.z0arg     <- (VaR.z0*ones - RetClaims) / bw
  partial.z0    <- 1 - mean( 1-pnorm(VaR.z0arg) ) / (1 - alpha)

  if (ExLoss){  # Excess of Loss
```

```
              mat.check <-  t( 1* ( t(Xsim) > theta.vec) )
      } else {  # Quota Share
              mat.check <- Xsim  }
  temp          <- ( 1-pnorm(VaR.z0arg) ) %*% mat.check / numLoss
  partial.theta <- temp / (1 - alpha)
  return( c(partial.z0, partial.theta) )
  }  # end f0z.fct function

# Function to Construct a Frontier of Optimal Portfolios
#  p.z is the number of coefficients (including VaR)
ThetaOptim.fct <- function(Xsim, ExLoss = TRUE, p.z = 15,
                           RTCmax.vec, starter.coeff) {
  Output.mat <- matrix(0,nrow = length(RTCmax.vec), ncol = 1+6+p.z)
  f0.ES      <- function(coeff){ RetClaims <-
                                 RetClaims.fct(coeff, Xsim, ExLoss)
                return(ESKernmin.fct(coeff, RetClaims) ) }
  f0.grad    <- function(coeff){ RetClaims <-
                                 RetClaims.fct(coeff, Xsim, ExLoss)
                return(f0z.fct(coeff, Xsim, RetClaims, ExLoss) ) }
  RTC.optim  <- function(coeff){ RetClaims <-
                                 RetClaims.fct(coeff, Xsim, ExLoss)
                                 return(RTCR.fct(Xsim,RetClaims) )}
  # Use RTCR.fct for empirical RTCs, RTC.fct for lognormal

  for (jCost in 1:length(RTCmax.vec)){    # Start jCost Loop
    hin.ES <- function(coeff) {h <- NA    # Constraints in alabama
      h[1] <- RTCmax.vec[jCost] - RTC.optim(coeff)
      for (j in 1:p.z){h[j+1] <- 1e6*coeff[j] }
      if (ExLoss==FALSE){                 # For Quota Share
        for (j in 2:p.z){h[j+p.z] <-  1e6 * (1.0 - coeff[j]) }
      }
      return(h)
      }
  if (jCost > 1 ){starter.coeff <- ES.opt$par*1.1}
    z.params <- rep(0, p.z); LME.EX <- 0
    tryCatch({
        ES.opt <- alabama::auglag( par=starter.coeff,
           fn  = f0.ES,         # objective function
           gr  = f0.grad,       # gradient objective function
           hin = hin.ES,        # constraints
           control.outer=list(method="nlminb",trace=FALSE) )
        z.params <- ES.opt$par
        LME.EX <- ES.opt$lambda[1] },
        error=function(err) {
        z.params <- starter.coeff
        LME.EX <- 0} )
    RetClaims <- RetClaims.fct(z.params, Xsim, ExLoss)
    RTCR      <- RTC.optim(z.params)
    output    <- c(RTCmax.vec[jCost], RTCR ,
                   ESSummary.fct(z.params, RetClaims)[-c(3,5)],
                   z.params[-1] , LME.EX)
```

```
    cat("Theory Results",output, "\n")
    Output.mat[jCost,] <- output
    }  # end jCost Loop
    return(Output.mat)
    }  # end ThetaOptim.fct function
```

15.7.7 Example 7.6 Constructing a Portfolio Frontier for ANU Risks

Set-Up

```
# Example76
load(file = "../Data/SimData/AggClaimsMay100Kw0.7.Rdata")

Xsim          <- Claims[,-1]  # Remove Property
alpha         <- 0.80
RTCmax.vec    <- as.vector(433)
starter.coeff <- rep(0, 15 )
for (kidx in 1:14){ starter.coeff[kidx+1] <- quantile(Xsim[,kidx], p= alpha) }
starter.coeff[1] <- 2000
bw = 1

Time1 <- Sys.time()
Example741 <- ThetaOptim.fct(Xsim, ExLoss = TRUE, p.z = 15,
                             RTCmax.vec, starter.coeff)
Sys.time() - Time1   # 2.18 minutes

save(Example741,
     file = "../ChapTablesData/Chap7/Example741.Rdata")
```

Determine the Frontier

```
# Example76Eval
# Bring in ANU Claims Data
load(file = "../Data/SimData/AggClaimsMay100Kw0.7.Rdata")
Xsim    <- Claims[,-1]
TotCostANU <- mean (rowSums(Xsim))
CstLevNum <- 1 - c(0.05, seq(0.1, 0.9, length.out=9),0.95)
CstLevNum <- round(CstLevNum*100)/100
ANURTCmax.vec <- TotCostANU * CstLevNum

#  A Few Key Parameters
alpha     <- 0.95
bw        <- 1

starter.coeff <- rep(10, 15)

Time1 <- Sys.time()
OutMatANU.ES <- ThetaOptim.fct(Xsim, ExLoss = TRUE, p.z = 15,
```

```
                              RTCmax.vec=ANURTCmax.vec, starter.coeff)
Sys.time() - Time1   #58.55217 minutes for alpha = 0.95
                     #38.83844 for alpha = 0.80

u.names <- paste("u",1:14, sep="")
colnames(OutMatANU.ES) <- c("RTCmax", "RTC", "VaR", "Kern.VaR", "ESmin",
                            "ES", "StdDev",  u.names ,"LME")
save(ANURTCmax.vec, OutMatANU.ES,
     file= "../ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")
```

Plot Uncertainty Measures

```
#save(ANURTCmax.vec,OutMatANU.ES,
#     file= "../ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")
load(file= "ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")

dfPlot <- data.frame(OutMatANU.ES)
p1 <- ggplot(dfPlot, aes(x=RTC, y=VaR)) + geom_point() + theme_bw() + geom_line()
p2 <- ggplot(dfPlot, aes(x=RTC, y=ES)) + geom_point() +
       theme_bw() + geom_line()  + labs(y = 'ES')
p3 <- ggplot(dfPlot, aes(x=RTC, y=StdDev)) + geom_point() + theme_bw() + geom_line()
grid.arrange(p1,p2,p3,nrow=1)
```

Plot Coefficients

```
# ExLossESCoeff
#save(ANURTCmax.vec,OutMatANU.ES,
#     file= "../ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")
load(file= "ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")
dfPlot <- data.frame(OutMatANU.ES)

# save(ANU.deduct, ANU.Prem, ANU.lambda, ANU.mu,  ANU.stdDev, ANU.quan95,
#      ANU.quan99, Ind.ANU.quan99, riskTypes, riskTypesShort,
#      file = "../ChapTablesData/Chap7/PremTable.Rdata")
load(file = "ChapTablesData/Chap7/PremTable.Rdata")
riskTypesShort <- riskTypesShort[-1] # Omit Property Risk
ANU.quan99 <- ANU.quan99[-1]
rndfac <- 50
numVars <- 14

pltfuncRTC <- function(num,var){
    ulimy <-  max(max(var), ANU.quan99[num] )
    ggplot(dfPlot, aes(x=RTC, y=var)) +
          geom_point() + theme_bw() + geom_line() +
          geom_line(aes(y=ANU.quan99[num]),linetype="dashed",
                    color = FigBlue, size = 0.5) +
          scale_y_continuous(breaks =
                               rndfac*round(c(ulimy/100, ulimy)/rndfac) ) +
```

```
          labs(y = riskTypesShort[num]) +
          theme(axis.title.y = element_text(size = 6)) +
          scale_x_continuous(breaks = seq(from = rndfac*round(min(dfPlot$RTC)/rndfac),
                                          to = rndfac*round(max(dfPlot$RTC)/rndfac),
                                          length.out = 3))
    }
q <- list(rep(0,numVars))
for (iPlot in 1:numVars){
  q[[iPlot]] <- pltfuncRTC(iPlot,dfPlot[[paste("u",iPlot, sep="")]])
}
if (numVars == 2) {grid.arrange(q[[1]],q[[2]], nrow=1)}
if (numVars == 3) {grid.arrange(q[[1]],q[[2]],q[[3]], nrow=1)}
if (numVars == 13) {grid.arrange(q[[1]],  q[[2]],  q[[3]],  q[[4]], q[[5]],
                                 q[[6]],  q[[7]],  q[[8]],  q[[9]], q[[10]],
                                 q[[11]], q[[12]], q[[13]],         nrow=5)}
if (numVars == 14) {grid.arrange(q[[1]],  q[[2]],  q[[3]],  q[[4]], q[[5]],
                                 q[[6]],  q[[7]],  q[[8]],  q[[9]], q[[10]],
                                 q[[11]], q[[12]], q[[13]], q[[14]], nrow=5)}
```

15.7.8 Section 7.4.2 Comparing Different Objective Functions

```
# Section742
#  Retrieve ANU Simulated Data
load(file = "Data/SimData/AggClaimsMay100Kw0.7.Rdata")
# ls(pos = 2)
# detach()

#save(ANURTCmax.vec,OutMatANU.ES,
#     file= "../ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")
load(file= "ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")  # For RTCmax.vec

Xsim   <- Claims[,-1]
alpha  <- 0.8
nrisk  <- ncol(Xsim)
bw = 1

ANUStdRTC1.fct <- function(param, Xsim){
  RetClaims    <- rep(0, nrow(Xsim) )
  for (kidx in 1:nrisk){RetClaims <- RetClaims +
                                     pmin(Xsim[,kidx], param[kidx]) }
  CeededClaims <- rowSums(Xsim) - RetClaims
  RTC.R        <- mean( CeededClaims )
  stddev       <- sd(RetClaims)
  return( c(RTC.R, stddev) )
  }
#  Objective Functions
f01.fct <- function(param){ ANUStdRTC1.fct(param, Xsim)[2] }

Time1 <- Sys.time()
```

```
ExLoss <- TRUE
#  p.z is the number of coefficients (including VaR)
  p.z <- 15
  Outputvar.mat <- matrix(0,nrow = length(ANURTCmax.vec),
                          ncol = 1 + 6 + p.z)
  starter.coeff <- rep(0.01, 15 ) ; starter.coeff[1] <- 100
  for (jCost in 1:length(ANURTCmax.vec)){         # Start jCost Loop
    Pmat1.fct <- function(param) {
    h <- NA
    h[1] <- ANURTCmax.vec[jCost] - ANUStdRTC1.fct(param, Xsim)[1]
    for (j in 1:nrisk){ h[j+1] <- 1e6*param[j] }
    return(h)
    }
  if (jCost > 1 ){starter.coeff <- var.opt$par*1.1}
    z.params <- rep(0, 15); LME.EX <- 0
    tryCatch({
        var.opt <- alabama::auglag( par=starter.coeff,
           fn = f01.fct,          # objective function
           #gr = f0.grad,         # gradient objective function
           hin = Pmat1.fct,        # constraints
           control.outer=list(method="nlminb",trace=FALSE) )
        z.params <- var.opt$par
        LME.EX <- var.opt$lambda[1] },
        error=function(err) {
        z.params <- starter.coeff
        LME.EX <- 0} )
    output <- c(ANURTCmax.vec[jCost],
                ESSummary.fct(z.params,Xsim, ExLoss)[-c(4,6)],
                z.params[-1] , LME.EX)
    cat("Theory Results",output, "\n")
    Outputvar.mat[jCost,] <- output
    }  # end jCost Loop
Sys.time() - Time1

save(Outputvar.mat,
     file= "../ChapTablesData/Chap7/Section742Feb2024.Rdata")
```

15.7.9 Section 7.5.4 Comparing Risk Retention Functions

```
# Section754QuanRegA
alpha = 0.80

load(file = "../Data/SimData/AggClaimsMay100Kw0.7.Rdata")
Xsim   <- Claims[,-1]
nsim <- nrow(Xsim)
numVars <- ncol(Xsim)
#save(ANURTCmax.vec, OutMatANU.ES,
#     file= "../ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")
load(file= "../ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")
```

```
# save(ANU.deduct, ANU.Prem, ANU.lambda, ANU.mu,  ANU.stdDev, ANU.quan95,
#      ANU.quan99, Ind.ANU.quan99, riskTypes, riskTypesShort,
#      file = "../ChapTablesData/Chap7/PremTable.Rdata")
load(file = "../ChapTablesData/Chap7/PremTable.Rdata")

ANU.mu <- ANU.mu[-1]
TotCost <- sum(ANU.mu)
library(CVXR)
X.sigma <- cov(Xsim)

q.names <- paste("c",1:numVars, sep="")

SumCVXRForm <- function(coeff,RTCmax){
  RTC      <- TotCost - t(coeff[1:numVars]) %*% ANU.mu
  Variance <- t(coeff[1:numVars]) %*% X.sigma %*% coeff[1:numVars]
  Claims.retained <- (Xsim %*% coeff[1:numVars])[,1]
  VaR      <- quantile(Claims.retained, prob = alpha)
  CTE      <- VaR + mean(pmax(Claims.retained - VaR, 0))/(1-alpha)
  output   <- c(round(c(RTCmax, RTC, VaR, CTE, sqrt(Variance)), digits=0),
                round(c(coeff[1:numVars]),digits=3))
  return(output)
}

time1 <- Sys.time()
OutMat.Q.VaR <- matrix(0,nrow = length(ANURTCmax.vec), ncol = 5+numVars)
colnames(OutMat.Q.VaR) <- c("RTCmax","RTC", "VaR", "ES", "Std Dev", q.names)

# https://cvxr.rbind.io/cvxr_examples/cvxr_quantile-regression/
quant_loss <- function(z) { 0.5 * abs(z) + (alpha - 0.5) * z }

for (jCost in 1:length(ANURTCmax.vec)){
  param   <- Variable(numVars+1)
  QRparam <- param[numVars+1]
  Claims.retained <- Xsim %*% param[1:numVars]/nsim
  RTC     <- TotCost - t(param[1:numVars]) %*% ANU.mu
  obj     <- sum(quant_loss(Claims.retained  - QRparam))
  constraints <- list(RTC <= ANURTCmax.vec[jCost],
                      param >= 0, param[1:numVars] <= 1)
  prob    <- Problem(Minimize(obj), constraints)
  result  <- solve(prob, solver = "ECOS") # "SCS")
  OutMat.Q.VaR[jCost,] <-
          SumCVXRForm(result$getValue(param),RTCmax=ANURTCmax.vec[jCost])
 cat("CVXR results",OutMat.Q.VaR[jCost,], "\n")
}
#  Based on nsim = 100000
save(OutMat.Q.VaR, file = "../ChapTablesData/Chap7/VaROptQuotaShare80.Rdata")
Sys.time() - time1 # Time difference of 1.971827 mins
```

15.7.10 Section 7.6 Simulation Uncertainty

```
# Section76SimUncertainty
#  Bring in Claims Data
load(file = "../Data/SimData/AggClaimsMay100Kw0.7.Rdata")
nsim    <- 5000  #100000 # For Boot
alpha   <- 0.80
bw      <- 1
numBoot <- 5 # 20
Xsim    <- Claims[1:(nsim*numBoot),-1]  #  For resample
#Xsim    <- Claims[1:nsim,-1]           #  For Boot
numVars <- ncol(Xsim)
starter.coeff <- rep(10, 15)
Output.array <- array(0, dim = c(11, 1 + 6 + 14 + 1, numBoot))

time1 <- Sys.time()
for (rBoot in 1:numBoot){
    block <- seq(from = (rBoot-1)*nsim+1, to = rBoot*nsim, by = 1) #  For resample
#    block <- sample(1:nsim, size=nsim, replace=TRUE)  #  For Bootstrap
    XsimBlock <- Xsim[block,]
    Output.array[,,rBoot] <- ThetaOptim.fct(Xsim = XsimBlock, ExLoss = TRUE,
                                            p.z = 15, RTCmax.vec=ANURTCmax.vec, starter.coeff)
    } #  end rBoot Loop

save(Output.array, file="../ChapTablesData/Chap7/SimUncertainFeb2024.Rdata") #  For resample
#save(Output.array,
#     file="../ChapTablesData/Chap7/SimUncertainFeb2024Boot.Rdata") #  For Boot
Sys.time() - time1
# For Resample Boot=20, nsim=5000, Time difference of 1.094439 hours
#  For bootsrap Boot = 5, nsim=100000, 4.357783 hours
```

15.8 Chapter Eight Code

15.8.1 Section 8.1.2. Risk Retention for a Specific Member

Load Data and Produce Table 8.1

```
#  CoverageTable2
load(file="Data/WiscPropFundData/dataout.Rdata")
PolNum = 2
# Coverage is in millions, not in logs
Coverage <- cbind(dataout$CoverageBC,
                  dataout$CoverageIM,
                  dataout$CoverageCN,
                  dataout$CoverageCO,
                  dataout$CoveragePN,
                  dataout$CoveragePO)
Coverage <- Coverage[PolNum,]
names(Coverage) <- c("BC", "IM", "CN", "CO", "PN", "PO")

TableGen1(TableData=t(Coverage),
         TextTitle='Six Coverage Amounts for Member 2',
         Align='r', Digits=3, ColumnSpec=1:5,
         BorderRight=1) %>%
  footnote(general = "Coverages are in millions of U.S. Dollars.",
           general_title = "Note:",
           footnote_as_chunk = TRUE)
```

Table 8.2

```
# AggClaimsForecast2
#save(Prop.array,
#     file = "../Data/WiscPropFundData/SimPropArray.RData")
load(file = "Data/WiscPropFundData/SimPropArray.RData")
SummaryOut <- Prop.array[,,PolNum]
dfSummaryOut <- data.frame(t(SummaryOut))

colnames(dfSummaryOut) <- c("BC", "IM", "CN", "CO", "PN", "PO")
library(doBy)
fun1 = function(x) { c(ma=min(x), m1=median(x),m=mean(x),
                       mq = quantile(x, probs =0.95),
                       mb=quantile(x, probs =0.99),
                       mz = 100*mean(x)/(mean(SummaryOut)*6) ) }

t1<- summaryBy(BC ~ 1, data = dfSummaryOut, FUN = fun1 )
t2<- summaryBy(IM ~ 1, data = dfSummaryOut, FUN = fun1 )
t3<- summaryBy(CN ~ 1, data = dfSummaryOut, FUN = fun1 )
t4<- summaryBy(CO ~ 1, data = dfSummaryOut, FUN = fun1 )
t5<- summaryBy(PN ~ 1, data = dfSummaryOut, FUN = fun1 )
```

```
t6<- summaryBy(PO ~ 1, data = dfSummaryOut, FUN = fun1 )

ClaimTable <- cbind(t(t1),t(t2),t(t3),t(t4),t(t5),t(t6))
rownames(ClaimTable) <- c("Minimum", "Median", "Mean","95th Percentile",
                          "99th Percentile", "Mean Percent of Total")
colnames(ClaimTable) <-  c("BC", "IM", "CN", "CO", "PN", "PO")

TableGen1(TableData=ClaimTable,
         TextTitle='Forecast Loss Distribution for Member 2',
         Align='r', Digits=2, ColumnSpec=1:5,
         ColWidth0= "2.5cm",
         BorderRight= 1, ColWidth= ColWidth6) %>%
  footnote(general = "Losses are in thousands of U.S. Dollars.",
           general_title = "Note:",
           footnote_as_chunk = TRUE)
```

Construct Optimal Frontier

```
# Section812ES
# This code utilizes the 'ThetaOptim.fct' and 'ESSummary.fct'
#   functions, introduced in Section 7.4
#   See 'ConstructPortFunctionsMar2024.Rmd'

PolNum = 2
#save(Prop.array,
#     file = "../Data/WiscPropFundData/SimPropArray.RData")
load(file = "../Data/WiscPropFundData/SimPropArray.RData")
Xsim <- t(Prop.array[,,PolNum])

TotCostProp2    <- mean (rowSums(Xsim))
CstLevNum       <- 1 - c(0.05, seq(0.1, 0.9, length.out=9),0.95)
CstLevNum       <- round(CstLevNum*100)/100
Prop2RTCmax.vec <- TotCostProp2 * CstLevNum

#  A Few Key Parameters
bw            <- 1
alpha         <- 0.80
starter.coeff <- c(10, rep(0.1, 6) )

Time1 <- Sys.time()
OutMatProp2.ES <- ThetaOptim.fct(Xsim, ExLoss = TRUE, p.z = 7,
                                 RTCmax.vec=Prop2RTCmax.vec, starter.coeff)
Sys.time() - Time1   #Time difference of 1.663976 mins alpha = 0.80

u.names <- paste("u",1:6, sep="")
colnames(OutMatProp2.ES) <- c("RTCmax", "RTC", "VaR", "Kern.VaR", "ES",
                              "ESKern", "StdDev",  u.names ,"LME")
#aaa
NaivePort.1 <- c(0, 1e10, rep(0,5))
```

```
RetNaive.1 <- RetClaims.fct(NaivePort.1, Xsim, ExLoss = TRUE)
NaivePort1Out <- c(RTCR.fct(Xsim, RetNaive.1),ESSummary.fct(RetNaive.1, Xsim)[c(1,4,7)])

names(NaivePort1Out) <-   c("RTC", "VaR", "ES", "StdDev" )
NaivePort.2   <- c(0,  0,  rep(100,5))
RetNaive.2 <- RetClaims.fct(NaivePort.2, Xsim, ExLoss = TRUE)
NaivePort2Out <- c(RTCR.fct(Xsim, RetNaive.2),ESSummary.fct(RetNaive.2, Xsim)[c(1,4,7)])
names(NaivePort2Out) <-   c("RTC", "VaR", "ES", "StdDev" )

save(Prop2RTCmax.vec, NaivePort1Out, NaivePort2Out,OutMatProp2.ES,
     file= "../ChapTablesData/Chap7/OutMatProp2.Feb2024.Rdata")
```

15.8.2 Section 8.1.3. Fund Risk Retention

Constructing a Portfolio Frontier for the Wisconsin Property Fund

Remark. The code employed for this section can be used either with excess of loss or quota share. It also utilizes kernel smoothing and gradients to speed convergence. It is similar to the Section 7.4 code on "Constructing a Portfolio Frontier" but modified so that limits are a fraction of the coverage amounts.

```
# PropFundOptimization
#  Determine Retained Claims
PropRetClaims  <- function(coeff){
  theta.vec     <- coeff[-1]
  CovTheta <- matrix(0, nrow = npol, ncol=nrisk )
  for (jrisk in 1:6){
    CovTheta[,jrisk] <- Coverage[,jrisk]*theta.vec[jrisk] }
  TCovTheta <- t(CovTheta)

  limParams <- array(0, dim=c(nrisk, nsim, npol))
  for (jsim in 1:nsim){
    limParams[,jsim,] <- TCovTheta }
  limClaims <- pmin(Prop.array,limParams )
  temp0.array <- aperm(limClaims, c(2, 1,3) )
  retainedClaims <- rowSums(temp0.array, dims=1)

  temp1.array <- aperm(Prop.array, c(2, 1,3) )
  sumClaims   <- rowSums(temp1.array, dims=1)
  ceededClaims <- sumClaims - retainedClaims
  return( cbind(retainedClaims, ceededClaims) )
}

# Evaluate the Objective function used in ES1 minimization
PropESKernmin.fct <- function(coeff, Reinsure){
  VaR.z0        <- coeff[1]
  if (Reinsure == FALSE){ RetClaims  <- PropRetClaims(coeff)[,1]
                 } else { RetClaims  <- PropRetClaims(coeff)[,2] }
  ESKern1.fct <-
    function(ytidle) { as.numeric( mean(
```

```
                        pmax(RetClaims + bw * ytidle - VaR.z0, 0) ) ) *
                        dnorm(ytidle)  }
  ESKernmin     <- VaR.z0 + integrate(Vectorize(ESKern1.fct),
                                      lower = -6, upper = 6)$value /
                                        (1-alpha)
  return( ESKernmin )
}

#Function to Summarize Results
PropESSummary.fct <- function(coeff, Reinsure){
  VaR.z0     <- coeff[1]
  theta.vec  <- coeff[-1]
  p.num      <- length(theta.vec)
  ones       <- rep(1, nsim )
  ClaimsPair <- PropRetClaims(coeff)
  RTC.R      <- mean( ClaimsPair[,2] )
 if (Reinsure == FALSE){
     RetClaims <- ClaimsPair[,1]
     }  else {
     RetClaims <- ClaimsPair[,2]
     }
  stddev    <- sd(RetClaims)
  VaR       <- as.numeric(quantile(RetClaims, prob = alpha, na.rm = TRUE))
  Kern.df   <- function(y){ yarg <- (y*ones - RetClaims)/bw
                            return(mean(pnorm(yarg)) - alpha)}
  Kern.VaR  <- uniroot(f=Kern.df,lower = VaR - 100,
                                 upper = VaR + 100)$root
  ESmin     <- as.numeric(VaR.z0 +
                          mean(pmax(RetClaims - VaR.z0, 0)) /(1-alpha) )
  ES        <- as.numeric(VaR   +
                          mean(pmax(RetClaims - VaR,   0)) /(1-alpha) )
  ESKern1.fct <-
    function(y) { as.numeric ( mean( pmax(
                   RetClaims -  VaR.z0 +  bw * y, 0) ) ) * dnorm(y) }
  ESKernmin  <- VaR.z0 +
                   integrate(Vectorize(ESKern1.fct),
                             lower = -6, upper = 6)$value / (1-alpha)
  ESKern2.fct <-
    function(y) { as.numeric ( mean( pmax(
                   RetClaims - Kern.VaR + bw * y, 0) ) ) * dnorm(y) }
  ESKern     <- Kern.VaR + integrate(Vectorize(ESKern2.fct), lower = -6,
                                     upper = 6)$value / (1-alpha)
  return( c(RTC.R, VaR, Kern.VaR, ESmin, ES,  ESKernmin, ESKern, stddev) )
}

# Gradient Function
Propf0z.fct <- function(coeff, Reinsure){
  VaR.z0        <- coeff[1]
  theta.vec     <- coeff[-1]
  if (Reinsure ==  FALSE){ RetClaims  <- PropRetClaims(coeff)[,1]
                 } else { RetClaims  <- PropRetClaims(coeff)[,2] }
```

```
  ones          <- rep(1, nsim)
  VaR.z0arg     <- (VaR.z0*ones  - RetClaims) / bw
  partial.z0    <- 1 - mean( 1-pnorm(VaR.z0arg) ) / (1 - alpha)

  CovTheta <- matrix(0, nrow = npol, ncol=nrisk )
  for (jrisk in 1:6){
    CovTheta[,jrisk] <- Coverage[,jrisk]*theta.vec[jrisk] }
  TCovTheta   <- t(CovTheta)
  TCoverage   <- t(Coverage)

  limParams   <- array(0, dim=c(nrisk, nsim, npol)) -> Exp.array
  for (jsim in 1:nsim){
    limParams[,jsim,] <- TCovTheta
    Exp.array[,jsim,] <- TCoverage   }
  temp.array  <- Exp.array * (Prop.array < limParams )
  #temp1.array <- aperm(temp.array, c(1, 3, 2) )
  mat.check   <- t( rowSums(temp.array, dims=2) )
  temp        <- ( 1-pnorm(VaR.z0arg) ) %*% mat.check / nsim
  partial.theta <- temp / (1 - alpha)
  return( c(partial.z0, partial.theta) )
  }  # end Propf0z.fct function

# Function to Construct a Frontier of Optimal Portfolios
#  p.z is the number of coefficients (including VaR)
PropThetaOptim.fct <-
  function(RTCmax.vec=PropRTCmax.vec, starter.coeff) {
    Output.mat <- matrix(0,nrow = length(RTCmax.vec), ncol = 1+6+p.z )
    RTC.optim <- function(coeff){ mean(PropRetClaims(coeff)[,2]) }
    # Use RTCR.fct for empirical RTCs, RTC.fct for lognormal

    time1 <- Sys.time()

    for (jCost in 1:length(RTCmax.vec)){         # Start jCost Loop
       # time1 <- Sys.time()
      hin.ES <- function(coeff) {h <- NA         # Constraints in alabama
        h[1] <- RTCmax.vec[jCost] - RTC.optim(coeff)
        for (j in 1:p.z){h[j+1] <- 1e6*coeff[j] }
        return(h)
        }
    if (jCost > 1 ){starter.coeff <- ES.opt$par*1.1}
      z.params <- rep(0, 7); LME.EX <- 0
      tryCatch({
        ES.opt <- alabama::auglag( par=starter.coeff,
           fn = PropESKernmin.fct,          # objective function
           gr = Propf0z.fct,        # gradient objective function
           hin = hin.ES,        # constraints
           control.outer=list(method="nlminb",trace=FALSE) )
        z.params <- ES.opt$par
        LME.EX <- ES.opt$lambda[1] },
        error=function(err) {
        z.params <- starter.coeff
```

```
        LME.EX <- 0} )
      output <- c(RTCmax.vec[jCost],
                PropESSummary.fct(z.params)[-c(4,6)],
                z.params[-1] , LME.EX)
       cat("Theory Results",output,"time take",
                           Sys.time() - time1, "\n")
      Output.mat[jCost,] <- output

      }  # end jCost Loop

    Sys.time() - time1
          #  Time difference of 19.18719 hours
    u.names <- paste("u",1:6, sep="")
    colnames(Output.mat) <- c("RTCmax", "RTC", "VaR", "Kern.Var",
                          "ES", "ESKern", "StdDev", u.names, "LME")
    save(Output.mat,
         file="../Data/WiscPropFundData/OutMat10KFeb2024.Rdata")
      return(Output.mat)
    }  # end PropThetaOptim.fct function
```

Construct Optimal Frontier

```
# PropertyFundSetUp
load(file="../Data/WiscPropFundData/dataout.Rdata")
# Coverage is in millions, not in logs
Coverage <- cbind(dataout$CoverageBC, dataout$CoverageIM,
                  dataout$CoverageCN, dataout$CoverageCO,
                  dataout$CoveragePN, dataout$CoveragePO  )
colnames(Coverage) <- c("BC", "IM", "CN", "CO", "PN", "PO")
rm(dataout)

bw    <- 1
alpha <- 0.80

#save(Prop.array,
#     file = "../Data/WiscPropFundData/SimPropArray.RData")
load(file = "../Data/WiscPropFundData/SimPropArray.RData")

nsim  <- length(Prop.array[1,,1])
npol  <- length(Prop.array[1,1,])
nrisk <- length(Prop.array[,1,1])

SummaryOut <-cbind(rowSums(Prop.array[1,,]), rowSums(Prop.array[2,,]),
                   rowSums(Prop.array[3,,]), rowSums(Prop.array[4,,]),
                   rowSums(Prop.array[5,,]), rowSums(Prop.array[6,,]) )
dfSummaryOut <- data.frame(SummaryOut)

###################################
#temp1.array <- aperm(Prop.array, c(1, 3,2) )
```

```
#TotCost   <- sum( rowSums(temp1.array, dims=2) )/ nsim
TotCost   <- sum(colMeans(dfSummaryOut))
CstLev <- 1 - seq(0.05, 0.95, by = 0.10)
CstLev <- round(CstLev*100)/100
PropRTCmax.vec<- TotCost * CstLev

p.z = 7
# Function to Construct a Frontier of Optimal Portfolios
#  p.z is the number of coefficients (including VaR)

Reinsure <- FALSE
#Reinsure <- TRUE

starter.coeff     <- c( 1500, 30, rep(50,5) )

# PropThetaOptim.fct <- function(RTCmax.vec=PropRTCmax.vec,
#                                starter.coeff, Reinsure = FALSE) {
  RTCmax.vec <- PropRTCmax.vec
  Output.mat <- matrix(0,nrow = length(RTCmax.vec), ncol = 1 + 6 + p.z )
  f0         <- function(coeff){PropESKernmin.fct(coeff, Reinsure)}
  f0grad     <- function(coeff){Propf0z.fct(coeff, Reinsure)}
  RTC.optim  <- function(coeff){  mean(PropRetClaims(coeff)[,2]) }

  time1 <- Sys.time()

  for (jCost in 1:length(RTCmax.vec)){         # Start jCost Loop
    hin.ES <- function(coeff) {h <- NA         # Constraints in alabama

      if (Reinsure == FALSE){ h[1] <- RTCmax.vec[jCost] - RTC.optim(coeff)
#    }  else { h[1] <- RTC.optim(coeff) - RTCmax.vec[jCost] }
    }  else { h[1] <- RTC.optim(coeff) - RTCmax.vec[length(RTCmax.vec)-jCost+1] }
      for (j in 1:p.z){h[j+1] <- 1e6*coeff[j] }
      return(h)
      }
  if (jCost > 1 ){starter.coeff <- ES.opt$par*0.9}
    z.params <- rep(0, 7); LME.EX <- 0
    tryCatch({
        ES.opt <- alabama::auglag( par=starter.coeff,
           fn = f0,              # objective function
           #gr = f0grad,         # gradient objective function
           hin = hin.ES,         # constraints
           control.outer=list(method="nlminb",trace=FALSE) )
        z.params <- ES.opt$par
        LME.EX <- ES.opt$lambda[1] },
        error=function(err) {
        z.params <- starter.coeff
        LME.EX <- 0} )
    #z.params[is.na(z.params)] <- 0
    output <- c(RTCmax.vec[length(RTCmax.vec)-jCost+1],
                PropESSummary.fct(z.params, Reinsure)[-c(4,6)],
                z.params[-1] , LME.EX)
```

```
    #output[is.na(output)] <- 0
    cat("Theory Results",output,"time taken", Sys.time() - time1, "\n")
      Output.mat[jCost,] <- output
    }  # end jCost Loop

  Sys.time() - time1

            #  Time difference of 19.18719 hours
  u.names <- paste("u",1:6, sep="")
  colnames(Output.mat) <- c("RTCmax", "RTC", "VaR", "Kern.Var",
                            "ES", "ESKern", "StdDev",
                            u.names, "LME")
 save(Output.mat, file="../Data/WiscPropFundData/OutMatReins10KFeb2024.Rdata")

    # return(Output.mat)
    # }  # end PropThetaOptim.fct function

  coeff <- c(0, output[8:13])
```

Figure 8.3

```
# save(Output.mat, file="../Data/WiscPropFundData/OutMat10KFeb2024.Rdata")
load(file="Data/WiscPropFundData/OutMat10KFeb2024.Rdata")

rndfac <- 50
dfPlot <- data.frame(Output.mat)
p1 <- ggplot(dfPlot, aes(x=RTC, y=VaR)) + geom_point() + theme_bw() +
          scale_x_continuous(breaks = seq(from = rndfac*round(min(dfPlot$RTC)/rndfac),
                                          to = rndfac*round(max(dfPlot$RTC)/rndfac),
                                          length.out = 3))+geom_line()
p2 <- ggplot(dfPlot, aes(x=RTC, y=ES)) + geom_point() + theme_bw() +
          scale_x_continuous(breaks = seq(from = rndfac*round(min(dfPlot$RTC)/rndfac),
                                          to = rndfac*round(max(dfPlot$RTC)/rndfac),
                                          length.out = 3))+geom_line()  + labs(y = 'ES')
p3 <- ggplot(dfPlot, aes(x=RTC, y=StdDev)) + geom_point() + theme_bw() +
          scale_x_continuous(breaks = seq(from = rndfac*round(min(dfPlot$RTC)/rndfac),
                                          to = rndfac*round(max(dfPlot$RTC)/rndfac),
                                          length.out = 3))+geom_line()
grid.arrange(p1,p2,p3,nrow=1)
```

15.8.3 Section 8.1.4. Reinsurer ES Optimization

```
# Section814Opt
#save(simBC1,simIM1,simPN1,simPO1,simCN1,simCO1,UUMat,
#     file = "../Data/WiscPropFundData/SimPropClaims10K.RData")
load(file = "../Data/WiscPropFundData/SimPropClaims10K.RData")
```

```
rm(UUMat)
# A Few Functions
nsim <- 100
EventLimit <- function(uLimit, CovVar, ClaimInput){
  limClaims <- ClaimInput
  for (jcol in 1:ncol(limClaims)){limClaims[,jcol] <-
                               pmin(uLimit*CovVar[jcol],limClaims[,jcol])}
  sumOut <- rowSums(limClaims)
  return(sumOut)
}

LimitClaims <- function(u.vec,indices){
  limClaims <- matrix(0, nrow = nsim, ncol = 6)
  limClaims[,1] <- EventLimit(uLimit=u.vec[1], CovVar=Coverage[,1],
                              ClaimInput=simBC1[indices,]/1000)
  limClaims[,2] <- EventLimit(uLimit=u.vec[2], CovVar=Coverage[,2],
                              ClaimInput=simIM1[indices,]/1000)
  limClaims[,3] <- EventLimit(uLimit=u.vec[3], CovVar=Coverage[,3],
                              ClaimInput=simCN1[indices,]/1000)
  limClaims[,4] <- EventLimit(uLimit=u.vec[4], CovVar=Coverage[,4],
                              ClaimInput=simCO1[indices,]/1000)
  limClaims[,5] <- EventLimit(uLimit=u.vec[5], CovVar=Coverage[,5],
                              ClaimInput=simPN1[indices,]/1000)
  limClaims[,6] <- EventLimit(uLimit=u.vec[6], CovVar=Coverage[,6],
                              ClaimInput=simPO1[indices,]/1000)
  return(limClaims)
}

UnLimClaims <- LimitClaims(rep(1e10, 6),1:nsim)
# summary(UnLimClaims)

alpha = 0.95
QuanRMReins.fct <- function(coeff){
  VaR.x   <- coeff[1]
  u.vec   <- coeff[-1]
  LimClaims <- LimitClaims(u.vec,1:nsim)
  temp    <- colMeans(UnLimClaims) - colMeans(LimClaims)
  RTC.R   <- sum(temp)
  AggULY  <- rowSums(UnLimClaims)
  AggY    <- rowSums(LimClaims)
  VaR     <- as.numeric(quantile(AggULY-AggY, prob = alpha))
  CTEmin  <- as.numeric(VaR.x +
                        ( mean(pmax(AggULY-AggY - VaR.x, 0)) )/(1-alpha) )
  CTE     <- as.numeric(VaR   +
                        ( mean(pmax(AggULY-AggY- VaR,   0)) )/(1-alpha) )
  stddev  <- sd(AggULY-AggY)
  return(c(RTC.R, VaR, CTEmin, CTE, stddev))
}

time1 <- Sys.time()
numVars = 6
```

```
TotCost   <- sum(colMeans(UnLimClaims))
CstLev <- 1 - seq(0.05, 0.95, by = 0.10)
CstLev <- round(CstLev*100)/100
RTCmax.vec<- TotCost * CstLev

#starter.coeff     <-rep(6e-2,7)
starter.coeff     <- c( 25000, 5, rep(500,5) )

OutMatReins.CTE <- matrix(0,nrow = length(RTCmax.vec), ncol = numVars+5)
u.names <- paste("u",1:numVars, sep="")
colnames(OutMatReins.CTE) <- c("RTCmax", "RTC", "VaR", "ES", "StdDev", u.names)

#for (jCost in 1:length(RTCmax.vec)){
for (jCost in 1:3){
  f0Reins.CTE <- function(coeff){QuanRMReins.fct(coeff)[3]}
  RTCReins.CTE <- function(coeff){QuanRMReins.fct(coeff)[1]}
  hinReins.CTE <- function(coeff) {h <- NA
#    h[1] <- -( RTCmax.vec[jCost] - RTCReins.CTE(coeff) )
    h[1] <- -( RTCmax.vec[length(RTCmax.vec)-jCost+1] - RTCReins.CTE(coeff) )
    for (j in 1:(1+numVars)){h[j+1] <- 1e6*coeff[j] }
    #h[8] <- 10 - coeff[2]
    return(h)
    }
  if (jCost > 1 ){starter.coeff[2] <- CTEReins.opt$par[2]}
  CTEReins.opt <- alabama::auglag(par=starter.coeff,
           fn=f0Reins.CTE,
           hin=hinReins.CTE,
           control.outer=list(method="nlminb",trace=FALSE))
    uparams <- CTEReins.opt$par[-1]
    output <- c(RTCmax.vec[jCost], QuanRMReins.fct(CTEReins.opt$par)[-3], uparams)
  cat("alabama results",output, "time taken",Sys.time() - time1,"\n")
    OutMatReins.CTE[jCost,] <- output
    Sys.time() - time1
}

Sys.time() - time1

# > Sys.time() - time1
# Time difference of 8.854467 hours

#save(OutMatReins.CTE,
#     file="../Data/WiscPropFundData/OutMatReins10KReverse.Rdata")

# > Sys.time() - time1
# Time difference of 1.217191 days
```

15.8.4 Section 8.2.1. Including Property Risk

```
# ANUPropertySetUp
# Bring in ANU Claims Data
load(file = "../Data/SimData/AggClaimsMay100Kw0.7.Rdata")
#  Fixing Property Background Risk at a Specific Upper Limit
#Xsim    <- Claims
uPropFixed    <- 5000
TotCostANU    <- mean (Claims[,-1])
CstLevNum     <- 1 - c(0.05, seq(0.1, 0.9, length.out=9),0.95)
CstLevNum     <- round(CstLevNum*100)/100
ANURTCmax.vec <- TotCostANU * CstLevNum

#  A Few Key Parameters
alpha         <- 0.80
bw            <- 1
starter.coeff <- c(5000, rep(1, 14) )

# Function to Compute Retained Claims
ANUPropRetClaims.fct <- function(coeff, Xsim, ExLoss = TRUE){
  uPropFixed <- 5000
  Xsim[,1]  <- pmin(Xsim[,1], uPropFixed)
  theta.vec <- coeff[-1]
  p.num     <- length(theta.vec)
  RetClaims <- Xsim[,1]
  if (ExLoss){  # Excess of Loss
     for (kidx in 1:p.num){RetClaims <- RetClaims +
                     pmin(Xsim[,1+kidx], theta.vec[kidx]) }
     } else {  # Quota Share
            RetClaims  <- as.vector(Xsim[,-1] %*% theta.vec)   }
  return( RetClaims )
}  # end ANUPropRetClaims.fct function

#Function to Compute Simulated RTC
ANUPropRTCR.fct <- function(Xsim, RetClaims){
  PropLim  <- pmin(Xsim[,1], uPropFixed = 5000)
  CeededClaimsNotProp <- rowSums(Xsim[,-1]) - (RetClaims - PropLim)
  RTC.R     <- mean( CeededClaimsNotProp )
  return( RTC.R )
}  # end ANUPropRTCR.fct function

# Function to Compute Gradient of the Objective Function
ANUPropf0z.fct <- function(coeff, Xsim, RetClaims, ExLoss = TRUE){
  VaR.z0        <- coeff[1]
  theta.vec     <- coeff[-1]
  numLoss       <- nrow(Xsim)
  ones          <- rep(1, numLoss )
  VaR.z0arg     <- (VaR.z0*ones - RetClaims) / bw
  partial.z0    <- 1 - mean( 1-pnorm(VaR.z0arg) ) / (1 - alpha)

  if (ExLoss){  # Excess of Loss
```

```
      mat.check <-  t( 1* ( t(Xsim[,-1]) > theta.vec) )
    } else {  # Quota Share
          mat.check <- Xsim
          }
  temp          <- ( 1-pnorm(VaR.z0arg) ) %*% mat.check / numLoss
  partial.theta <- temp / (1 - alpha)
  return( c(partial.z0, partial.theta) )
}  # end ANUPropf0z.fct function
```

```
# OptimANUConProp
# Temporary - re-assign these functions
RetClaims.fct <- ANUPropRetClaims.fct
RTCR.fct      <- ANUPropRTCR.fct
f0z.fct       <- ANUPropf0z.fct

Time1 <- Sys.time()
OutANUProp.ES <- ThetaOptim.fct(Xsim=Claims, ExLoss = TRUE, p.z = 15,
                                RTCmax.vec=ANURTCmax.vec, starter.coeff)
Sys.time() - Time1   #59.71116 mins for alpha = 0.80

u.names <- paste("u",1:14, sep=".")
colnames(OutANUProp.ES) <- c("RTCmax", "RTC", "VaR", "Kern.VaR", "ESmin",
                             "ES", "StdDev",  u.names ,"LME")
save(ANURTCmax.vec, OutANUProp.ES,
     file= "../ChapTablesData/Chap8/OutANUProp.ESMar2024.Rdata")
```

15.8.5 Section 8.2.2. Varying Alpha

```
# Section822VaryAlpha
# save(ANURTCmax.vec, OutANUProp.ES,
#      file= "../ChapTablesData/Chap8/OutANUProp.ESMar2024.Rdata")
load(file= "../ChapTablesData/Chap8/OutANUProp.ESMar2024.Rdata")

CompRow       <- 6
#starter.Alpha <- c(OutANUProp.ES[CompRow,4], OutANUProp.ES[CompRow,8:21] )
#Alphavec     <- c( 0.5, 0.6, 0.80, 0.95, 0.98)
starter.Alpha <- c( 5000, rep(100,14) )
Alphavec      <- c(seq(from=0.5, to=0.95, length.out=10),0.98)
OutMat.Alpha  <- matrix(0,nrow = length(Alphavec), ncol = 9+14)
u.names       <- paste("u",2:15, sep="")
colnames(OutMat.Alpha) <- c("Alpha","RTCmax", "RTC", "VaR", "Kern.VaR",
                            "ES", "ESKern", "StdDev", u.names, "LME.EX")

# Temporary - re-assign these functions
RetClaims.fct <- ANUPropRetClaims.fct
RTCR.fct      <- ANUPropRTCR.fct
f0z.fct       <- ANUPropf0z.fct
```

```
Time1 <- Sys.time()
for (jCost in 1:length(Alphavec)){
  alpha <- Alphavec[jCost] ->  OutMat.Alpha[jCost,1]

  OutMat.Alpha[jCost,2:23]  <-
     as.vector(ThetaOptim.fct(Xsim=ClaimsClaims[1:20000,], ExLoss = TRUE, p.z = 15,
                              RTCmax.vec=ANURTCmax.vec[CompRow], starter.Alpha) )
  }
Sys.time() - Time1   #18.75775 mins for sim num = 20000

save(OutMat.Alpha,
     file= "../ChapTablesData/Chap8/OutMat.AlphaMar2024.Rdata")
```

15.8.6 Section 8.2.3. Range Value at Risk

```
# Section823VaryRVaR
# save(ANURTCmax.vec, OutANUProp.ES,
#      file= "../ChapTablesData/Chap8/OutANUProp.ESMar2024.Rdata")
load(file= "../ChapTablesData/Chap8/OutANUProp.ESMar2024.Rdata")

CompRow       <- 6
starter.Alpha <- c( 5000, rep(100,14) )
Alphavec      <- c(seq(from=0.5, to=0.95, length.out=10),0.98)
OutMat.Alpha  <- matrix(0,nrow = length(Alphavec), ncol = 9+14)
u.names       <- paste("u",2:15, sep="")
colnames(OutMat.Alpha) <- c("Alpha","RTCmax", "RTC", "VaR", "Kern.VaR",
                            "ES", "ESKern", "StdDev", u.names, "LME.EX")

# Temporary - re-assign these functions
RetClaims.fct <- ANUPropRetClaims.fct
RTCR.fct      <- ANUPropRTCR.fct
f0z.fct       <- ANUPropf0z.fct

Time1 <- Sys.time()
for (jCost in 1:length(Alphavec)){
  alpha <- Alphavec[jCost] ->  OutMat.Alpha[jCost,1]
  OutMat.Alpha[jCost,2:23]  <-
    as.vector(ThetaOptim.fct(Xsim=ClaimsClaims[1:20000,],
                             ExLoss = TRUE, p.z = 15,
                             RTCmax.vec=ANURTCmax.vec[CompRow],
                             starter.Alpha) )
  }
Sys.time() - Time1   #18.75775 mins for sim num = 20000

save(OutMat.Alpha,
     file= "../ChapTablesData/Chap8/OutMat.AlphaMar2024.Rdata")
```

15.8.7 Section 8.2.4. Claim Level Retention

```
load(file = "SimDataMay/IndClaimsMay100Kw0.7.Rdata")
load(file = "SimDataMay/AggClaimsMay100Kw0.7.Rdata")

uPropFixed2 <- 1000
nsimEvnt    <- 20000
ClaimGPA    <- ClaimGPA[1:nsimEvnt,]
ClaimTravel <- ClaimTravel[1:nsimEvnt,]
ClaimMV     <- ClaimMV[1:nsimEvnt,]
Claims      <- Claims[1:nsimEvnt,]

EventLimit <- function(uLimit,ClaimInput){
  ClaimTemp<- matrix(pmin(uLimit,ClaimInput),
                     nrow=nrow(ClaimInput),ncol=ncol(ClaimInput))
  ClaimSumOuput  <- rowSums(ClaimTemp)
  return(ClaimSumOuput)
}

LimitClaims <- function(u.vec){
  ClaimsLim <- Claims
  ClaimsLim[,7] <- EventLimit(uLimit=u.vec[7],ClaimInput=ClaimGPA)
  ClaimsLim[,8] <- EventLimit(uLimit=u.vec[8],ClaimInput=ClaimTravel)
  ClaimsLim[,13] <- EventLimit(uLimit=u.vec[13],ClaimInput=ClaimMV)
  TweedieClaims <- c(1:6, 9:12, 14, 15)
  for (jVar in TweedieClaims){
    ClaimsLim[,jVar] = pmin(Claims[,jVar],u.vec[jVar])
  }
  return(ClaimsLim)
  }

UnLimClaims <- LimitClaims(rep(1e30, 15))

QuanRMEventLim.fct <- function(coeff){
  VaR.x   <- coeff[1]
#  u.vec  <- c(uPropFixed, coeff[-1])  #property is fixed
  u.vec   <- c(uPropFixed2, coeff[-1])  #property is fixed
  LimClaims <- LimitClaims(u.vec)
  temp    <- colMeans(UnLimClaims) - colMeans(LimClaims)
  RTC.R   <- sum(temp[-1])
  AggY    <- rowSums(LimClaims)
  VaR     <- as.numeric(quantile(AggY, prob = alpha))
  CTEmin  <- as.numeric(VaR.x + ( mean(pmax(AggY - VaR.x, 0)) )/(1-alpha) )
  CTE     <- as.numeric(VaR   + ( mean(pmax(AggY- VaR,   0)) )/(1-alpha) )
  stddev  <- sd(AggY)
  return(c(RTC.R, VaR, CTEmin, CTE, stddev))
}

time1   <- Sys.time()
numVars <- 14
```

```
TotCost       <- sum(ANU.mu[-1])
CstLevNumEvnt <- 1 - seq(0.05, 0.95, by = 0.05)
CstLevNumEvnt <- round(CstLevNumEvnt*100)/100
RTCmax.vecEvnt<- TotCost * CstLevNumEvnt

load(file = "Data/OutMatProp.CTE.Rdata")
starter.u         <- as.numeric(OutMatProp.CTE[6, 6:19])
VaR.initial       <- as.numeric(OutMatProp.CTE[6, 3])
starter.coeff     <- c(VaR.initial,starter.u)*.75
#  Starting Values for Claim Level  Coverages
starter.coeff[7]  <- quantile(ClaimGPA[,1], prob = 0.5)
starter.coeff[8]  <- quantile(ClaimTravel[,1], prob = 0.5)
starter.coeff[13] <- quantile(ClaimMV[,1], prob = 0.5)

OutMatEventLim.CTE <- matrix(0,nrow = length(RTCmax.vecEvnt),
                             ncol = numVars+5)
u.names <- paste("u",2:(1+numVars), sep="")
colnames(OutMatEventLim.CTE) <- c("RTCmax", "RTC", "VaR",
                                  "ES", "StdDev", u.names)

for (jCost in 1:length(RTCmax.vecEvnt)){
  f0EventLim.CTE <- function(coeff){QuanRMEventLim.fct(coeff)[3]}
  RTCEventLim.CTE <- function(coeff){QuanRMEventLim.fct(coeff)[1]}
  hinEventLim.CTE <- function(coeff) {h <- NA
    h[1] <- RTCmax.vecEvnt[jCost] - RTCEventLim.CTE(coeff)
    for (j in 1:numVars){h[j+1] <- coeff[j] }
    return(h)
    }
  if (jCost > 1 ){starter.coeff <- CTEEventLim.opt$par}
  CTEEventLim.opt <- alabama::auglag(par=starter.coeff,
           fn=f0EventLim.CTE,
           hin=hinEventLim.CTE,
           control.outer=list(method="nlminb",trace=FALSE))
    uparams <- CTEEventLim.opt$par[-1]
    output <- c(RTCmax.vecEvnt[jCost],
                QuanRMEventLim.fct(CTEEventLim.opt$par)[-3],
                uparams)
  cat("alabama results",output, "\n")
    OutMatEventLim.CTE[jCost,] <- output
    Sys.time() - time1
}

save(OutMatEventLim.CTE,
     file="../Data/ANUCaseData/OutMatEventLim20KProp.Rdata")

Sys.time() - time1
# Time difference of 9.82370451 hours
```

15.8.8 Appendix. Generate Property Fund Data

```
#  Chap8PropFundData
library(tweedie)
require(statmod)
library(BB)
library(MASS)
library(VineCopula)
library(copula)

load("../Data/WiscPropFundData/data.RData")
load("../Data/WiscPropFundData/dataout.RData")
source("../Data/WiscPropFundData/setup.R")
load("../Data/WiscPropFundData/TweedieEstimates.RData")
load("../Data/WiscPropFundData/TweedieDependenceParams.RData")

out1phi.max <- 165.814
out1xi.max  <- 1.669388
out2phi.max <- 849.5301
out2xi.max <- 1.461224
out3phi.max <- 376.1897
out3xi.max <- 1.418367
out4phi.max <- 322.6615
out4xi.max <- 1.508163
out5phi.max <- 336.2968
out5xi.max <- 1.467347
out6phi.max <- 302.5563
out6xi.max <- 1.526531

#######################################################
# Tweedie Outsample

fittedout <- matrix(10,nrow(dataout),6)
fittedout[BCyout,1] <- exp(predict(twBC,newdata=doutBC))
fittedout[IMyout,2] <- exp(predict(twIM,newdata=doutIM))
fittedout[PNyout,3] <- exp(predict(twPN,newdata=doutPN))
fittedout[POyout,4] <- exp(predict(twPO,newdata=doutPO))
fittedout[CNyout,5] <- exp(predict(twCN,newdata=doutCN))
fittedout[COyout,6] <- exp(predict(twCO,newdata=doutCO))
rownames(fittedout) <- dataout$PolicyNum
colnames(fittedout) <- c("BC","IM","PN","PO","CN","CO")
#head(fittedout)

time0 <- Sys.time()
nsim = 10000
nPol = nrow(dataout)
set.seed(12345)
simBC <- simIM <- simPN <- simPO <- simCN <- simCO <- matrix(0,nsim,nPol)

UUMat <- rCopula(nsim*nPol,normalCopula(P2p(pairmat),dim=6,dispstr="un"))
```

```
for (isim in 1:nsim) {
  simBC[isim,] <- qtweedie(UUMat[((isim-1)*nPol+1):(isim*nPol),1],
                           xi=out1xi.max, mu=fittedout[,1], phi=out1phi.max)
  simIM[isim,] <- qtweedie(UUMat[((isim-1)*nPol+1):(isim*nPol),2],
                           xi=out2xi.max, mu=fittedout[,2], phi=out2phi.max)
  simPN[isim,] <- qtweedie(UUMat[((isim-1)*nPol+1):(isim*nPol),3],
                           xi=out3xi.max, mu=fittedout[,3], phi=out3phi.max)
  simPO[isim,] <- qtweedie(UUMat[((isim-1)*nPol+1):(isim*nPol),4],
                           xi=out4xi.max, mu=fittedout[,4], phi=out4phi.max)
  simCN[isim,] <- qtweedie(UUMat[((isim-1)*nPol+1):(isim*nPol),5],
                           xi=out5xi.max, mu=fittedout[,5], phi=out5phi.max)
  simCO[isim,] <- qtweedie(UUMat[((isim-1)*nPol+1):(isim*nPol),6],
                           xi=out6xi.max, mu=fittedout[,6], phi=out6phi.max)
}

simBC1 <- simIM1 <- simPN1 <- simPO1 <- simCN1 <- simCO1 <- matrix(0,nsim,nPol)

simBC1[,BCyout] = simBC[,BCyout]
simIM1[,IMyout] = simIM[,IMyout]
simPN1[,PNyout] = simPN[,PNyout]
simPO1[,POyout] = simPO[,POyout]
simCN1[,CNyout] = simCN[,CNyout]
simCO1[,COyout] = simCO[,COyout]

# Save Simulations
save(simBC1,simIM1,simPN1,simPO1,simCN1,simCO1,UUMat,
     file = "../Data/WiscPropFundData/SimPropClaims10K.RData")

difftime(Sys.time(), time0, units='mins')

#simTotal <- rowSums(simBC1+simIM1+simPN1+simPO1+simCN1+simCO1)

Prop.array <- array(0, dim = c(6, nrow(simBC1), ncol(simBC1)))
Prop.array[1,,] <- simBC1
Prop.array[2,,] <- simIM1
Prop.array[3,,] <- simCN1
Prop.array[4,,] <- simCO1
Prop.array[5,,] <- simPN1
Prop.array[6,,] <- simPO1
Prop.array <- Prop.array / 1000
save(Prop.array, file = "../Data/WiscPropFundData/SimPropArray.RData")
```

15.8.9 Appendix Section 8.3.5. ANU Risk Distribution

Generate Summary Statistics from Inputs

```
#  ANURiskDistn
#  Bring in Risk Parameters
```

```
RiskParameters<- read.csv("../Data/ANURiskParametersInput.csv",header=T)

riskTypes <- RiskParameters$Class.of.Insurance
riskTypesShort <- RiskParameters$Class

ANU.lambda <- RiskParameters$ExpNum
ANU.Prem   <- RiskParameters$Premium / 1000  #thousands of AUD
ANU.mu     <- rep(NA,length(ANU.lambda))
ANU.deduct <- RiskParameters$Deductible / 1000  #thousands of AUD
ANU.deduct1 <- ANU.deduct
ANU.deduct1[ANU.deduct1<1] <-1
ANU.deduct10 <- ANU.deduct
ANU.deduct10[ANU.deduct10<10] <-10
power      <- 1.67
power.vec <- c(1.9, 1.9, rep(1.67,13))

TweedieClaims <- c(1:6, 9:12, 14, 15)
NotTweedieClaims <- c(7, 8, 13)

for (jPrem in TweedieClaims){

  Delta.Prem <- function(mu.fct){
         power <- power.vec[jPrem]
         phi.fct = mu.fct**(2-power)/(ANU.lambda[jPrem]*(2-power))
         sfF <- function(x){1-tweedie::ptweedie(x, power=power,
                                                mu=mu.fct, phi=phi.fct)}
         prem <- mu.fct - integrate(sfF, lower = 0,
                                    upper = ANU.deduct[jPrem])$value
         return(ANU.Prem[jPrem] - prem)}
    ANU.mu[jPrem] <- uniroot(Delta.Prem, lower = 2, upper = 4000000)$root
}

ANU.phi <- ANU.mu**(2-power.vec)/(ANU.lambda*(2-power.vec))
ANU.stdDev <- sqrt(ANU.phi)*ANU.mu**(power.vec/2)

# For GPA (jLine=7), Travel (jLine=8), and Motor Vehicle(jLine=13),
          #use Lognormal Distribution
#Group Personal Accident
ANU.lambda[7] = 20
paramGPA1     = 6.982327314 - log(1000)
paramGPA2     = 0.103236644
ANU.mu[7]     = ANU.lambda[7]*exp(paramGPA1 + paramGPA2/2)
ANU.stdDev[7] = ANU.mu[7]*sqrt( exp(paramGPA2)-1 )
#Travel
ANU.lambda[8] = 220
paramTrv1     = 6.3487941  - log(1000)
paramTrv2     = 0.220671512
ANU.mu[8]     = ANU.lambda[8]*exp(paramTrv1 + paramTrv2/2)
ANU.stdDev[8] = ANU.mu[8]*sqrt( exp(paramTrv2)-1 )
# Motor Vehicle
ANU.lambda[13]= 80
```

```
paramMV1     = 7.937941050 - log(1000)
paramMV2     = 0.414643912
ANU.mu[13]    = ANU.lambda[13]*exp(paramMV1 + paramMV2/2)
ANU.stdDev[13]= ANU.mu[13]*sqrt( exp(paramMV2)-1 )

ANU.quan95 <- 0*ANU.mu -> ANU.quan99
for (jPrem in TweedieClaims){
  ANU.quan95[jPrem] <-
    tweedie::qtweedie(p=0.95, power=power.vec[jPrem],
                      mu=ANU.mu[jPrem], phi=ANU.phi[jPrem])
  ANU.quan99[jPrem] <-
    tweedie::qtweedie(p=0.99, power=power.vec[jPrem],
                      mu=ANU.mu[jPrem], phi=ANU.phi[jPrem])
}
ANU.quan95[NotTweedieClaims] <-
    ANU.mu[NotTweedieClaims] +
       qnorm(.95)*ANU.stdDev[NotTweedieClaims]
ANU.quan99[NotTweedieClaims] <-
    ANU.mu[NotTweedieClaims] +
       qnorm(.99)*ANU.stdDev[NotTweedieClaims]
#  Quantiles for Individuals Claims
Ind.ANU.quan99 <- ANU.quan99
Ind.ANU.quan99[7] <- exp( paramGPA1 + qnorm(.99)*sqrt(paramGPA2) )
Ind.ANU.quan99[8] <- exp( paramTrv1 + qnorm(.99)*sqrt(paramTrv2) )
Ind.ANU.quan99[13] <- exp( paramMV1 + qnorm(.99)*sqrt(paramMV2) )

save(ANU.deduct, ANU.Prem, ANU.lambda, ANU.mu,  ANU.stdDev, ANU.quan95,
     ANU.quan99, Ind.ANU.quan99, riskTypes, riskTypesShort, ANU.phi,
     file = "../ChapTablesData/Chap7/PremTable.Rdata")
```

Table 8.14

```
# ANURiskSummary
# save(ANU.deduct, ANU.Prem, ANU.lambda, ANU.mu,  ANU.stdDev, ANU.quan95,
#      ANU.quan99, Ind.ANU.quan99, riskTypes, riskTypesShort, ANU.phi,
#      file = "../ChapTablesData/Chap7/PremTable.Rdata")

load(file = "ChapTablesData/Chap7/PremTable.Rdata")

Param.mat <- cbind(ANU.deduct, ANU.Prem, ANU.lambda, ANU.mu,
                   ANU.stdDev, ANU.quan95, ANU.quan99 )
Param.mat <- round(Param.mat, digits = 0)
Param.mat[,3] <- ANU.lambda
row.names(Param.mat) <- riskTypes
colnames(Param.mat) <- c("Deductible", "Premium", "Expected Number",
  "Mean",   "Standard Dev", "95th Percentile", "99th Percentile")

if (HtmlEval){
kableExtra::kbl(Param.mat, caption='**ANU Risk Summary Statistics**', align = 'rrrrrrrr',
```

```
            table.attr = "style='width:100%;'") %>%
  kableExtra::kable_classic(full_width = F,  html_font = "Cambria") %>%
  kable_styling(bootstrap_options = c("striped", "condensed"))
  } else
   {
  kableExtra::kbl(Param.mat, caption='ANU Risk Summary Statistics', align = 'rrrrrrrr',
            booktabs = T) %>%
       column_spec(1, width = "2.5cm") %>%
       column_spec(2:8, width = "1.2cm")  %>%
   kableExtra::kable_styling(latex_options = c("striped", "condensed"), font = 9)
     }
```

15.8.10 Appendix Section 8.3.6. Generate ANU Simulated Distributions

Remark. This code creates 11 distributions, one for each value of a mixing parameter that is described in Section 9.1.

```
# Chap8ANUdataGen

#  Bring in Risk Parameters
RiskParameters<- read.csv("../Data/ANURiskParametersInput.csv", header=T)
#  Bring in Risk Distribution work
# save(ANU.deduct, ANU.Prem, ANU.lambda, ANU.mu,  ANU.stdDev, ANU.quan95,
#      ANU.quan99, Ind.ANU.quan99, riskTypes, riskTypesShort, ANU.phi,
#      file = "../ChapTablesData/Chap7/PremTable.Rdata")
load(file = "../ChapTablesData/Chap7/PremTable.Rdata")

time1 <- Sys.time()

#  Set Simulation Parameters
nsim     <- 1000
set.seed(202020)
numRisks <- 15
wRV      <- runif(nsim)
UInd     <- matrix(runif(nsim*numRisks), nrow = nsim, ncol = numRisks)
UUpper   <- UInd[,1] %*% t(as.vector(rep(1, numRisks)))

wStar.vec <- seq(from=0, to=1, by= 0.1)
power.vec <- c(1.9, 1.9, rep(1.67,13))
TweedieClaims    <- c(1:6, 9:12, 14, 15)
NotTweedieClaims <- c(7, 8, 13)

genLogNormal <- function(LineNum, NumClaims.vec){  #  Start genLogNormal function
    sigsqLN  <- log( 1+ (ANU.stdDev[LineNum]/ ANU.mu[LineNum])**2 )
    muLN     <- log( ANU.mu[LineNum]/ANU.lambda[LineNum] ) - sigsqLN/2
    LNClaim  <- rnorm(n=nsim*max(NumClaims.vec), mean=muLN, sd=sqrt(sigsqLN) )
    ClaimOut <- matrix( exp(LNClaim), nrow = nsim, ncol = max(NumClaims.vec) )
    for (jCol in 1:ncol(ClaimOut)){
     ClaimOut[,jCol] <- ClaimOut[,jCol]*(jCol <= NumClaims.vec)
     }
```

```
    return(ClaimOut)
    }  #  End genLogNormal function

for (iw in 1:length(wStar.vec)){   # Start Mixing Parameter Loop
  wStar <- wStar.vec[iw]
  UCop <- (wRV <= wStar)* UInd + (wRV > wStar)* UUpper

  Claims <- matrix(0, nrow = nsim, ncol = numRisks)
  for (jVar in TweedieClaims){
   power <- power.vec[jVar]
    Claims[,jVar] = tweedie::qtweedie(UCop[,jVar], power=power,
                                      mu=ANU.mu[jVar], phi=ANU.phi[jVar])
    }

# Start with Claims Number
  NumClaims <- matrix(0, nrow = nsim, ncol = length(NotTweedieClaims))
  inum <- 0
  for (jVar in NotTweedieClaims){inum = inum +1
    NumClaims[,inum] <- qpois(p = UCop[,jVar], lambda = ANU.lambda[jVar])
    }

  ClaimGPA    <- genLogNormal(LineNum=7,  NumClaims.vec=NumClaims[,1])
  ClaimTravel <- genLogNormal(LineNum=8,  NumClaims.vec=NumClaims[,2])
  ClaimMV     <- genLogNormal(LineNum=13, NumClaims.vec=NumClaims[,3])

  Claims[,7]  <- rowSums(ClaimGPA)
  Claims[,8]  <- rowSums(ClaimTravel)
  Claims[,13] <- rowSums(ClaimMV)

  fileName <- paste("SimDataMay/AggClaimsMay10Kw",wStar,".Rdata",sep="" )
  save(Claims, file=fileName)

  if (iw == 8) {save(ClaimGPA, ClaimTravel, ClaimMV,
                     file = "SimDataMay/IndClaimsMay10Kw0.7.Rdata")}
  } # End Mixing Parameter Loop
```

15.9 Chapter Nine Code

15.9.1 Example 9.1. Effects of Dependence on Optimal an ANU Risk Portfolio

Parameter Set-Up

```
# Code to Bring in Risk Distribution Parameters

varSelect <- 2:15  # Choose = 2:15 to omit Property
numVars   <- length(varSelect)

#  Bring in Some Data
RiskParameters<- read.csv("Data/ANURiskParameters.csv",header=T)

riskTypes             <- RiskParameters$Class.of.Insurance
riskTypesShort        <- RiskParameters$Class
riskTypesSelect       <- riskTypes[varSelect]
riskTypesShortSelect  <- riskTypesShort[varSelect]

ANU.mu  <- RiskParameters$Premium / 1000  #thousands of AUD
TotCost <- sum(ANU.mu[-1])
```

Create Optimal Portfolios over a Range of Dependence

```
RMAggClaims.fct <- function(propCoeff=uPropFixed, AggClaims=Claims, coeff){
  VaR.x     <- coeff[1]
  u.vec     <- c(propCoeff, coeff[-1])  #property is fixed
  LimClaims <- AggClaims
  for (jVar in 1:ncol(LimClaims)){
    LimClaims[,jVar] = pmin(LimClaims[,jVar],u.vec[jVar])
  }
  temp  <- colMeans(AggClaims) - colMeans(LimClaims)
  RTC.R <- sum(temp[-1])        # do not include property
  AggY  <- rowSums(LimClaims)
  VaR   <- as.numeric(quantile(AggY, prob = alpha))
  CTEmin<- as.numeric(VaR.x + ( mean(pmax(AggY - VaR.x, 0)) )/(1-alpha) )
  CTE   <- as.numeric(VaR   + ( mean(pmax(AggY - VaR,   0)) )/(1-alpha) )
  stddev  <- sd(AggY)
  return(c(RTC.R, VaR, CTEmin, CTE, stddev))
}


# save(ANURTCmax.vec, OutANUProp.ES,
#      file= "../ChapTablesData/Chap8/OutANUProp.ESMar2024.Rdata")
load(file= "../ChapTablesData/Chap8/OutANUProp.ESMar2024.Rdata")
starter.coeff <- OutANUProp.ES[6,c(3,8:21)]
RTCmax        <- ANURTCmax.vec[6]
```

```
set.seed(123)
alpha        <- 0.80
nsim         <- 10000
uPropFixed   <- 5000

f0Depend.ES  <- function(coeff){RMAggClaims.fct(propCoeff=uPropFixed,
                                                AggClaims=Claims, coeff)[3]}
RTCDepend.ES <- function(coeff){RMAggClaims.fct(propCoeff=uPropFixed,
                                                AggClaims=Claims, coeff)[1]}

time1 <- Sys.time()
readData <- function(wStar,nsim=nsim){
  fileName <- paste("../Data/SimDataMay/AggClaimsMay100Kw",wStar,".Rdata",sep="" )
  load(file = fileName)
  AggClaims <- Claims[1:nsim,]
  return(AggClaims)
  }

wStar.vec <-  seq(from=0.0, to=1, length.out=11)
RTCmaxFix <- TotCost * 0.5

OutMat.Depend  <- matrix(0,nrow = length(wStar.vec), ncol = 15 + 5)
u.names <- paste("$u_{",2:(1+14), "}$", sep="")
colnames(OutMat.Depend) <- c("$w$", "LME1" , "$RTC$", "$VaR$", "$ES$", "Std Dev", u.names)

starterDir <- OutANUProp.ES[6,c(3,8:21)] * 0.8


hinDepend.ES <- function(coeff) {h <- NA
    h[1] <- (RTCmax - RTCDepend.ES(coeff))*1e6
    for (j in 1:numVars){h[j+1] <- 1e6 * coeff[j]}
    return(h)}

for (jCost in 1:length(wStar.vec)){
  Claims <- readData(wStar = wStar.vec[jCost],nsim=nsim)
  Depend.opt <- alabama::auglag(par=starterDir,
           fn=f0Depend.ES,
           hin=hinDepend.ES,
           control.outer=list(method="nlminb",trace=FALSE))
  uparams <- Depend.opt$par[-1]
  output <- c(1-wStar.vec[jCost], Depend.opt$lambda[1],
              RMAggClaims.fct(propCoeff=uPropFixed, AggClaims=Claims,
                              Depend.opt$par)[-c(3)], uparams)
  cat("alabama results",output, "\n")
  OutMat.Depend[jCost,] <- output
}

save(OutMat.Depend,
     file= "../ChapTablesData/Chap9/OutMatDepend.Mar2024.Rdata")

Sys.time() - time1 # 5.901147 mins for 10000 nsim
```

Table 9.1

```
# save(OutMat.Depend,
#      file= "../ChapTablesData/Chap9/OutMatDepend.Mar2024.Rdata")
load(file= "ChapTablesData/Chap9/OutMatDepend.Mar2024.Rdata")

OutMatPrint <- OutMat.Depend[,1:(6+3)]
OutMatPrint[,3:9] <- round(OutMatPrint[,3:9], digits = 0)
colnames(OutMatPrint) <- c("$w$", "LME1", "$RTC$", "$VaR$", "$ES$",
                           "Std Dev", "$u_2$", "$u_3$", "$u_4$")

TableGen1(TableData=OutMatPrint[,-2],
         TextTitle='ANU Excess of Loss Optimization - Varying Dependency',
         Align='r', Digits=2, ColumnSpec=1:6, ColWidth0= "1.5cm",
         BorderRight= c(2,5) , ColWidth  = ColWidth6 )
```

Figure 9.1

```
dfPlot <- data.frame(OutMat.Depend)

p1 <- ggplot2::ggplot(dfPlot, aes(x=w, y=VaR)) + geom_point() + theme_bw()+ geom_line()
p2 <- ggplot2::ggplot(dfPlot, aes(x=w, y=CTE)) + geom_point() + theme_bw()+
               geom_line()  + labs(y = 'ES')
p3 <- ggplot2::ggplot(dfPlot, aes(x=w, y=StdDev)) + geom_point() + theme_bw()+ geom_line()

grid.arrange(p1,p2,p3,nrow=1)
```

Figure 9.2

```
#  Define a Function to Plot for Each Allocation
rndfac <- 20
bdd <- 2
pltfunc <- function(num,varx,vary,labelx){
     limy1 <- max(rndfac*(round(min(vary)/rndfac)-bdd),0)
     limy2 <- rndfac*(round(max(vary)/rndfac)+bdd)
     limx1 <- min(varx)
     limx2 <- max(varx)
     ggplot(dfPlot, aes(x=varx, y=vary)) + geom_point() + theme_bw() +geom_line() +
          scale_y_continuous(breaks = seq(from = limy1, to =   limy2, length.out = 3),
                             limits =  c(limy1, limy2)) +
          labs(y = riskTypesShortSelect[num]) +
          theme(axis.title.y = element_text(size = 6)) + labs(x = labelx) +
          scale_x_continuous(breaks = seq(from = limx1, to =   limx2, length.out = 3),
                             limits =  c(limx1, limx2) )
   }
```

```
dfPlot <- data.frame(OutMat.Depend)
q <- list(rep(0,14))
for (iPlot in 2:15){
  q[[iPlot-1]] <- pltfunc(iPlot,varx=dfPlot[["w"]],
                          vary=dfPlot[[paste("u",iPlot, sep="")]],labelx="w")
   }

grid.arrange(q[[1]],  q[[2]],  q[[3]],  q[[4]], q[[5]],
             q[[6]],  q[[7]],  q[[8]],  q[[9]], q[[10]],
             q[[11]], q[[12]], q[[13]], q[[14]], nrow=5 )
```

15.9.2 Example 9.3. Interpreting ANU Lagrange Multipliers

```
# save(ANURTCmax.vec, OutMatANU.ES,
#      file= "../ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")
load(file= "ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")

OutMatPrint.ES <- OutMatANU.ES[, c(1,2,6,22)]
OutMatPrint.ES[,1:3] <- round(OutMatPrint.ES[,1:3], digits = 0)
OutMatPrint.ES[,4] <- round(OutMatPrint.ES[,4], digits = 1)
# colnames(OutMatPrint.ES) <- c("$RTC_{max}$", "$RTC$", "$ES$","$LME_1$" )
colnames(OutMatPrint.ES) <- c("RTCmax", "RTC", "ES","LME1" )

dfPlot <- data.frame(OutMatPrint.ES)
p1     <- ggplot(dfPlot, aes(x=RTC, y=ES)) + geom_point() +
          theme_bw() + geom_line()

tt <- ttheme_default(colhead=list(fg_params = list(parse=TRUE)))
table  <- tableGrob(OutMatPrint.ES, theme=tt)

grid.arrange(table, p1, nrow=1)
```

15.9.3 Example 9.5. Level of Confidence and the ANU Case

```
#load(file = "../Data/ANUCaseData/OutMat.Alpha100K.Rdata")
load(file = "Data/ANUCaseData/OutMat.Alpha100K.Rdata")

OutMatPrint.Alpha <- OutMat.Alpha[,c(1, 3:5)]
colnames(OutMatPrint.Alpha) <- c("Alpha",  "VaR", "ES", "Deriv ES")
OutMatPrint.Alpha[,4] <-
       (OutMatPrint.Alpha[,"ES"]-OutMatPrint.Alpha[,"VaR"])/
                  (1- OutMatPrint.Alpha[,"Alpha"])/100
OutMatPrint.Alpha[,c(1,4)] <-
       round(OutMatPrint.Alpha[,c(1,4)], digits = 2)
OutMatPrint.Alpha[,c(2,3)] <-
       round(OutMatPrint.Alpha[,c(2,3)], digits = 0)
```

```
dfPlot <- data.frame(OutMat.Alpha)
p1     <- ggplot2::ggplot(dfPlot, aes(x=Alpha, y=CTE)) + geom_point() +
            theme_bw()+ geom_line()  + labs(y = 'ES')
table  <- tableGrob(OutMatPrint.Alpha)
grid.arrange(table, p1, nrow=1)
```

15.9.4 Example 9.6. Range Value at Risk and the ANU Case

```
#load(file = "../Data/ANUCaseData/OutMatRVaR100K.Rdata")
load(file = "Data/ANUCaseData/OutMatRVaR100K.Rdata")

OutMatPrint.Beta <- OutMat.RVaR[,c(1,3,6,7)]
colnames(OutMatPrint.Beta) <- c("Beta", "VaR", "RVaR", "Deriv RVaR")
OutMatPrint.Beta[,4] <-
     - (OutMatPrint.Beta[,"VaR"]-OutMatPrint.Beta[,"RVaR"])/
        (1- OutMatPrint.Beta[,"Beta"])/100
OutMatPrint.Beta <- round(OutMatPrint.Beta, digits = 2)

dfPlot <- data.frame(OutMat.RVaR)
p2 <- ggplot2::ggplot(dfPlot, aes(x=Beta, y=RVaR))   +
       geom_point() + theme_bw()+ geom_line() +
       ylim(5700,6800) +
       scale_x_continuous(breaks = c(0,.05,.1), limits =  c(0, .1))

table  <- tableGrob(OutMatPrint.Beta)
grid.arrange(table, p2, nrow=1)
```

15.9.5 Example 9.9. Allocation Sensitivities for a Portfolio of Insurance Stock Returns

```
load(file= "Data/Chap1Data/InsurSectorReturns.Rdata")
load(file="Data/Chap1Data/Frontier.Rdata")

# Training period summary statistics
numTradingDays <- 252 # Approximately nrow(Return_trn)/5
Sigma  <- cov(Return_trn)*numTradingDays
StdDev <- sqrt(diag(Sigma))
Means  <- colMeans(Return_trn)*numTradingDays
#  Determine the optimal allocations

portfolioMarkowitz <- function(mu, Sigma, lmd = 5) {
  c.prop <- Variable(nrow(Sigma))
  prob   <- Problem(Maximize(t(mu) %*% c.prop -
                               lmd*quad_form(c.prop, Sigma)),
                    constraints = list(#c.prop >= 0,
                                       sum(c.prop) == 1))
```

```
  result <- solve(prob)
  c.prop <- as.vector(result$getValue(c.prop))
  names(c.prop) <- colnames(Sigma)
  return(c.prop)
}
LPort <- 10
cstar <- portfolioMarkowitz(mu=Means, Sigma=Sigma, lmd=LPort)

PortfolioMean     <- t(as.vector(Means)) %*% cstar
PortfolioVariance <- t(cstar) %*% Sigma %*% cstar
PortfolioStdDev   <- sqrt(PortfolioVariance)

SigmaInv   <- solve(Sigma)
one.vec    <- rep(1,8)
one.SigInv <- one.vec  %*% SigmaInv
sum.SigInv <- sum(one.SigInv) # sum(SigmaInv)
LMEmu.vec  <- one.SigInv/sum.SigInv
c.mu       <- matrix(0, nrow=8,ncol=8)
one.vec    <- rep(1,8)

MeanPort.muvec     <- rep(0,8)
VariancePort.muvec <- rep(0,8)
for (iaux in 1:8){
  zero.vec         <- rep(0,8) -> oneone.vec
  oneone.vec[iaux] <- 1
  c.mu[iaux,]      <- 1/(2*LPort)*
                SigmaInv %*% (oneone.vec - one.vec*LMEmu.vec[iaux])
  MeanPort.muvec[iaux] <- t(as.vector(Means)) %*% c.mu[iaux,] + cstar[iaux]
  VariancePort.muvec[iaux] <- t(c.mu[iaux,]) %*% Sigma %*% cstar +
                               t(cstar)  %*% Sigma  %*% c.mu[iaux,]
  }

StdDevPort.muvec <- as.numeric(0.5 * PortfolioStdDev**(-1)) * VariancePort.muvec
AssetAllocMat <- rbind(Means, StdDev, cstar, diag(c.mu), MeanPort.muvec, StdDevPort.muvec)
rownames(AssetAllocMat) <- c("$\\mu$", "$\\sigma$",
                        "$c^*$", "$\\partial_{\\mu}~ c^*(\\mu)$",
                        "$\\partial_{\\mu}~ [c^{*'}\\mu]$",
                        "$\\partial_{\\mu}~ SD[c^{*'}X]$")
colnames(c.mu) <- colnames(AssetAllocMat) -> rownames(c.mu)
#rowSums(c.mu)  #  Sum of each row is zero, as anticipated
```

15.9.6 Example 9.10. Robust Allocationss for a Portfolio of Insurance Stock Returns

Parameter Set-Up

```
load(file = "Data/Chap1Data/InsurSectorPrices.RData")

Return <- (prices/stats::lag(prices) - 1)[-1]
```

```
#We now divide the data into a training set and test set:
# Training 2015-01-05 to 2019-12-31
# Testing 2020-06-01 to 2022-05-31

Return_trn <- Return[1:1299, ]
# Return_tst <- Return[1427:1927, ]

T_trn <- nrow(Return_trn)
mu    <- colMeans(Return_trn, na.rm = TRUE)
Sigma <- cov(Return_trn, use = "pairwise.complete.obs")
```

Determine Optimal Robust Allocations

```
portfolioMarkowitzRobust <- function(mu_hat, Sigma_hat, kappa,  LME = 10) {
  N <- length(mu_hat)
  SqRootSig <- chol(Sigma_hat)  # t(SqRootSig) %*% SqRootSig = Sigma
  c.prop <- Variable(N)
  prob <- Problem(Maximize(t(c.prop) %*% mu_hat - kappa*norm2(SqRootSig %*% c.prop)
                  - (LME/2)*( norm2(SqRootSig %*% c.prop) )^2),
                  constraints = list(c.prop  >= 0, sum(c.prop) == 1))
  result <- solve(prob)
  return(as.vector(result$getValue(c.prop)))
}

c_Markowitz            <- portfolioMarkowitzRobust(mu, Sigma, kappa = 0)
names(c_Markowitz)     <- colnames(Sigma)
c_all_Markowitz_robust <- cbind(c_Markowitz)

# multiple robust solutions
T_trn <- nrow(Return_trn)
kappa <- 1.0

c_Markowitz_robust_noisy <- portfolioMarkowitzRobust(mu, Sigma, kappa)
c_all_Markowitz_robust   <- cbind(c_all_Markowitz_robust, c_Markowitz_robust_noisy)

set.seed(357)
library(mvtnorm)
for (i in 1:2) {
  X_noisy <- rmvnorm(n = T_trn, mean = mu, sigma = Sigma)
  mu_noisy <- colMeans(X_noisy)
  c_Markowitz_robust_noisy <- portfolioMarkowitzRobust(mu_noisy, Sigma, kappa)
  c_all_Markowitz_robust   <- cbind(c_all_Markowitz_robust, c_Markowitz_robust_noisy)
}
colnames(c_all_Markowitz_robust) <- c("c_Markowitz", "c_robust_base", "c_robust_1", "c_robust_2")
```

Figure 9.6

```
if (HtmlEval) {
barplot(t(c_all_Markowitz_robust), col = rainbow8equal[1:4],
        legend = colnames(c_all_Markowitz_robust), cex=0.8,
        args.legend = list(x = "topright", ncol = 1, cex=0.6),
        cex.names=0.8, beside = TRUE, cex.lab=0.8,
        main = "", xlab = "Insurance Stocks", ylab = "Proportions")
} else {
  barplot(t(c_all_Markowitz_robust),
          col = c("black", "gray", "lightgrey" ),
          legend = colnames(c_all_Markowitz_robust), cex=0.8,
          args.legend = list(x = "topright", ncol = 1, cex=0.6),
          cex.names=0.8, beside = TRUE, cex.lab=0.8,
          main = "", xlab = "Insurance Stocks", ylab = "Proportions")
  }
```

15.10 Chapter Ten Code

15.10.1 Example 10.1. RTC_{max} as an Auxiliary Variable in the ANU Case

```
# Example101
alpha <- 0.80
bw <- 10
# Function to Compute Retained Claims
RetClaims.fct <- function(coeff, Xsim, ExLoss = TRUE){
  VaR.z0    <- coeff[1]
  theta.vec <- coeff[-1]
  p.num     <- length(theta.vec)
  RetClaims <- rep(0, nrow(Xsim) )
  if (ExLoss){  # Excess of Loss
     for (kidx in 1:p.num){RetClaims <- RetClaims +
                             pmin(Xsim[,kidx], theta.vec[kidx]) }
     } else {  # Quota Share
            RetClaims  <- as.vector(Xsim %*% theta.vec)   }
  return( RetClaims )
}  # end RetClaims.fct function

#  ESKernmin.fct evaluates the ES objective function
#    using kernel smoothing
ESKernmin.fct <- function(coeff, RetClaims){
  VaR.z0        <- coeff[1]
  theta.vec     <- coeff[-1]
  ESKern1.fct   <- function(ytidle) { as.numeric( mean(
                        pmax(RetClaims + bw * ytidle - VaR.z0, 0) ) ) *
                               dnorm(ytidle)  }
  ESKernmin     <- VaR.z0 + integrate(Vectorize(ESKern1.fct),
                                      lower = -6, upper = 6)$value / (1-alpha)
  return( ESKernmin )
}  # end ESKernmin.fct function

# Bring in ANU Claims Data
load(file = "../Data/SimData/AggClaimsMay100Kw0.7.Rdata")
Xsim    <- Claims[,-1]

# save(ANURTCmax.vec, OutMatANU.ES,
#      file= "../ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")
load(file= "../ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")

Comprow <- 5

coeffEX <- OutMatANU.ES[Comprow,c(3,8:21)]
LME.EX <- OutMatANU.ES[Comprow,22]

ESKernminTempEX.fct <- function(coeff){
  RetClaims <- RetClaims.fct(coeff, Xsim, ExLoss = TRUE)
```

```
  return(ESKernmin.fct(coeff, RetClaims) ) }
f0zzEXNum <- numDeriv::hessian(f=ESKernminTempEX.fct, x =  coeffEX)

LambdaMat <- matrix(0,15,15)
u.vec <- coeffEX[-1]
for (kidx in 1:14){
  arg <- (u.vec[kidx] - Xsim[,kidx])/bw
  LambdaMat[kidx+1,kidx+1] <-  mean( dnorm(arg) ) / bw
  }
SLAzz <- f0zzEXNum + LME.EX * LambdaMat
kappa(f0zzEXNum, exact = TRUE)  #285.0969
kappa(SLAzz, exact = TRUE)  #52.42189
round(LambdaMat, digits = 3)

#Function to Compute Simulated RTC with Kernel Smoothing
RTCKernR.fct <- function(coeff.j, Xsim.j){
  RTCKern1.fct  <- function(ytidle) { as.numeric( mean(
                        pmax(Xsim.j - coeff.j + bw * ytidle, 0) ) ) *
                               dnorm(ytidle)  }
  RTCKern      <- integrate(Vectorize(RTCKern1.fct),
                                    lower = -6, upper = 6)$value
  return( RTCKern )
}  # end RTCKernR.fct function

fcon1z.fct <- function(coeff, Xsim){
    fconz.vec <- rep(0,15)
    theta.vec <- coeff[-1]
    for (kidx in 1:14){
    RTCKernj.fct   <- function(coeff.j){RTCKernR.fct(coeff.j, Xsim.j = Xsim[,kidx])}
    fconz.vec[kidx+1] <- numDeriv::grad(f=RTCKernj.fct, x =  theta.vec[kidx])}
   return(fconz.vec)
}

fcon1zEX.vec <- fcon1z.fct(coeffEX, Xsim = Xsim)


SLAzz.Inv <- solve(SLAzz)
partialLME <- -1 / ( t(fcon1zEX.vec) %*% SLAzz.Inv %*% fcon1zEX.vec )
partialz.star <- - SLAzz.Inv %*% fcon1zEX.vec * as.numeric(partialLME)
partialz.starprint <- round(partialz.star, digits = 3)

save(f0zzEXNum, LambdaMat, SLAzz, fcon1zEX.vec, partialLME,
     partialz.star, partialz.starprint,
     file= "../ChapTablesData/Chap10/Example1011.Rdata")
```

15.10.2 Example 10.2. Confidence Level α as an Auxiliary Variable in the ANU Case

```
# Example102
alpha <- 0.80
bw <- 10

# Function to Summarize Risk Measures
ESSummaryKern.fct <- function(coeff, RetClaims, alpha){
  VaR.z0    <- coeff[1]
  theta.vec <- coeff[-1]
  ones      <- rep(1, length(RetClaims) )
  stddev    <- sd(RetClaims)
  VaR       <- as.numeric(quantile(RetClaims, prob = alpha, na.rm = TRUE))
  Kern.df   <- function(y){ yarg <- (y*ones - RetClaims)/bw
                            return(mean(pnorm(yarg)) - alpha)}
  Kern.VaR  <- uniroot(f=Kern.df,lower = VaR - 100, upper = VaR + 100)$root
  ESmin     <- as.numeric(VaR.z0 +
                            mean(pmax(RetClaims - VaR.z0, 0)) /(1-alpha) )
  ES        <- as.numeric(VaR   +
                            mean(pmax(RetClaims - VaR,   0)) /(1-alpha) )
  ESKern1.fct <- function(y) { as.numeric ( mean( pmax(
                           RetClaims -  VaR.z0 +  bw * y, 0) ) ) * dnorm(y) }
  ESKernmin   <- VaR.z0 + integrate(Vectorize(ESKern1.fct),
                                    lower = -6, upper = 6)$value / (1-alpha)
  ESKern2.fct <- function(y) { as.numeric ( mean( pmax(
                           RetClaims - Kern.VaR + bw * y, 0) ) ) * dnorm(y) }
  ESKern      <- Kern.VaR + integrate(Vectorize(ESKern2.fct), lower = -6,
                                      upper = 6)$value / (1-alpha)
  return( c(VaR, Kern.VaR, ESmin, ES,  ESKernmin, ESKern, stddev) )
}  # end ESSummaryKern.fct function

f0alpha.fct <- function(coeff){
  RetClaims <- RetClaims.fct(coeff, Xsim, ExLoss = TRUE)
  ES.Var <- ESSummaryKern.fct(coeff, RetClaims, alpha)[c(2,3)]
  f0alpha <- (ES.Var[2] -ES.Var[1] ) / ( 1 - alpha)
  return(f0alpha)
  }
f0zalphaEXNum <- numDeriv::grad(f=f0alpha.fct, x =  coeffEX)

# save(f0zzEXNum, LambdaMat, SLAzz, fcon1zEX.vec, partialLME,
#      partialz.star, partialz.starprint,
#      file= "../ChapTablesData/Chap10/Example1011.Rdata")
load(file= "../ChapTablesData/Chap10/Example1011.Rdata")

SLAzz.Inv <- solve(SLAzz)
denom <- t(fcon1zEX.vec) %*% SLAzz.Inv %*% fcon1zEX.vec
partialLME <- -t(fcon1zEX.vec) %*% SLAzz.Inv %*% f0zalphaEXNum / denom

partialz.star <- - SLAzz.Inv %*% ( f0zalphaEXNum +
                                   fcon1zEX.vec * as.numeric(partialLME) )
```

```
partialz.starprint <- round(partialz.star, digits = 3) /100

save(f0zzEXNum, LambdaMat, SLAzz, fcon1zEX.vec, f0zalphaEXNum,
     partialLME, partialz.star, partialz.starprint,
     file= "../ChapTablesData/Chap10/Example1012.Rdata")
```

15.10.3 Example 10.3. Parameter Sensitivities for Excess of Loss with Two Risks

Baseline Results from Four Optimization Problems

```
# Example103Base
# First, determine optimal parameters without auxiliary variables
#
#  Set Simulation Parameters
nsim <- 2000000
set.seed(202020)
risk1shape <- 2
risk1scale <- 2000
risk2shape <- 3
risk2scale <- 2000
#  Independent normal random variables
rGaus    <- matrix(rnorm(2*nsim),nrow=nsim,ncol=2)
sigparam <- 0.5
Sigma.11 <- matrix(c(1,sigparam,sigparam,1),nrow=2,ncol=2)
rGaus.new <- rGaus %*% chol(Sigma.11)
UCop1    <- pnorm(rGaus.new[,1])
UCop2    <- pnorm(rGaus.new[,2])
#  Marginal Distribution Random variables
X1 <-          qgamma(p=UCop1, shape = risk1shape, scale = risk1scale)
X2 <- actuar::qpareto(p=UCop2, shape = risk2shape, scale = risk2scale)
alpha0 <- 0.95
alpha1 <- 0.85
alpha2 <- 0.75

# Objective Function
f0.a <- function(u1,u2,a){
  S <- pmin(X1,u1) +  pmin(X2*(1+a/100),u2)
  stddev  <- sd(S)
  Quan0   <- quantile(S, probs = alpha0, na.rm = TRUE)
  Quan1   <- quantile(S, probs = alpha1, na.rm = TRUE)
  Quan2   <- quantile(S, probs = alpha2, na.rm = TRUE)
  excess0 <- pmax(S-Quan0,0)
  CTE0    <- Quan0 + mean(excess0)/(1-alpha0)
  excess1 <- pmax(S-Quan1,0)
  CTE1    <- Quan1 + mean(excess1)/(1-alpha1)
  excess2 <- pmax(S-Quan2,0)
  CTE2    <- Quan2 + mean(excess2)/(1-alpha2)
  Output  <- c(stddev, CTE0, CTE1, CTE2)
```

```
  return(Output)
}

#  Constraint Function
RTC.a <- function(u1,u2,a){
  TotalCost <-
    actuar::mgamma(order=1, shape = risk1shape, scale = risk1scale) +
    actuar::mpareto(order=1, shape = risk2shape, scale = risk2scale*(1+a/100))
  TotalRetained <-
    actuar::levgamma(limit = u1,shape = risk1shape, scale = risk1scale) +
    actuar::levpareto(limit = u2,shape = risk2shape, scale = risk2scale*(1+a/100))
  RTCost<- TotalCost - TotalRetained
  return(RTCost)
}

RTC.Upper <-  actuar::mgamma( order=1, shape = risk1shape, scale = risk1scale) +
              actuar::mpareto(order=1, shape = risk2shape, scale = risk2scale)
RTCmax <- 0.20*RTC.Upper
#  Start retention parameters at mean values
starter <- c(actuar::mgamma(order=1, shape = risk1shape, scale = risk1scale),
            actuar::mpareto(order=1, shape = risk2shape, scale = risk2scale))

fineq <- function(par) {
  h <- NA
  h[1] <- {-(RTC.a(u1=par[1],u2=par[2],a=0)-RTCmax)}
  h[2] <- par[1]
  h[3] <- par[2]
  return(h)}

RMAlternatives <- 1:4
OptOutMat <- matrix(0,7,length(RMAlternatives))

colnames(OptOutMat) <-
  c("Std Dev", paste("ES:",alpha0),
     paste("ES:",alpha1),  paste("ES:",alpha2) )
rownames(OptOutMat) <-
  c(paste("Optimal","$u_1$"), paste("Optimal","$u_2$"), paste("Optimal","$LME$"),
    "Std Dev",  paste("ES:",alpha0),
    paste("ES:",alpha1),  paste("ES:",alpha2))

time4<-Sys.time()
for (kindex in 1:length(RMAlternatives)) {

  f0.Vec <- function(upar){f0.a(u1=upar[1],u2=upar[2],a=0)[kindex]}
  f0.opt <- auglag(par=starter,                    # initial value
                fn=f0.Vec,                        # objective function
                hin=fineq,                        # inequality constraint
                control.outer=list(method="nlminb",trace=FALSE))
  OptOutMat[1:2,kindex] <- f0.opt$par
  OptOutMat[3,kindex]   <- f0.opt$lambda[1]
  OptOutMat[4:7,kindex] <- f0.a(f0.opt$par[1],f0.opt$par[2],a=0)
```

```
}

save(OptOutMat, file="../ChapTablesData/Chap10/Example1021Base.Rdata")
Sys.time() - time4     #Time difference of 5 mins
```

Baseline Results Table 10.1

```
# Example103BaseTable
#load(file = "../ChapTablesData/Chap10/Example1021Base.Rdata")
load(file = "ChapTablesData/Chap10/Example1021Base.Rdata")

alpha0 <- 0.95
alpha1 <- 0.85
alpha2 <- 0.75
rownames(OptOutMat) <-
  c(paste("Optimal","$u_1$"), paste("Optimal","$u_2$"), paste("Optimal", "$LME_1$"),
    "Std Dev",  paste("$ES:$",alpha0),
    paste("$ES:$",alpha1),  paste("$ES:$",alpha2))
colnames(OptOutMat) <-
  c("Std Dev", paste("$ES:$",alpha0),
     paste("$ES:$",alpha1),  paste("$ES:$",alpha2) )

TableGen1(TableData=OptOutMat,
         TextTitle='Excess of Loss: Baseline Results from Four Optimization Problems',
         Align='r', Digits=2, ColumnSpec=1:4, ColWidth0= "3.2cm",
         BorderRight= 1 , ColWidth  = "2.2cm" )
```

Function to Evaluate ES Hessian Using Kernel Methods

```
# ESKernHessFct
ESKernHess.fct <- function(coeff, Xsim, ExLoss = TRUE){
  VaR.z0        <- coeff[1]
  theta.vec     <- coeff[-1]
  numLoss       <- nrow(Xsim)
  nRisks        <- ncol(Xsim)
  ones          <- rep(1, numLoss )
  RetClaims     <- rep(0, numLoss )
  if (ExLoss){  # Excess of Loss
  for (kidx in 1:nRisks){RetClaims <- RetClaims +
                     pmin(Xsim[,kidx], theta.vec[kidx]) }
     } else {  # Quota Share
                    RetClaims  <- as.vector(Xsim %*% theta.vec)  }
  wr    <- dnorm( (VaR.z0*ones - RetClaims)/bw ) /
                    ( numLoss* (1 - alpha)  * bw)
  sqrtWr <- sqrt(wr)
  if (ExLoss){  # Excess of Loss
      grad.G     <-  t( 1* ( t(Xsim) > theta.vec) )
```

```
    } else {  # Quota Share
      grad.G     <- Xsim
    }
  XTilde <- cbind(sqrtWr, -sqrtWr*grad.G)
  hess <- t(XTilde) %*% XTilde
  return(hess)
}
```

New Function to Evaluate *ES* Hessian Using Kernel Methods

```
# ESKernHessFctNEW
#  A check that the new ESKernHess.fct matches the old....
nVars <- 15
nsim <- 2000000
Xsim <- matrix(rnorm(nVars*nsim),nrow=nsim,ncol=nVars)
coeff <- rep(0, nVars+1)
retclaims <- rep(0, nsim)
for (j in 1:nVars){coeff[j+1] <- quantile(Xsim[,j], p = .8)
  retclaims <- retclaims +  pmin(Xsim[,j],coeff[j+1])
  }
coeff[1] <- quantile(retclaims, p = .8)

#coeffEX1 <- c(5615,coeffEX[2:3])
mat1 <- ESKernHess1.fct(coeff = coeff,  Xsim = Xsim, ExLoss = TRUE)
mat2 <- ESKernHess.fct(coeff = coeff, Xsim = Xsim, ExLoss = TRUE)

sum(abs(mat1-mat2))


retclaims <- pmin(X1,coeffEX[2]) +  pmin(X2,coeffEX[3])
summary(retclaims)

ESKernHess1.fct <- function(coeff, Xsim, ExLoss = TRUE){
  VaR.z0        <- coeff[1]
  theta.vec     <- coeff[-1]
  numLoss       <- nrow(Xsim)
  nRisks        <- ncol(Xsim)
  ones          <- rep(1, numLoss )
  RetClaims     <- rep(0, numLoss )
  if (ExLoss){  # Excess of Loss
  for (kidx in 1:nRisks){RetClaims <- RetClaims +
                          pmin(Xsim[,kidx], theta.vec[kidx]) }
     } else {  # Quota Share
                      RetClaims  <- as.vector(Xsim %*% theta.vec)  }
  wr            <- dnorm( (VaR.z0*ones - RetClaims)/bw ) /
                          ( numLoss* (1 - alpha)  * bw)
  partial.z0z0  <- sum( wr )
  if (ExLoss){  # Excess of Loss
      grad.G     <-  t( 1* ( t(Xsim) > theta.vec) )
```

```
    } else {  # Quota Share
      grad.G     <- Xsim
    }
  sqrtWr <- sqrt(wr)
  XTilde <- cbind(sqrtWr, -sqrtWr*grad.G)
  hess <- t(XTilde) %*% XTilde
  return( hess )
}
```

Envelope Theorem Functions

```
# EnvThmFcts
# Xsim[10000,] = Xsim[10000,]

fcon1a.fct <- function(coeff, Xsim, kindex, a, ExLoss = TRUE){
  VaR.z0    <- coeff[1]
  theta.vec <- coeff[-1]
  Xsim[,kindex] <- Xsim[,kindex]*(1+a/100)
  RetClaims <- rep(0, nrow(Xsim) )
  if (ExLoss){ # Excess of Loss
        for (kidx in 1:ncol(Xsim)){RetClaims <- RetClaims +
                                  pmin(Xsim[,kidx], theta.vec[kidx]) }
        RTCza   <-  mean(1 * (Xsim[,kindex] <= theta.vec[kindex]) )
   } else { # Quota Share
        RetClaims <- as.vector(Xsim %*% theta.vec)
        RTCza   <- - mean( Xsim[,kindex] )  }
  CeededClaims <- rowSums(Xsim) - RetClaims
  RTCa <- mean( CeededClaims )
  return( c(RTCa, RTCza ) )
  }

f0za.fct <- function(coeff, Xsim, kindex, a, ExLoss = TRUE){
  VaR.z0      <- coeff[1]
  theta.vec   <- coeff[-1]
  Xsim[,kindex] <- Xsim[,kindex]*(1+a/100)
  numLoss     <- nrow(Xsim)
  ones        <- rep(1, numLoss )
  RetClaims   <- rep(0, numLoss )
  if (ExLoss){        #  Excess Loss
     for ( kidx in 1:ncol(Xsim)){RetClaims <- RetClaims  +
               pmin(Xsim[, kidx], theta.vec[ kidx])  }
     } else { # Quota Share
             RetClaims <- Xsim %*% theta.vec
     }
  VaR.z0arg <- (VaR.z0*ones - RetClaims)/bw
  f0z0a     <- mean(pnorm(VaR.z0arg)) / ( 1 - alpha)
  if (ExLoss){  # Excess of Loss
      grad.G     <-  t( 1* ( t(Xsim) > theta.vec) )
    } else {  # Quota Share
```

```
      grad.G     <- Xsim
    }
  Temp      <- (1 - pnorm(VaR.z0arg)) * grad.G
  f0za      <- colMeans(Temp) / ( 1 - alpha)
  return( c(f0z0a, f0za) )
}
```

Pareto Scale Sensitivity Calculations

```
# Example103ParetoScale
# Use the baseline model to develop sensitivities
#  Start with scale of the second risk
# save(OptOutMat,
#      file="../ChapTablesData/Chap10/Example1021Base.Rdata")
load(file = "../ChapTablesData/Chap10/Example1021Base.Rdata")
bw =1
alpha=0.8
Xsim <- cbind(X1, X2)

RMAlternatives <- 1:4
kindex <- 1

alpha0 <- 0.95
alpha1 <- 0.85
alpha2 <- 0.75

alphavec <- c(0.95, 0.85, 0.75)

partial.zstar <- matrix(0, 3, length(RMAlternatives))
partialLME <- rep(0, length(RMAlternatives) )

time4<-Sys.time()
for (kRMalt in 2:length(RMAlternatives)) {
  alpha <- alphavec[kRMalt-1]
  coeffEX <- c(5615,OptOutMat[c(1:2),kRMalt])
  LME.EX  <- OptOutMat[3,kRMalt]
  f0zz    <- ESKernHess.fct(coeff = coeffEX, Xsim = Xsim, ExLoss = TRUE)
  Lambda  <- matrix(0,3,3)
  u.vec   <- coeffEX[-1]
  for (kidx in 1:2){ Lambda[kidx+1,kidx+1] <-
           density(x=Xsim[,kidx], from = u.vec[kidx], to = u.vec[kidx])$x[1] }
  SLAzz   <- f0zz + LME.EX * Lambda

  fcon1a.afct <- function(a){fcon1a.fct(coeffEX, Xsim, kindex, a, ExLoss = TRUE)[1]}
  fcon1aAst   <- numDeriv::grad(fcon1a.afct, x = 0)
  fcon1za.vecAst <- rep(0,3)
  fcon1z.afct <- function(a){fcon1a.fct(coeffEX, Xsim, kindex, a, ExLoss = TRUE)[2]}
  fcon1za.vecAst[kindex+1] <- numDeriv::grad(fcon1z.afct, x = 0)
```

```
  f0za.vecAst <- rep(0,3)
  for (kidx in 1:3){
    f0za.afct   <- function(a){f0za.fct(coeffEX, Xsim, kindex, a,
                                        ExLoss = TRUE)[kidx]}
    f0za.vecAst[kidx] <- numDeriv::grad(f0za.afct, x = 0)
    }
  SLAza     <- f0za.vecAst + LME.EX * fcon1za.vecAst
  SLAzz.Inv <- solve(SLAzz)
  Denom     <- t(fcon1za.vecAst) %*% SLAzz.Inv %*% fcon1za.vecAst


partialLME[kRMalt] <- (fcon1aAst - t(fcon1za.vecAst)
                              %*% SLAzz.Inv %*% SLAza ) / Denom
partial.zstar[, kRMalt] <- - SLAzz.Inv %*%
                              ( SLAza + fcon1za.vecAst * partialLME[kRMalt] )

}  # end kRMalt loop

round(partial.zstar, digits = 3)

Sys.time()-time4

Scale2OutMat <- matrix(0,3,length(RMAlternatives))
colnames(Scale2OutMat) <- c("Std Dev", paste("ES:",alpha0) ,
                            paste("ES:",alpha1),  paste("ES:",alpha2) )
rownames(Scale2OutMat) <- c(paste("Sensitivity","$u_1$"),
                            paste("Sensitivity","$u_2$"),
                            paste("Sensitivity","$LME$"))
```

Pareto Scale Sensitivity Results Table 10.2

```
# Example103Table102
#load(file = "../ChapTablesData/Chap10/Example1021BaseScale2.Rdata")
load(file = "ChapTablesData/Chap10/Example1021BaseScale2.Rdata")

rownames(Scale2OutMat) <- c(paste("Sensitivity","$u_1$"),
                            paste("Sensitivity","$u_2$"),
                            paste("Sensitivity","$LME_1$"))
colnames(Scale2OutMat) <- c("Std Dev", paste("$ES:$",alpha0) ,
                            paste("$ES:$",alpha1),  paste("$ES:$",alpha2) )

TableGen1(TableData=Scale2OutMat,
         TextTitle='Excess of Loss Sensitivities with Auxiliary $a$ = Pareto Scale',
         Align='r', Digits=3,ColumnSpec=1:4, ColWidth0= "3.2cm",
         BorderRight= 1, ColWidth  = ColWidth4 )
```

Gamma Scale Sensitivity Calculations

```
# Example103GammaScale
# Use the baseline model to develop sensitivities
#
load(file = "../ChapTablesData/Chap10/Example1021Base.Rdata")

#
# Objective Function
f0.a <- function(u1,u2,a){
  S <- pmin(X1*(1+a/100),u1) +  pmin(X2,u2)
  stddev  <- sd(S)
  Quan0   <- quantile(S, probs = alpha0, na.rm = TRUE)
  Quan1   <- quantile(S, probs = alpha1, na.rm = TRUE)
  Quan2   <- quantile(S, probs = alpha2, na.rm = TRUE)
  excess0 <- pmax(S-Quan0,0)
  CTE0    <- Quan0 + mean(excess0)/(1-alpha0)
  excess1 <- pmax(S-Quan1,0)
  CTE1    <- Quan1 + mean(excess1)/(1-alpha1)
  excess2 <- pmax(S-Quan2,0)
  CTE2    <- Quan2 + mean(excess2)/(1-alpha2)
  Output  <- c(stddev, CTE0, CTE1, CTE2)
  return(Output)
}

#  Constraint Function
RTC.a <- function(u1,u2,a){
  TotalCost <-
    actuar::mgamma(order=1, shape = risk1shape, scale = risk1scale*(1+a/100)) +
    actuar::mpareto(order=1, shape = risk2shape, scale = risk2scale)
  TotalRetained <-
    actuar::levgamma(limit = u1,shape = risk1shape, scale = risk1scale*(1+a/100)) +
    actuar::levpareto(limit = u2,shape = risk2shape, scale = risk2scale)
  RTCost<- TotalCost - TotalRetained
  return(RTCost)
}

RMAlternatives <- 1:4
Scale1OutMat <- matrix(0,3,length(RMAlternatives))
colnames(Scale1OutMat) <- c("Std Dev", paste("ES:",alpha0) ,
                            paste("ES:",alpha1),  paste("ES:",alpha2) )
rownames(Scale1OutMat) <- c(paste("Sensitivity","$u_1$"),
                            paste("Sensitivity","$u_2$"),  paste("Sensitivity","$LME$"))

time4<-Sys.time()
for (kindex in 1:length(RMAlternatives)) {

  uparam <- OptOutMat[1:2,kindex]
  LME <- OptOutMat[3,kindex]

  f0.Theta.a.Vec<- function(par){f0.a(u1=par[1],u2=par[2],a=par[3])[kindex]}
```

```
  f0.Theta.a  <- numDeriv::hessian(f=f0.Theta.a.Vec, x = c(uparam,0))

  RTC.a.vec   <- function(par){u1=par[1];u2=par[2];a=par[3]
                           RTC.a(u1,u2,a) }
  Deriv.eq    <- numDeriv::grad(f=RTC.a.vec, x = c(uparam,0))
  feq.Theta   <- Deriv.eq[1:2]
  feq.a       <- Deriv.eq[3]
  feq.Theta.a <- numDeriv::hessian(f=RTC.a.vec, x = c(uparam,0))

  LHS.924     <- c(-feq.a, -(f0.Theta.a[1:2,3] + feq.Theta.a[1:2,3]* LME) )
  SLA         <- f0.Theta.a[1:2,1:2] + LME*feq.Theta.a[1:2,1:2]

  EnvMat      <- matrix(0,ncol = 3, nrow = 3)
  EnvMat[2:3,1:2] <- SLA
  EnvMat[2:3,3]   <- feq.Theta -> EnvMat[1,1:2]

  if (sum(abs(LHS.924)) > 1e-8) {Scale1OutMat[,kindex] <- solve(EnvMat, b=LHS.924 ) }

} # end kindex loop

save(Scale1OutMat, file="../ChapTablesData/Chap10/Example1021BaseScale1.Rdata")
```

Gamma Scale Sensitivity Results Table 10.3

```
# Example103Table103
#load(file = "../ChapTablesData/Chap10/Example1021BaseScale1.Rdata")
load(file = "ChapTablesData/Chap10/Example1021BaseScale1.Rdata")

colnames(Scale1OutMat) <- c("Std Dev", paste("$ES:$",alpha0) ,
                            paste("$ES:$",alpha1),  paste("$ES:$",alpha2) )
rownames(Scale1OutMat) <- c(paste("Sensitivity","$u_1$"),
                            paste("Sensitivity","$u_2$"),
                            paste("Sensitivity","$LME_1$"))

TableGen1(TableData=Scale1OutMat,
         TextTitle='Excess of Loss Sensitivities with Auxiliary $a$ = Gamma Scale',
         Align='r', Digits=3, ColumnSpec=1:4,ColWidth0= "3.2cm",
         BorderRight= 1, ColWidth =  ColWidth4  )
```

15.10.4 Section 10.2.3. Wisconsin Property Fund Sensitivities

Develop Wisconsin Base Results

```
# Section1023WiscBase
# A Few Functions
load(file="../Data/WiscPropFundData/dataout.Rdata")
load(file="../Data/WiscPropFundData/OutMatPolNum210K.Rdata")
load(file = "../Data/WiscPropFundData/SimPropClaims10K.RData")
```

```
PolNum = 2
nsim <- 10000

# Multiplicative auxiliary
LimitClaims <- function(a.vec,indices){
  a <- a.vec[1];SensType <- a.vec[2]
  u.vec <- a.vec[-c(1,2)]
  limClaims <- matrix(0, nrow = nsim, ncol = 6)
  limClaims[,1] <- pmin(u.vec[1], (1+a*(SensType==1)/100)*simBC1[indices,PolNum]/1000)
  limClaims[,2] <- pmin(u.vec[2], (1+a*(SensType==2)/100)*simIM1[indices,PolNum]/1000)
  limClaims[,3] <- pmin(u.vec[3], (1+a*(SensType==3)/100)*simCN1[indices,PolNum]/1000)
  limClaims[,4] <- pmin(u.vec[4], (1+a*(SensType==4)/100)*simCO1[indices,PolNum]/1000)
  limClaims[,5] <- pmin(u.vec[5], (1+a*(SensType==5)/100)*simPN1[indices,PolNum]/1000)
  limClaims[,6] <- pmin(u.vec[6], (1+a*(SensType==6)/100)*simPO1[indices,PolNum]/1000)
  return(limClaims)
  }

# # Additive auxiliary
# LimitClaims <- function(a.vec,indices){
#   a <- a.vec[1];SensType <- a.vec[2]
#   u.vec <- a.vec[-c(1,2)]
#   limClaims <- matrix(0, nrow = nsim, ncol = 6)
#   limClaims[,1] <- pmin(u.vec[1], (a*(SensType==1) + simBC1[indices,PolNum]/1000)
#   limClaims[,2] <- pmin(u.vec[2], (a*(SensType==2) + simIM1[indices,PolNum]/1000)
#   limClaims[,3] <- pmin(u.vec[3], (a*(SensType==3) + simCN1[indices,PolNum]/1000)
#   limClaims[,4] <- pmin(u.vec[4], (a*(SensType==4) + simCO1[indices,PolNum]/1000)
#   limClaims[,5] <- pmin(u.vec[5], (a*(SensType==5) + simPN1[indices,PolNum]/1000)
#   limClaims[,6] <- pmin(u.vec[6], (a*(SensType==6) + simPO1[indices,PolNum]/1000)
#   return(limClaims)
# }

alpha = 0.95
QuanRM.fct <- function(a.coeff){
  a <- a.coeff[1];SensType <- a.coeff[2]
  coeff <- a.coeff[-c(1,2)]
  VaR.x   <- coeff[1]
  u.vec   <- coeff[-1]
  a.vec <- c(a, SensType, u.vec)
  LimClaims <- LimitClaims(a.vec,1:nsim)
  a.vecUnLim <- c(a, SensType, rep(1e10, 6))
  UnLimClaims <- LimitClaims(a.vecUnLim,1:nsim)
  temp    <- colMeans(UnLimClaims) - colMeans(LimClaims)
  RTC.R   <- sum(temp)
  AggY    <- rowSums(LimClaims)
  VaR     <- as.numeric(quantile(AggY, prob = alpha))
  CTEmin  <- as.numeric(VaR.x + ( mean(pmax(AggY - VaR.x, 0)) )/(1-alpha) )
  CTE     <- as.numeric(VaR   + ( mean(pmax(AggY- VaR,   0)) )/(1-alpha) )
  stddev  <- sd(AggY)
  return(c(RTC.R, VaR, CTEmin, CTE, stddev))
  }
```

```
numVars = 6
a.vecUnLim    <- c(0, 1, rep(1e10, 6))
UnLimClaims.0 <- LimitClaims(a.vecUnLim,1:nsim)
TotCost       <- sum(colMeans(UnLimClaims.0))
CstLev = 0.55
RTCmax        <- TotCost * CstLev
starter.coeff <-rep(6e-2,7)

f0.CTE  <- function(coeff){ QuanRM.fct(c(0, 1,coeff))[3] }
RTC.CTE <- function(coeff){ QuanRM.fct(c(0, 1,coeff))[1] }
hin.CTE <- function(coeff) {h <- NA
    h[1] <- RTCmax - RTC.CTE(coeff)
    for (j in 1:(1+numVars)){h[j+1] <- 1e6*coeff[j] }
    return(h)
    }
CTE.opt <- alabama::auglag( par=starter.coeff,
           fn=f0.CTE,
           hin=hin.CTE,
           control.outer=list(method="nlminb",trace=FALSE) )
uparams <- CTE.opt$par[-1]
WisconsinBase <- c(RTCmax , QuanRM.fct(c(0, 1,CTE.opt$par))[-3],
                   uparams, CTE.opt$lambda[1])

save(WisconsinBase, file="../ChapTablesData/Chap10/Section1023Base.Rdata")
```

Develop Wisconsin Sensitivities

```
# Section1023WiscSensi
time1 <-  Sys.time()

load(file="../ChapTablesData/Chap10/Section1023Base.Rdata")
uparam <- WisconsinBase[6:11]
LME    <- WisconsinBase[12]
WiscSensOutMat1 <- matrix(0,ncol = 7, nrow = 6)

for (SensType in 1:6){
  f0.aCTE <- function(a.uparam){
    a <- a.uparam[1]
    uparam <- a.uparam[-1]
    coeff <- c(a, SensType,0, uparam)
    QuanRM.fct(coeff)[3]
    }
  RTC.aCTE <- function(a.uparam){
    a <- a.uparam[1]
    uparam <- a.uparam[-1]
    coeff <- c(a, SensType,0, uparam)
    QuanRM.fct(coeff)[1]
    }
  f0.Theta.a  <- numDeriv::hessian( f=f0.aCTE, x = c(0,uparam) )
```

```
  Deriv.eq    <- numDeriv::grad( f=RTC.aCTE, x = c(0,uparam) )
  feq.Theta   <- Deriv.eq[2:7]
  feq.a       <- Deriv.eq[1]
  feq.Theta.a <- numDeriv::hessian(f=RTC.aCTE, x = c(0,uparam))
  LHS.924     <- c(-feq.a, -(f0.Theta.a[2:7,1] + feq.Theta.a[2:7,1]* LME) )
  SLA         <- f0.Theta.a[2:7,2:7] + LME*feq.Theta.a[2:7,2:7]

  EnvMat      <- matrix(0,ncol = 7, nrow = 7)
  EnvMat[2:7,1:6] <- SLA
  EnvMat[2:7,7]   <- feq.Theta -> EnvMat[1,1:6]

  if ( sum(abs(LHS.924)) > 1e-8) {
       WiscSensOutMat1[SensType,] <- solve(EnvMat, b=LHS.924 ) }
  temp<-  solve(EnvMat, b=LHS.924 )
  # str(temp)
  # tempPrint <- round(WiscSensOutMat1[SensType,], digits=4)
  # cat("Help!", tempPrint, "\n")
}

round(WiscSensOutMat1, digits =3)
save(WiscSensOutMat1, file="../ChapTablesData/Chap10/WiscSensOutMat1Perc.Rdata")

Sys.time() - time1 # Time difference of 17  secs
```

Wisconsin Sensitivities Table 10.4 Results

```
# Section1023WiscTable
load(file = "ChapTablesData/Chap10/WiscSensOutMat1Perc.Rdata")
load(file = "ChapTablesData/Chap10/Section1023Base.Rdata")
numVars = 6

WiscSensOutMat2 <- rbind(WisconsinBase[6:12], WiscSensOutMat1)
#u.names <- paste("u",1:numVars, sep="")

u.names <- c("BC", "IM", "CN", "CO", "PN", "PO")
temp.names <- paste("$u_{",u.names, "}$", sep="")
colnames(WiscSensOutMat2) <- c(temp.names, "$LME$" )
rownames(WiscSensOutMat2) <- c( "$Base$", paste("$Scale_{",u.names, "}$", sep="") )

TableGen1(TableData=WiscSensOutMat2,
         TextTitle='Fund Member 2 Excess of Loss: Scale Parameter Sensitivities',
         Align='r', Digits=3, ColumnSpec=1:7, ColWidth0 = "2cm",
         BorderRight= 1, ColWidth = ColWidth7 )
```

15.10.5 Section 10.2.4. ANU Case Sensitivities

Develop ANU Case Base Results

```
# Section1024ANUBase
# First need to determine baseline optimum
time1 <- Sys.time()
load(file = "../Data/ANUCaseData/OutMatProp.CTE.Rdata")

#  Bring in Claims Data
load(file = "../Data/SimData/AggClaimsMay100Kw0.7.Rdata")
#load(file = "Data/SimData/AggClaimsMay100Kw0.7.Rdata")

RMAggClaims.fct <- function(propCoeff=uPropFixed, AggClaims=Claims, coeff){
  VaR.x     <- coeff[1]
  u.vec     <- c(propCoeff, coeff[-1])  #property is fixed
  LimClaims <- AggClaims
  for (jVar in 1:ncol(LimClaims)){
    LimClaims[,jVar] = pmin(LimClaims[,jVar],u.vec[jVar])
  }
  temp  <- colMeans(AggClaims) - colMeans(LimClaims)
  RTC.R <- sum(temp[-1])        # do not include property
  AggY  <- rowSums(LimClaims)
  VaR   <- as.numeric(quantile(AggY, prob = alpha))
  CTEmin<- as.numeric(VaR.x + ( mean(pmax(AggY - VaR.x, 0)) )/(1-alpha) )
  CTE   <- as.numeric(VaR   + ( mean(pmax(AggY - VaR,   0)) )/(1-alpha) )
  stddev  <- sd(AggY)
  return(c(RTC.R, VaR, CTEmin, CTE, stddev))
}

f0Prop.CTE  <- function(coeff){RMAggClaims.fct(propCoeff=uPropFixed,
                                               AggClaims=Claims, coeff)[3]}
RTCProp.CTE <- function(coeff){RMAggClaims.fct(propCoeff=uPropFixed,
                                               AggClaims=Claims, coeff)[1]}
alpha = 0.95
numVars = 14
uPropFixed = 5000
jCost = 6

RTCmax.vec <- OutMatProp.CTE[,"RTCmax"]
startera   <- OutMatProp.CTE[jCost,6:19]
VaRDir     <-  OutMatProp.CTE[jCost,"VaR"]
starterDir <- as.numeric( c(VaRDir,startera) )

  hinProp.CTE <- function(coeff) {h <- NA
    h[1] <- RTCmax.vec[jCost] - RTCProp.CTE(coeff)
    for (j in 1:numVars){h[j+1] <- coeff[j]}
    return(h)}

  CTEProp.opt <- alabama::auglag(par=starterDir,
           fn=f0Prop.CTE,
           hin=hinProp.CTE,
```

```
          control.outer=list(method="nlminb",trace=FALSE))
    uparams <- CTEProp.opt$par[-1]
    OutVecProp.CTE <- c(RTCmax.vec[jCost],
                RMAggClaims.fct(propCoeff=uPropFixed,AggClaims=Claims,CTEProp.opt$par)[-3],
                                CTEProp.opt$lambda[1], uparams)
save(OutVecProp.CTE,
     file="../ChapTablesData/Chap10/OutVecProp.CTEjCost6.Rdata")

Sys.time() - time1
```

ANU Excess of Loss: Scale Parameter Sensitivities

```
time1 <-  Sys.time()
load(file="../ChapTablesData/Chap10/OutVecProp.CTEjCost6.Rdata")
uparam <- OutVecProp.CTE[7:20]
LME    <- OutVecProp.CTE[6]
ANUSensOutMat1 <- matrix(0,ncol = 15, nrow = 15)
for (SensType in 1:15){
  f0Prop.aCTE  <- function(a.coeff){
    a <- a.coeff[1];coeff <- c(0,a.coeff[-1])
    Claims.a <- Claims
    Claims.a[,SensType] <- Claims.a[,SensType]*(1+a/100)
    RMAggClaims.fct(propCoeff=uPropFixed, AggClaims=Claims.a, coeff)[3]
    }
  RTCProp.aCTE <- function(a.coeff){
    a <- a.coeff[1];coeff <- c(0,a.coeff[-1])
    Claims.a <- Claims
    Claims.a[,SensType] <- Claims.a[,SensType]*(1+a/100)
    RMAggClaims.fct(propCoeff=uPropFixed, AggClaims=Claims.a, coeff)[1]
    }
  f0.Theta.a  <- numDeriv::hessian(f=f0Prop.aCTE, x = c(0,uparam))
  Deriv.eq    <- numDeriv::grad(f=RTCProp.aCTE, x = c(0,uparam))
  feq.Theta   <- Deriv.eq[2:15]
  feq.a       <- Deriv.eq[1]
  feq.Theta.a <- numDeriv::hessian(f=RTCProp.aCTE, x = c(0,uparam))

  LHS.924     <- c(-feq.a, -(f0.Theta.a[2:15,1] + feq.Theta.a[2:15,1]* LME) )
  SLA         <- f0.Theta.a[2:15,2:15] + LME*feq.Theta.a[2:15,2:15]

  EnvMat      <- matrix(0,ncol = 15, nrow = 15)
  EnvMat[2:15,1:14] <- SLA
  EnvMat[2:15,15]   <- feq.Theta -> EnvMat[1,1:14]

  if ( kappa(EnvMat) > 1e10) {
              ANUSensOutMat1[SensType,] <- ginv(EnvMat) %*% LHS.924 }
  if ( kappa(EnvMat) < 1e10) {
              ANUSensOutMat1[SensType,] <- solve(EnvMat, b=LHS.924 ) }
# temp<-  solve(EnvMat, b=LHS.924 )
# temp<-  ginv(EnvMat) %*% LHS.924
```

```
# condNumber <- kappa(EnvMat) # Condition Number for a Matrix
#  str(temp)
  tempPrint <- round(ANUSensOutMat1[SensType,], digits=4)
  cat("Help!", tempPrint, "\n")
}  # end SensType loop

round(ANUSensOutMat1, digits =5)
save(ANUSensOutMat1, file="../ChapTablesData/Chap10/ANUSensOutMatjCost6.Rdata")

Sys.time() - time1  #Time difference of 46  mins
```

Table 10.5

```
# TableExLossSensiANU
load(file = "ChapTablesData/Chap10/ANUSensOutMatjCost6.Rdata")
load(file = "ChapTablesData/Chap10/OutVecProp.CTEjCost6.Rdata")

numVars = 14
ANUSensOutMat2 <- rbind(c(OutVecProp.CTE[7:20],OutVecProp.CTE[6]),
                        ANUSensOutMat1[-1,])
u.names <- paste("$u_{",2:(1+numVars), "}$", sep="")
colnames(ANUSensOutMat2) <- c( u.names, "$LME$" )
rownames(ANUSensOutMat2) <-  c( "$Base$", paste("$Scale_{",2:(1+numVars), "}$", sep="") )


TableGen1(TableData=ANUSensOutMat2[,-15],
         TextTitle='ANU Excess of Loss: Scale Parameter Sensitivities',
         Align='r', Digits=2, ColumnSpec=1:14,
         BorderRight= 1, ColWidth = ColWidth14 ,
         latexFont = 6)
```

ANU Excess of Loss: Bootstrap Standard Errors of Scale Parameter Sensitivities

```
# Section1024ANUBoot
# Bootstrap the results

# library(MASS)
load(file="../ChapTablesData/Chap10/OutVecProp.CTEjCost6.Rdata")

#  Bring in Claims Data
load(file = "../Data/SimData/AggClaimsMay100Kw0.7.Rdata")

uparam <- OutVecProp.CTE[7:20]
LME <- OutVecProp.CTE[6]

numBoot = 10
Output.array <- array(0, dim = c(14, 15, numBoot))
time1 <-  Sys.time()
```

```
size  <- nrow(Claims)/1

for (rBoot in 1:numBoot){
  ANUSensOutMat1 <- matrix(0, nrow = 14, ncol = 15)
  indices <- sample(1:size, size=size, replace=TRUE)
  resample.Claims <- Claims[indices,]
  for (jSensType in 1:14){
    SensType <- jSensType +1
    f0Prop.aCTE  <- function(a.coeff){
    a <- a.coeff[1];coeff <- c(0,a.coeff[-1])
    Claims.a <- resample.Claims
    Claims.a[,SensType] <- Claims.a[,SensType]*(1+a/100)
    RMAggClaims.fct(propCoeff=uPropFixed, AggClaims=Claims.a, coeff)[3]
    }
  RTCProp.aCTE <- function(a.coeff){
    a <- a.coeff[1];coeff <- c(0,a.coeff[-1])
    Claims.a <- resample.Claims
    Claims.a[,SensType] <- Claims.a[,SensType]*(1+a/100)
    RMAggClaims.fct(propCoeff=uPropFixed, AggClaims=Claims.a, coeff)[1]
    }
  f0.Theta.a  <- numDeriv::hessian(f=f0Prop.aCTE, x = c(0,uparam))
  Deriv.eq    <- numDeriv::grad(f=RTCProp.aCTE, x = c(0,uparam))
  feq.Theta   <- Deriv.eq[2:15]
  feq.a       <- Deriv.eq[1]
  feq.Theta.a <- numDeriv::hessian(f=RTCProp.aCTE, x = c(0,uparam))

  LHS.924     <- c(-feq.a, -(f0.Theta.a[2:15,1] + feq.Theta.a[2:15,1]* LME) )
#  LHS.924     <- c(-feq.a, -f0.Theta.a[2:15,1] )
  SLA         <- f0.Theta.a[2:15,2:15] + LME*feq.Theta.a[2:15,2:15]

  EnvMat      <- matrix(0,ncol = 15, nrow = 15)
  EnvMat[2:15,1:14] <- SLA
  EnvMat[2:15,15]   <- feq.Theta -> EnvMat[1,1:14]

  if ( kappa(EnvMat) > 1e10) {
    ANUSensOutMat1[jSensType,] <- ginv(EnvMat) %*% LHS.924 }
  if ( kappa(EnvMat) < 1e10) {
    ANUSensOutMat1[jSensType,] <- solve(EnvMat, b=LHS.924 ) }
  }  # end jSensType Loop

  tempPrint <- round(ANUSensOutMat1, digits=2)
  cat("Another Iteration", tempPrint, "\n")
  Output.array[,,rBoot] <- ANUSensOutMat1
  }  # end rBoot Loop

Output.mean <- ANUSensOutMat1*0 -> Output.sq
for (rBoot in 1:numBoot){
  Output.mean <- Output.mean + Output.array[,,rBoot]
  Output.sq   <- Output.sq   + Output.array[,,rBoot]* Output.array[,,rBoot]
  }
Output.mean <- Output.mean/numBoot
```

```
Output.var  <- Output.sq /numBoot - Output.mean*Output.mean
Output.var  <- Output.var*(Output.var>0)
Output.se   <- sqrt(Output.var) #/sqrt(numBoot)

u.names <- paste("u",2:(1+numVars), sep="")
colnames(Output.mean) <-  c( u.names, "LME")  -> colnames(Output.se)
rownames(Output.mean) <-
       c(paste("$Scale$",2:(1+numVars), sep="")) -> rownames(Output.se)
round(Output.se, digits=3)
save(Output.mean, Output.se, Output.array ,
     file="../ChapTablesData/Chap10/Sect1024BaseScaleRepeat.Rdata")

Sys.time() - time1   # Time difference of 3.9 hours for nsim = 100000.
```

Table 10.6

```
# TableExLossSensiANUBoot
load(file =     "ChapTablesData/Chap10/Sect1024BaseScaleRepeat.Rdata")
u.names <- paste("$u_{",2:(1+numVars), "}$", sep="")
colnames(Output.mean) <-  c( u.names, "$LME$" )  -> colnames(Output.se)
rownames(Output.mean) <-
  paste("$Scale_{",2:(1+numVars), "}$", sep="")  -> rownames(Output.se)

TableGen1(TableData=Output.se[,-15],
         TextTitle='Bootstrap Standard Errors of Scale Parameter Sensitivities',
         Align='r', Digits=2, ColumnSpec=1:14,
         BorderRight= 1, ColWidth="0.5cm" ,
         latexFont = 6)
```

15.10.6 Example 10.4. Property Fund Portfolio Condition Numbers

```
#  CompareEXMats
# Compare Condition Numbers among these alternatives

#save(CompareExMats,file = "RDataFiles/CompareMatEXMar2024.Rdata")
load(file = "ChapTablesData/Chap10/CompareMatEXMar2024.Rdata")

row.names(CompareExMats)<- c("Excess of Loss $f_{0zz}$", "Excess of Loss $SLAzz$")

TableGen1(TableData=as.matrix(CompareExMats[,5]),
         TextTitle='Comparison of Excess of Loss Condition Numbers',
         Align='r', Digits=2, ColumnSpec=1,ColWidth0= "5cm",
         BorderRight=1, ColWidth = ColWidth4)
```

15.10.7 Section 10.3. Investments and Quota Share Portfolios Condition Numbers

```
# CompareQSMats
# Compare Condition Numbers among QS alternatives

#save(CompareQSMats,file = "RDataFiles/CompareMatQSMar2024.Rdata")
load(file = "ChapTablesData/Chap10/CompareMatQSMar2024.Rdata")

row.names(CompareQSMats)<- c("Quota Share $SLA_{zz}=f_{0zz}$",
                             "Quota Share Quad Constraint $f_{0zz}$",
                             "Quota Share Quad Constraint $SLA_{zz}$")
TableGen1(TableData=as.matrix(CompareQSMats[,5])/1e6 ,
         TextTitle='Comparison of Quota Share Condition Numbers (in Millions)',
         Align='r', Digits=3, ColumnSpec=1, ColWidth0= "8cm",
         BorderRight=1, ColWidth = ColWidth4)
```

15.10.8 Example 10.5. Property Fund Portfolio Stochastic Sensitivities

```
# Example105
load(file = "ChapTablesData/Chap10/StochSenEx06Jan2024.Rdata")
row.names(StochSens) <- c("$VaR$",
                          "$u_1^*$", "$u_2^*$", "$u_3^*$",
                          "$u_4^*$", "$u_5^*$", "$u_6^*$")
colnames(StochSens)  <- c("Dec Var","Std Error",
                          "Mean", "Std Dev")

TableGen1(TableData = StochSens,
         TextTitle='Stochastic Sensitivities for Excess of Loss',
         Align='r', ColumnSpec=1:4, Digits = 1,
         BorderRight=c(1,3), ColWidth = ColWidth4)  %>%
     add_header_above(c("","Stochastic Sens" = 2, "Bootstrap" = 2), bold = T) %>%
  footnote(general = "Frees and Shi (2024)",
           general_title = "Source:",
           footnote_as_chunk = TRUE)
```

15.11 Chapter Eleven Code

15.11.1 Example 11.10. Varying the Cyber Risk Premium

```
uPropFixed = 5000
bw = 1

load(file= "ChapTablesData/Chap8/OutMat.Load.Rdata")
load(file = "Data/SimData/AggClaimsMay100Kw0.7.Rdata")

nsim.Load <- 100000
Claims.Load <- Claims[1:nsim.Load,]
MatOut.RM2 <- matrix(0, nrow=nrow(OutMat.Load), ncol =ncol(Claims.Load) ) ->
              MatOut.seRM2

for (MLRow in 1:nrow(MatOut.RM2) ){
  coeff  <- OutMat.Load[MLRow,6:19]
  u.vec  <- c(uPropFixed, coeff)
  VaR    <- OutMat.Load[MLRow,3]
  LimClaims <- Claims.Load
  for (jVar in 1:ncol(Claims.Load)){
    LimClaims[,jVar] = pmin(Claims.Load[,jVar],u.vec[jVar])
    }
  AggY    <- rowSums(LimClaims)
  z0.arg  <- (VaR -  AggY) / bw
  krtheta <- 1 - pnorm(z0.arg)
for (jRisk in 1:ncol(Claims.Load)){
  denom <-  1*(Claims.Load[,jRisk] > u.vec[jRisk])
  numer <-  denom * krtheta
  MatOut.RM2[MLRow,jRisk] <- mean(numer) / mean(denom)
  r12 <- cor(numer,denom)
  CV1 <- sd(numer) /  mean(numer)
  CV2 <- sd(denom) /  mean(denom)
  MatOut.seRM2[MLRow,jRisk] <- MatOut.RM2[MLRow,jRisk] *
            sqrt(CV1**2 + CV2**2 - 2*r12*CV1*CV2) / sqrt(nsim.Load)
  }  #   jRisk  Loop
  }  #   MLRow  Loop
MatOut.RM2[,3]   <- MatOut.RM2[,3]   / OutMat.Load[,1]
MatOut.seRM2[,3] <- MatOut.seRM2[,3] / OutMat.Load[,1]
```

15.12 Chapter Twelve Code

15.12.1 Example 12.1 Reinsurance Expected Payments

This is also the code for **Example 12.3**.

```
# Example 12.1, Reinsurance Expected Payments

data(loss)

Yu <- as.matrix(subset(loss, censored == 0, select = c("loss","alae")))/1000
Yc <- as.matrix(subset(loss, censored == 1, select = c("loss","alae")))/1000

alphastart1 <- 2/(1- mean(Yu[,1])**2/var(Yu[,1]))
thetastart1 <- mean(Yu[,1])*(alphastart1-1)
alphastart2 <- 2/(1- mean(Yu[,2])**2/var(Yu[,2]))
thetastart2 <- mean(Yu[,2])*(alphastart2-1)
fixpar <- c(alphastart1,thetastart1,alphastart2,thetastart2)
lbound = c(1+1e-6,1e-6,1+1e-6,1e-6,-1+1e-6)
ubound = c(1e1,1e6,1e6,1e6,1-1e-6)

#  Start with univariate margins

loglikLoss <- function(par){
  # uncensored
  f1u <- actuar::dpareto(Yu[,1], shape=par[1], scale=par[2])
  lmarg.u  <- -sum(log(f1u))
# censored
  F1c <- actuar::ppareto(Yc[,1], shape=par[1], scale=par[2])
  lmarg.c  <-  -sum(log(1-F1c))
  ll  <- lmarg.u +  lmarg.c
  return(ll)
}

loss.opt.out <- optim(fixpar[1:2],loglikLoss,method=c("L-BFGS-B"),
                      lower=lbound[1:2],upper=ubound[1:2],hessian=TRUE,
                      control=list(maxit=1e6))
#loss.opt.out
tryCatch(loss.SE.out <- sqrt(diag(ginv(loss.opt.out$hessian))),
         error   = function(e) {loss.SE.out <- 0})
loss.BiEstResults <- cbind(loss.opt.out$par,loss.SE.out)

loglikAlae <- function(par){
  # uncensored
  f2u <- actuar::dpareto(Yu[,2], shape=par[1], scale=par[2])
  lmarg.u  <- -sum(log(f2u))
  return(lmarg.u)
}

alae.opt.out <- optim(fixpar[3:4],loglikAlae,method=c("L-BFGS-B"),
```

```
                    lower=lbound,upper=ubound,hessian=TRUE,
                    control=list(maxit=1e6))
tryCatch(alae.SE.out <- sqrt(diag(ginv(alae.opt.out$hessian))),
         error   = function(e) {alae.SE.out <- 0})
alae.BiEstResults <- cbind(alae.opt.out$par,alae.SE.out)

loglikcen <- function(par){
  # uncensored
  f1u <- actuar::dpareto(Yu[,1], shape=par[1], scale=par[2])
  F1u <- actuar::ppareto(Yu[,1], shape=par[1], scale=par[2])
  f2u <- actuar::dpareto(Yu[,2], shape=par[3], scale=par[4])
  F2u <- actuar::ppareto(Yu[,2], shape=par[3], scale=par[4])
  lmarg.u  <- -sum(log(f1u))-sum(log(f2u))
  ljoint.u  <- -sum(log(as.matrix(BiCopPDF(F1u,F2u, family=1, par=par[5]),ncol=1)))
# censored
  #f1c <- dpareto(Yc[,1], shape=par[1], scale=par[2])
  F1c <- actuar::ppareto(Yc[,1], shape=par[1], scale=par[2])
  f2c <- actuar::dpareto(Yc[,2], shape=par[3], scale=par[4])
  F2c <- actuar::ppareto(Yc[,2], shape=par[3], scale=par[4])
  lmarg.c  <- -sum(log(f2c))
  ljoint.c  <- -sum(log(1-as.matrix(
    VineCopula::BiCopHfunc2(F1c,F2c, family=1, par=par[5]),ncol=1)))
  ll  <- lmarg.u + ljoint.u + lmarg.c + ljoint.c
  return(ll)
}

startpar = c(loss.opt.out$par,alae.opt.out$par,0.3)

opt.out <- optim(startpar,loglikcen,method=c("L-BFGS-B"),
                 lower=lbound,upper=ubound,hessian=TRUE,
                 control=list(maxit=1e6))
tryCatch(SE.out <- sqrt(diag(ginv(opt.out$hessian))),
         error   = function(e) {SE.out <- 0})
BiEstResults <- cbind(opt.out$par,SE.out)

Uni.EstResults <- rbind(round(as.matrix(rbind(
                    loss.BiEstResults,alae.BiEstResults)),digits=4), c("NA","NA"))
```

```
set.seed(2023)
nsim    <- 5000000
params  <- BiEstResults[,1]
rho     <- params[5]
cop     <- copula::normalCopula(param = rho,dim=2)
U       <- copula::rCopula(n=nsim, copula = cop)
SimLoss <- actuar::qpareto(U[,1], shape=params[1], scale=params[2]*1000)
SimAlae <- actuar::qpareto(U[,2], shape=params[3], scale=params[4]*1000)
zLoss   <- qnorm(U[,1])
zAlae   <- qnorm(U[,2])
zVec    <- cbind(zLoss,zAlae)
```

```
Sigma      <- matrix(c(1,rho,rho,1), nrow=2, ncol=2)
SigmaChol  <- chol(Sigma)
SigmaInv   <- solve(Sigma)
DerivSigma <- matrix(c(0,1,1,0),nrow=2,ncol=2)
Correction <- tr(SigmaInv %*% DerivSigma)
ZVecStar   <- SigmaInv %*% t(zVec)
zTzP       <- ZVecStar * (DerivSigma %*% ZVecStar)
Weight     <- colSums(zTzP)
Limit      <- c(10,100,500,1000)*1000
R.L        <- c(0,0.25,0.5,0.75,0.95)
RePrems <- matrix(0,nrow = length(Limit),ncol = length(R.L)) -> SeRePrems -> DependSen
for (i in 1:length(Limit)){
  for (j in 1:length(R.L)){Lim = Limit[i]
  Ret   <- R.L[j]*Lim
  g.R.L <- (Ret<SimLoss)*(SimLoss<Lim)*
     (SimLoss-Ret + (SimLoss-Ret)/SimLoss*SimAlae) +
     (SimLoss>Lim)* (Lim-Ret + (Lim-Ret)/Lim*SimAlae)
  FirstTerm     <- g.R.L*Weight
  RePrems[i,j]   <- mean(g.R.L)
  SeRePrems[i,j] <- sd(g.R.L)/sqrt(nsim)
  DependSen[i,j] <- 0.5 * mean(FirstTerm) - 0.5*Correction*RePrems[i,j]
  } }
colnames(RePrems) <- c("0.0", "0.25", "0.50", "0.75", "0.95") -> colnames(SeRePrems)  ->
         colnames(DependSen)
rownames(RePrems) <- c("10,000", "100,000", "500,000", "1,000,000") ->
         rownames(SeRePrems) -> rownames(DependSen)
save(nsim, rho, opt.out, Uni.EstResults, BiEstResults, RePrems, SeRePrems, DependSen,
     file="../ChapTablesData/Chap12/Example1211.Rdata")
```

15.12.2 Example 12.2 Value at Risk for an Insurance Portfolio

This is also the code for **Example 12.4**.

```
# Example 12.2. Value at Risk for the Portfolio

# save(simBC1,simIM1,simPN1,simPO1,simCN1,simCO1,UUMat,
#      file = "SimulatedClaims2000.RData")
load("NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\SimulatedClaims2000.RData")

# save(pairmat, file = "TweedieDependenceParams.RData")
load("NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\TweedieDependenceParams.RData")

load("NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\dataout.RData")

#  Data are Unbalanced
CoverageInd <- cbind(
                    1*(dataout$CoverageBC >0), 1*(dataout$CoverageIM >0),
                    1*(dataout$CoveragePN >0), 1*(dataout$CoveragePO >0),
                    1*(dataout$CoverageCN >0), 1*(dataout$CoverageCO >0) )
#Value at Risk - Quantile
```

```
alphaPort <- 0.98
simTotal  <- rowSums(simBC1+simIM1+simPN1+simPO1+simCN1+simCO1)/1000

nsim   <- length(simTotal)
x.quan.sim  <- quantile(simTotal, probs = alphaPort)
#  Portfolio Density at the VaR
f.xquan.sim <- density(simTotal, from =x.quan.sim, to=x.quan.sim )$y[1]
#  Sensitivity Calculation
df.quan     <- 1*(simTotal <= x.quan.sim)
```

```
# save(pairmat, file = "TweedieDependenceParams.RData")
load("..\\NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\TweedieDependenceParams.RData")

# save(simBC1,simIM1,simPN1,simPO1,simCN1,simCO1,UUMat,
#      file = "SimulatedClaims2000.RData")
load("..\\NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\SimulatedClaims2000.RData")

load("..\\NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\dataout.RData")

#  Sensitivity Calculations
#  Data are Unbalanced
CoverageInd <- cbind(1*(dataout$CoverageBC >0), 1*(dataout$CoverageIM >0),
                     1*(dataout$CoveragePN >0), 1*(dataout$CoveragePO >0),
                     1*(dataout$CoverageCN >0), 1*(dataout$CoverageCO >0) )

Sigma  <- pairmat
nsim   <- 2000
nPol   <- 1098

UUMat.wide  <- array(0, dim = c(nsim,nPol,6) )
for (isim in 1:nsim) {
  UUMat.wide[isim,,1] <- UUMat[((isim-1)*nPol+1):(isim*nPol),1]
  UUMat.wide[isim,,2] <- UUMat[((isim-1)*nPol+1):(isim*nPol),2]
  UUMat.wide[isim,,3] <- UUMat[((isim-1)*nPol+1):(isim*nPol),3]
  UUMat.wide[isim,,4] <- UUMat[((isim-1)*nPol+1):(isim*nPol),4]
  UUMat.wide[isim,,5] <- UUMat[((isim-1)*nPol+1):(isim*nPol),5]
  UUMat.wide[isim,,6] <- UUMat[((isim-1)*nPol+1):(isim*nPol),6]
}
nsStar.wide   <- qnorm(UUMat.wide)

#  This is an array of Sigma Inverse, with zeros where no coverage
SigInvArray <- array(0,dim = c(6,6,nPol))
for (jpol in 1:nPol){
  if ( sum(CoverageInd[jpol,]) > 0 )
  indexj     <- which(CoverageInd[jpol,]==1)
  SigmaInvj  <- solve( Sigma[indexj, indexj] )
  SigMat     <- matrix(0,nrow = 6, ncol = 6)
  SigMat[indexj, indexj] <- SigmaInvj
  SigInvArray[,,jpol] <- SigMat
}
```

```
DFDiff      <- matrix(0,nrow = 6,ncol = 6)
WeightSimMat  <- matrix(0, nrow=nsim, ncol=15)
iWeight <- 0
for (i in 1:5){
  for (j in (i+1):6){
    iWeight         <- iWeight + 1
    DerivSigma      <- matrix(0, nrow=6, ncol=6)
    DerivSigma[i,j] <- 1 -> DerivSigma[j,i]
    WeightSim       <- rep(0, nsim)
    for (isim in 1:nsim) {
      sumzTzPjj <- 0
      for (jpol in 1:nPol){
        if ( sum(CoverageInd[jpol,]) > 0 )
        indexj     <- which(CoverageInd[jpol,]==1)
        SigMatInvjj <- SigInvArray[indexj,indexj, jpol]
        nsStar.pol <- nsStar.wide[isim, jpol, indexj]
        zTzPjj    <- t(nsStar.pol) %*% SigMatInvjj  %*%  DerivSigma[indexj ,indexj ]  %*%
                          SigMatInvjj  %*% nsStar.pol
        sumzTzPjj <- sumzTzPjj + zTzPjj
        }
      WeightSim[isim] <- sumzTzPjj
    }
    WeightSimMat[,iWeight] <- WeightSim
     DFDiff[i,j] <-  0.5*cov(df.quan, WeightSim) -> DFDiff[j,i]
    }
  }
DependSenVar <- -DFDiff/f.xquan.sim
colnames(DependSenVar) <- c("BC","IM","PN","PO","CN","CO") ->
         rownames(DependSenVar)

save(x.quan.sim , DependSenVar, WeightSimMat,SigInvArray,
     file = "../ChapTablesData/Chap12/Example1212.RData")
```

15.12.3 Example 12.5 Expected Shortfall for an Insurance Portfolio

```
#  expected shortfall for the Portfolio

# save(x.quan.sim , DependSenVar, WeightSimMat,SumSigInv,
#      file = "ChapTablesData/Chap12/Example1212.RData")
load(file = "ChapTablesData/Chap12/Example1212.RData")

load("NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\SimulatedClaims2000.RData")

#Value at Risk - Quantile
alphaPort   <- 0.98
simTotal  <- rowSums(simBC1+simIM1+simPN1+simPO1+simCN1+simCO1)/1000

nsim        <- length(simTotal)
```

```
x.quan.sim  <- quantile(simTotal, probs = alphaPort)
excess      <- pmax(simTotal-x.quan.sim,0)
CTE.sim     <- x.quan.sim + mean(excess)/(1-alphaPort)
#  Sensitivity Calculation
df.quan     <- 1*(simTotal <= x.quan.sim)
IntDfDeriv      <- matrix(0,nrow = 6,ncol = 6)
iWeight     <- 0
for (i in 1:5){
  for (j in (i+1):6){
    iWeight <- iWeight + 1
    WeightSim <- WeightSimMat[,iWeight]
    IntDfDeriv[i,j] <-  0.5*cov( WeightSim, (simTotal-x.quan.sim) *
                                  (1-df.quan) ) ->   IntDfDeriv[j,i]
    }
}

DependSenCTE <-  IntDfDeriv/(1-alphaPort)
colnames(DependSenCTE) <- c("BC","IM","PN","PO","CN","CO") ->
             rownames(DependSenCTE)
```

15.12.4 Example 12.7 Effects of Contagion Among Pool Members

```
# Example 12.7 Effects of Contagion Among Pool Members

# save(simBC1,simIM1,simPN1,simPO1,simCN1,simCO1,UUMat,
#      file = "SimulatedClaims2000.RData")
load("..\\NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\SimulatedClaims2000.RData")

# save(twBC,twCN,twCO,twIM,twPN,twPO,
#      file = "TweedieEstimates.RData")
load("..\\NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\TweedieEstimates.RData")

# save(pairmat, file = "TweedieDependenceParams.RData")
load("..\\NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\TweedieDependenceParams.RData")

load("..\\NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\dataout.RData")

#  Data are Unbalanced
CoverageInd <- cbind(
                   1*(dataout$CoverageBC >0), 1*(dataout$CoverageIM >0),
                   1*(dataout$CoveragePN >0), 1*(dataout$CoveragePO >0),
                   1*(dataout$CoverageCN >0), 1*(dataout$CoverageCO >0) )
BCyout <- which(dataout$CoverageBC>0)
IMyout <- which(dataout$CoverageIM>0)
PNyout <- which(dataout$CoveragePN>0)
POyout <- which(dataout$CoveragePO>0)
CNyout <- which(dataout$CoverageCN>0)
COyout <- which(dataout$CoverageCO>0)
```

```
doutBC <- dataout[BCyout,]
doutIM <- dataout[IMyout,]
doutPN <- dataout[PNyout,]
doutPO <- dataout[POyout,]
doutCN <- dataout[CNyout,]
doutCO <- dataout[COyout,]

# Coverage is log coverage
doutBC$CoverageBC <- ifelse(doutBC$CoverageBC>0,log(doutBC$CoverageBC),NA)
doutIM$CoverageIM <- ifelse(doutIM$CoverageIM>0,log(doutIM$CoverageIM),NA)
doutPN$CoveragePN <- ifelse(doutPN$CoveragePN>0,log(doutPN$CoveragePN),NA)
doutPO$CoveragePO <- ifelse(doutPO$CoveragePO>0,log(doutPO$CoveragePO),NA)
doutCN$CoverageCN <- ifelse(doutCN$CoverageCN>0,log(doutCN$CoverageCN),NA)
doutCO$CoverageCO <- ifelse(doutCO$CoverageCO>0,log(doutCO$CoverageCO),NA)


fittedout <- matrix(10,nrow(dataout),6)
fittedout[BCyout,1] <- exp(predict(twBC,newdata=doutBC))
fittedout[IMyout,2] <- exp(predict(twIM,newdata=doutIM))
fittedout[PNyout,3] <- exp(predict(twPN,newdata=doutPN))
fittedout[POyout,4] <- exp(predict(twPO,newdata=doutPO))
fittedout[CNyout,5] <- exp(predict(twCN,newdata=doutCN))
fittedout[COyout,6] <- exp(predict(twCO,newdata=doutCO))
rownames(fittedout) <- dataout$PolicyNum
colnames(fittedout) <- c("BC","IM","PN","PO","CN","CO")

UUMatNormalStar <- qnorm(UUMat)

out1 <- data.frame(phi.max= 165.814,  xi.max = 1.669388)
out2 <- data.frame(phi.max= 849.5301, xi.max = 1.461224)
out3 <- data.frame(phi.max= 376.1897, xi.max = 1.418367)
out4 <- data.frame(phi.max= 322.6615, xi.max = 1.508163)
out5 <- data.frame(phi.max= 336.2968, xi.max = 1.467347)
out6 <- data.frame(phi.max= 302.5563, xi.max = 1.526531)

nsim = 2000
nPol = nrow(dataout)
#nPol = 225

time0 <- Sys.time()

sigsqAlpha.Vec <- c(0, 0.05, 0.10, 0.25, 0.50, 0.75)^2
set.seed(12345)
alphaStarStd    <- matrix(rnorm(6*nsim, mean=0, sd= 1), ncol=6, nrow=nsim)
num.sigAlpha    <- length(sigsqAlpha.Vec)
simTotal.matrix <- matrix(0, nrow = nsim, ncol = num.sigAlpha)
ones <- matrix(1, nrow = nPol, ncol = 1)

for (iAlpha in 1:num.sigAlpha) {
  sigsqAlpha <- sigsqAlpha.Vec[iAlpha]
  alphaStar  <- alphaStarStd * sqrt(sigsqAlpha)
```

```
  SigSubiSqRoot <- diag(rep(1/sqrt(1+sigsqAlpha),6) )
  yNormal    <- UUMatNormalStar
  simBC <- simIM <- simPN <- simPO <- simCN <- simCO <- matrix(0, nsim, nPol)
  for (isim in 1:nsim) {
    blockSim <- ((isim-1)*nPol+1):(isim*nPol)

    yNormal[blockSim,] <-  (ones %*%  alphaStar[isim,] +
                              UUMatNormalStar[blockSim,]) %*%  SigSubiSqRoot
    UDep[blockSim,] <- pnorm(yNormal[blockSim,])

    simBC[isim,] <- qtweedie(UDep[blockSim,1], xi=out1$xi.max,
                             mu=fittedout[1:nPol,1], phi=out1$phi.max)
    simIM[isim,] <- qtweedie(UDep[blockSim,2], xi=out2$xi.max,
                             mu=fittedout[1:nPol,2], phi=out2$phi.max)
    simPN[isim,] <- qtweedie(UDep[blockSim,3], xi=out3$xi.max,
                             mu=fittedout[1:nPol,3], phi=out3$phi.max)
    simPO[isim,] <- qtweedie(UDep[blockSim,4], xi=out4$xi.max,
                             mu=fittedout[1:nPol,4], phi=out4$phi.max)
    simCN[isim,] <- qtweedie(UDep[blockSim,5], xi=out5$xi.max,
                             mu=fittedout[1:nPol,5], phi=out5$phi.max)
    simCO[isim,] <- qtweedie(UDep[blockSim,6], xi=out6$xi.max,
                             mu=fittedout[1:nPol,6], phi=out6$phi.max)
    }
  simTotal <- rowSums(simBC+simIM+simPN+simPO+simCN+simCO)
  simTotal.matrix[,iAlpha] <- simTotal
  cat("Alpha Iteration ",iAlpha , "\n")
  }

save(simTotal.matrix, file= "../ChapTablesData/Chap12/Example1241.Rdata")

Sys.time() - time0
```

15.12.5 Example 12.8 Expected Shortfall Sensitivities for Contagion in an Insurance Pool

```
#  Calculate Weighted Sum of Normal Scores

# save(simBC1,simIM1,simPN1,simPO1,simCN1,simCO1,UUMat,
#      file = "SimulatedClaims2000.RData")
load("..\\NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\SimulatedClaims2000.RData")

# save(pairmat, file = "TweedieDependenceParams.RData")
load("..\\NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\TweedieDependenceParams.RData")

load("..\\NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\dataout.RData")

nsim   <- 2000
nPol <- 1098
```

```
UUMat.wide  <- array(0,dim = c(nsim,nPol,6))

for (isim in 1:nsim) {
  UUMat.wide[isim,,1] <- UUMat[((isim-1)*nPol+1):(isim*nPol),1]
  UUMat.wide[isim,,2] <- UUMat[((isim-1)*nPol+1):(isim*nPol),2]
  UUMat.wide[isim,,3] <- UUMat[((isim-1)*nPol+1):(isim*nPol),3]
  UUMat.wide[isim,,4] <- UUMat[((isim-1)*nPol+1):(isim*nPol),4]
  UUMat.wide[isim,,5] <- UUMat[((isim-1)*nPol+1):(isim*nPol),5]
  UUMat.wide[isim,,6] <- UUMat[((isim-1)*nPol+1):(isim*nPol),6]
}


#  Data are Unbalanced
CoverageInd <- cbind(
                    1*(dataout$CoverageBC >0), 1*(dataout$CoverageIM >0),
                    1*(dataout$CoveragePN >0), 1*(dataout$CoveragePO >0),
                    1*(dataout$CoverageCN >0), 1*(dataout$CoverageCO >0) )

nsStar.wide   <- qnorm(UUMat.wide)
Omega         <- pairmat
nsWeightedSum <- matrix(0, nrow = nsim, ncol = 6) -> nsWeighted.init

for (jpol in 1:nPol){
  if ( sum(CoverageInd[jpol,]) > 0 )
  indexj     <- which(CoverageInd[jpol,]==1)
  OmegaInvj  <- solve( Omega[indexj, indexj]  )
  #pol.block  <- ((jpol-1)*nsim + 1):(jpol*nsim)
  nsStar.pol <- nsStar.wide[, jpol, indexj]
  nsWeightedSum.pol <- nsWeighted.init
  nsWeightedSum.pol[, indexj ] <- as.matrix( nsStar.pol %*% OmegaInvj  )
  nsWeightedSum     <- nsWeightedSum + nsWeightedSum.pol
}


#  The 7th category is for all diagonals equal
WeightSigma <- matrix(0, nsim, 7)

for (jweight in 1:7){
  DerivSigma <- matrix(0,6,6)
  if (jweight < 7) {DerivSigma[jweight,jweight] <- 1
  } else  {DerivSigma <- diag(6) }
  for (isim in 1:nsim){
    WeightSigma[isim,jweight] <-  as.numeric( nsWeightedSum[isim,] %*%
                          DerivSigma %*% as.matrix(nsWeightedSum[isim,]) )
  }
  }

save(nsim, nsWeightedSum, WeightSigma, Omega,
     file = "../ChapTablesData/Chap12/Example1242.RData")
```

```
#  Expected Shortfall for the Portfolio

# save(simBC1,simIM1,simPN1,simPO1,simCN1,simCO1,UUMat,
#      file = "SimulatedClaims2000.RData")
load("NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\SimulatedClaims2000.RData")

# save(pairmat, file = "TweedieDependenceParams.RData")
load("NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\TweedieDependenceParams.RData")

load("NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\dataout.RData")

# save(twBC,twCN,twCO,twIM,twPN,twPO,
#      file = "TweedieEstimates.RData")
load("NumericalWork\\MultiFreqSevExample\\Finalcode21Feb2016\\TweedieEstimates.RData")

#  Data are Unbalanced
CoverageInd <- cbind(
                    1*(dataout$CoverageBC >0), 1*(dataout$CoverageIM >0),
                    1*(dataout$CoveragePN >0), 1*(dataout$CoveragePO >0),
                    1*(dataout$CoverageCN >0), 1*(dataout$CoverageCO >0) )
#Value at Risk - Quantile
alphaPort <- 0.80
simTotal  <- rowSums(simBC1+simIM1+simPN1+simPO1+simCN1+simCO1)/1000
nsim      <- length(simTotal)
x.quan.sim  <- quantile(simTotal, probs = alphaPort)

#  Portfolio Density at the VaR
f.xquan.sim <- density(simTotal, from =x.quan.sim, to=x.quan.sim )$y[1]

#  Sensitivity Calculation

# save(nsim,nsWeightedSum, WeightSigma,Omega,
#      file = "../ChapTablesData/Chap12/Example1242.RData")
load(file = "ChapTablesData/Chap12/Example1242.RData")

df.quan     <- 1*(simTotal <= x.quan.sim)

F.Sigma <- rep(0,7)
for (jweight in 1:7){
   F.Sigma[jweight] <- 0.5 * cov(WeightSigma[,jweight], df.quan )
}
# Value at Risk
deltaVaR <-  -F.Sigma/f.xquan.sim /1e3
#round(deltaVaR, digits = 2) * (0.05)^2

#  expected shortfall
excess      <- pmax(simTotal-x.quan.sim,0)
ES.sim     <- x.quan.sim + mean(excess)/(1-alphaPort)

IntF.Sigma <- rep(0,7)
for (jweight in 1:7){
```

```
    WeightSim <- WeightSigma[,jweight]
    IntF.Sigma[jweight] <-  0.5 * cov( WeightSim,
                                       (simTotal-x.quan.sim)*(1-df.quan) )
    }

deltaES <-  IntF.Sigma/( 1 - alphaPort ) / 1000 # in millions

deltaESOut <- round(deltaES, digits = 3)
Col1    <- c("BC",  "IM", "PN", "All Risks")
Col3    <- c("PO",  "CN", "CO", "")
Col2    <- c(deltaESOut[1:3], deltaESOut[7])
Col4    <- c(deltaESOut[4:6], "")
OutMat  <- cbind(Col1, Col2,Col3, Col4 )
colnames(OutMat) <- c("Risk", "$ES$ Sensitivity", "Risk", "$ES$ Sensitivity")
```

16

Under the Hood: Selected Proofs and Theory Development

16.1 Chapter Two Theory Development

16.1.1 Section 2.1. Show Development of the ES

To see the connections between the second and third equalities, use a variable substitution, $z = VaR_a(X) = F^{-1}(a)$ so that $F(z) = a$ and $f(z)dz = da$. With this, we have

$$\begin{aligned}\int_d^u VaR_a(X)\, da &= \int_d^u F_a^{-1}\, da = \int_{F_d^{-1}}^{F_u^{-1}} zf(z)dz \\ &= -z[1-F(z)]\Big|_{F_d^{-1}}^{F_u^{-1}} + \int_{F_d^{-1}}^{F_u^{-1}} [1-F(z)]dz \\ &= F_d^{-1}(1-d) - F_u^{-1}(1-u) + [\mathrm{E}(X \wedge F_u^{-1}) - \mathrm{E}(X \wedge F_d^{-1})].\end{aligned}$$

Thus,

$$\begin{aligned}\tfrac{1}{1-\alpha}\int_\alpha^1 VaR_a(X)da &= \tfrac{1}{1-\alpha}\left\{F_\alpha^{-1}(1-\alpha) + [\mathrm{E}(X) - \mathrm{E}(X \wedge F_\alpha^{-1})]\right\} \\ &= F_\alpha^{-1} + \tfrac{1}{1-\alpha}[\mathrm{E}(X) - \mathrm{E}(X \wedge F_\alpha^{-1})],\end{aligned}$$

as claimed.

16.1.2 Section 2.2.3. Show Development of the ES

For $\alpha < F(d)$, we have $VaR_\alpha[g(X)] = 0$. Using the second expression in equation (2.3),

$$\begin{aligned}ES_\alpha[g(X)] &= \tfrac{1}{1-\alpha}\mathrm{E}[g(X)] \\ &= \tfrac{c}{1-\alpha}\left\{\mathrm{E}(X \wedge u) - \mathrm{E}(X \wedge d)\right\},\end{aligned}$$

as required. For $\alpha \geq F(u)$, we have $VaR_\alpha[g(X)] = c(u-d) = ES_\alpha[g(X)]$, because g is limited by $c(u-d)$.

For $F(d) \leq \alpha < F(u)$, we have $VaR_\alpha[g(X)] = c(F_\alpha^{-1} - d)$ from the second line of Display (2.10). Using the first line from equation (2.3), we have

$$\begin{aligned}ES_\alpha[g(X)] &= \tfrac{1}{1-\alpha}\int_\alpha^1 VaR_a[g(X)]da \\ &= \tfrac{c}{1-\alpha}\int_\alpha^{F(u)}[F_a^{-1} - d]da + \tfrac{c}{1-\alpha}\int_{F(u)}^1 [u-d]da\end{aligned}.$$

Further, in the proof of equation (2.3), we showed that

$$\int_d^u F_a^{-1}\, da = F_d^{-1}(1-d) - F_u^{-1}(1-u) + [\mathrm{E}(X \wedge F_u^{-1}) - \mathrm{E}(X \wedge F_d^{-1})].$$

Thus, the expected shortfall for retained risks can be expressed as

$$\begin{aligned} ES_\alpha[g(X)] &= \tfrac{c}{1-\alpha}\int_\alpha^{F(u)}(F_a^{-1}-d)da + \tfrac{1-F(u)}{1-\alpha}c(u-d) \\ &= \tfrac{c}{1-\alpha}\left\{\alpha(1-F_\alpha^{-1}) - u[1-F(u)] + [\mathrm{E}(X\wedge u) - \mathrm{E}(X\wedge F_\alpha^{-1})]\right. \\ &\qquad \left. -d[F(u)-\alpha] + [1-F(u)]c(u-d)\right\} \\ &= \tfrac{c}{1-\alpha}\left\{\mathrm{E}(X\wedge u) - \mathrm{E}(X\wedge F_\alpha^{-1}) + (1-\alpha)(F_\alpha^{-1}-d)\right\}. \end{aligned}$$

This suffices to establish equation (2.11).

16.1.3 Section 2.2.3. Show Development of the $RVaR$

Starting with equations (2.4) and (2.10), we have

$$\begin{aligned} RVaR_{(\alpha,\beta)}[g(X)] &= \tfrac{1}{\beta}\int_\alpha^{\alpha+\beta} VaR_a[g(X)]da \\ &= \begin{cases} 0 & \alpha+\beta < F(d) \\ \frac{1}{\beta}\int_{F(d)}^{\alpha+\beta} c(F_a^{-1}-d)da & \alpha < F(d) < \alpha+\beta < F(u) \\ \frac{1}{\beta}\int_{F(d)}^{F(u)} c(F_a^{-1}-d)da + \frac{1}{\beta}\int_{F(u)}^{\alpha+\beta} c(u-d)da & \alpha < F(d) < F(u) < \alpha+\beta \\ \frac{1}{\beta}\int_{\alpha}^{F(u)} c(F_a^{-1}-d)da + \frac{1}{\beta}\int_{F(u)}^{\alpha+\beta} c(u-d)da & F(d) < \alpha < F(u) < \alpha+\beta \\ \frac{1}{\beta}\int_{\alpha}^{\alpha+\beta} c(F_a^{-1}-d)da & F(d) < \alpha < \alpha+\beta < F(u) \\ \frac{1}{\beta}\int_{\alpha}^{\alpha+\beta} c(u-d)da & F(d) < F(u) < \alpha \end{cases}. \end{aligned}$$

For the case $\alpha < F(d) < \alpha+\beta < F(u)$, we have

$$\begin{aligned} &\tfrac{1}{\beta}\textstyle\int_{F(d)}^{\alpha+\beta} c(F_a^{-1}-d)da \\ &= \tfrac{c}{\beta}\left\{d[1-F(d)] - F_{\alpha+\beta}^{-1}(1-\alpha-\beta)\right. \\ &\quad \left. +[\mathrm{E}(X\wedge F_{\alpha+\beta}^{-1}) - \mathrm{E}(X\wedge d)] - d[\alpha+\beta-F(d)]\right\} \\ &= \tfrac{c}{\beta}\left\{d(1-\alpha-\beta) - F_{\alpha+\beta}^{-1}(1-\alpha-\beta)\right. \\ &\quad \left. +[\mathrm{E}(X\wedge F_{\alpha+\beta}^{-1}) - \mathrm{E}(X\wedge d)]\right\} \\ &= \tfrac{c}{\beta}\left\{(1-\alpha-\beta)(d-F_{\alpha+\beta}^{-1}) + [\mathrm{E}(X\wedge F_{\alpha+\beta}^{-1}) - \mathrm{E}(X\wedge d)]\right\}. \end{aligned}$$

For the case $\alpha < F(d) < F(u) < \alpha+\beta$, we have

$$\begin{aligned} &\tfrac{1}{\beta}\textstyle\int_{F(d)}^{F(u)} c(F_a^{-1}-d)da + \frac{1}{\beta}\int_{F(u)}^{\alpha+\beta} c(u-d)da \\ &= \tfrac{c}{\beta}\left\{\textstyle\int_{F(d)}^{F(u)} F_a^{-1}da - d[F(u)-F(d)] + (u-d)[\alpha+\beta-F(u)]\right\} \\ &= \tfrac{c}{\beta}\left\{d[1-F(d)] - u[1-F(u)] + [\mathrm{E}(X\wedge u) - \mathrm{E}(X\wedge d)]\right. \\ &\quad \left. -d[F(u)-F(d)] + (u-d)[\alpha+\beta-F(u)]\right\} \\ &= \tfrac{c}{\beta}\left\{(1-\alpha-\beta)(d-u) + [\mathrm{E}(X\wedge u) - \mathrm{E}(X\wedge d)]\right\}. \end{aligned}$$

For the case $F(d) < \alpha < F(u) < \alpha + \beta$, we have

$$\begin{aligned}
&\tfrac{1}{\beta}\textstyle\int_{\alpha}^{F(u)} c(F_a^{-1} - d)da + \tfrac{1}{\beta}\int_{F(u)}^{\alpha+\beta} c(u-d)da \\
&= \tfrac{c}{\beta}\left\{\textstyle\int_{\alpha}^{F(u)} F_a^{-1}da - d(F(u)-\alpha) + (\alpha+\beta-F(u)(u-d)\right\} \\
&= \tfrac{c}{\beta}\left\{F_\alpha^{-1}(1-\alpha) - u[1-F(u)] + [\mathrm{E}(X\wedge u) - \mathrm{E}(X\wedge F_\alpha^{-1})]\right. \\
&\quad \left. -d(F(u)-\alpha) + [\alpha+\beta-F(u)](u-d)\right\} \\
&= \tfrac{c}{\beta}\left\{(1-\alpha)(F_\alpha^{-1}-u) - \beta(d-u) + [\mathrm{E}(X\wedge u) - \mathrm{E}(X\wedge F_\alpha^{-1})]\right\}
\end{aligned}$$

For the case $F(d) < \alpha < \alpha + \beta < F(u)$, we have

$$\begin{aligned}
&\tfrac{1}{\beta}\textstyle\int_{\alpha}^{\alpha+\beta} c(F_a^{-1} - d)da \\
&= \tfrac{c}{\beta}\left\{\textstyle\int_{\alpha}^{\alpha+\beta} F_a^{-1}da - d\beta\right\} \\
&= \tfrac{c}{\beta}\left\{F_\alpha^{-1}(1-\alpha) - F_{\alpha+\beta}^{-1}(1-\alpha-\beta) + [\mathrm{E}(X\wedge F_{\alpha+\beta}^{-1}) - \mathrm{E}(X\wedge F_\alpha^{-1})] - d\beta\right\} \\
&= \tfrac{c}{\beta}\left\{(1-\alpha)(F_\alpha^{-1} - F_{\alpha+\beta}^{-1}) - \beta(d - F_{\alpha+\beta}^{-1}) + [\mathrm{E}(X\wedge F_{\alpha+\beta}^{-1}) - \mathrm{E}(X\wedge F_\alpha^{-1})]\right\}
\end{aligned}$$

For the case $F(d) < F(u) < \alpha$, we have

$$\tfrac{1}{\beta}\textstyle\int_{\alpha}^{\alpha+\beta} c(u-d)da = c(u-d)$$

This development establishes equation (2.12).

16.2 Chapter Three Theory Development

16.2.1 Section 3.1. Verification of Proposition 3.1

For a fixed c, the end of the period expected utility is

$$U_1(c) = \mathrm{E}\ (U[W - cP_0 - (1-c)X) = \int\ U[W - x + c(x - P_0)]\ dF(x).$$

How does the choice of c affect end of period expected utility? Taking derivatives yields

$$\begin{aligned}
U_1'(c) &= \partial_c\ U_1(c) = \partial_c \textstyle\int\ U[W - x + c(x-P_0)]\ dF(x) \\
&= \textstyle\int\ U'[W - x + c(x-P_0)](x - P_0)\ dF(x)
\end{aligned}$$

and

$$\begin{aligned}
U_1''(c) &= \partial_c\ U_1'(c) = \partial_c \textstyle\int\ U''[W - x + c(x-P_0)](x - P_0)\ dF(x) \\
&= \textstyle\int\ U''[W - x + c(x-P_0)](x - P_0)^2\ dF(x) \leq 0
\end{aligned}$$

because $U''(\cdot) \leq 0$.

Evaluating $U_1'(c)$ at $c = 1$ results in

$$\begin{aligned} U_1'(1) &= \int U'[W - x + c(x - P_0)](x - P_0)\ dF(x)\big|_{c=1} \\ &= \int U'[W - P_0](x - P_0)\ dF(x) \\ &= U'[W - P_0)](\mathrm{E}[X] - P_0) \\ &= -\lambda_{SL}\ \mathrm{E}[X]\ U'[W - P_0)] \geq 0 \end{aligned}$$

because $\lambda_{SL} \leq 0$. By the mean value theorem, $U'(c) \geq 0$ for all $c \in [0, 1]$. Thus, the choice $c = 1$ maximizes end of the period expected utility, as required.

16.2.2 Section 3.1. Verification of Proposition 3.2

This proof follows that given by Bernard (2013) that in turn is based on the work of Cummins and Mahul (2004).

Fix $X = x$ to be a realization of the loss and let y be an indemnity amount. Define

$$f(y) = U(W - P_0 - x + y) - \lambda y, \quad 0 \leq y \leq u_{lim}.$$

The function $f(y)$ is the difference between a concave increasing function $U(\cdot)$ and a linear function and so, to maximize it, we can take a derivative. Setting the derivative equal to zero yields

$$\begin{aligned} f'(y)|_{y=y_{opt}} &= U'(W - P_0 - x + y)|_{y=y_{opt}} - \lambda = 0 \\ &\Rightarrow y_{opt} = x + P_0 - W + U'^{(-1)}(\lambda). \end{aligned}$$

Imposing the upper limit of liability, we have $y_{opt}(\lambda) = u_{lim} \wedge [x + P_0 - W + U'^{(-1)}(\lambda)]_+$.

Next, choose λ such that $\mathrm{E}[y_{opt}(\lambda)] = P_0/(1 + \lambda_{SL})$ and call this value λ_0. Now, by construction, for any $I(x) = y$,

$$\begin{aligned} &U[W - P_0 - x + I(x)] - \lambda_0 I(x) = f(y) \\ &\quad \leq f[y_{opt}(\lambda_0)] = U[W - P_0 - x + y_{opt}(\lambda_0)] - \lambda_0\ y_{opt}(\lambda_0). \end{aligned}$$

Take expectations of both sides yields

$$\mathrm{E}\ (U[W - P_0 - X + I(X)]) \leq \mathrm{E}\ (U[W - P_0 - X + y_{opt}(\lambda_0)]),$$

because $\mathrm{E}\ [I(X)] = \frac{P_0}{1+\lambda_{SL}} = \mathrm{E}\ [y_{opt}(\lambda_0)]$. Thus, the $y_{opt}(\lambda_0)$ produces a larger expected end of the period utility than any other candidate $I(x)$. We write $u_{opt} = W - P_0 - U'^{(-1)}(\lambda_0)$.

16.2.3 Section 3.1. Verification of Proposition 3.3

Proof of the Proposition. For any $g(\cdot)$ where $\text{Var}[g(X)] = Q$, the insurer's uncertainty can be written as

$$\begin{aligned} Var[X - g(X)] &= Var(X) + Var[g(X)] - 2Cov[X, g(X)] \\ &= Var(X) + Q - 2Corr[X, g(X)] \times \sqrt{Q}\sqrt{Var(X)}, \end{aligned}$$

using the law of total variation. In this expression, we remark that Q and $\text{Var}(X)$ do not change with the choice of g. Thus, we can minimize $\text{Var}[X - g(X)]$ by maximizing the correlation $Corr[X, g(X)]$. If we use a coinsurance contract, then $Corr[X, g(X)] = Corr[X, (1-c)X] = 1$, the maximum possible correlation. With the coinsurance contract, we have $\text{Var}(cX) = Q$ and so $c^2 = Q/Var(X)$. This establishes the proposition.

Albrecher et al. (2017) attribute this result to Pesonen (1984).

16.2.4 Section 3.1. Verification of Proposition 3.4

Proof of the Proposition. With $RTC_{max} = \text{E}[X] - \text{E}[g(X)]$, let us define $K = \text{E}[g(X)] = \text{E}[X] - RTC_{max}$. Now, add and subtract a constant u and expand the square to get

$$\begin{aligned} \text{Var}[g(X)] &= \text{E}[g(X) - K]^2 = \text{E}[g(X) - u + u - K]^2 \\ &= \text{E}[g(X) - u]^2 + (u - K)^2 + 2\text{E}[(g(X) - u)(u - K)] \\ &= \text{E}[g(X) - u]^2 - (u - K)^2, \end{aligned}$$

because $\text{E}[g(X)] = K$.

Now, for any retention function, we have $g_*(X) \le X$, that is, the retained claims are less than or equal to total claims. Using the notation $g_u(X) = X \wedge u$ for the upper limit on retained losses, we have

$$\begin{aligned} u - g_u(X) &= u - (X \wedge u) \\ &= (u - X) \wedge 0 \\ &\le [u - g_*(X)] \wedge 0. \end{aligned}$$

Squaring each side yields

$$[u - g_u(X)]^2 \le [u - g_*(X)]^2 \wedge 0 \le [u - g_*(X)]^2.$$

Returning to our expression for the variance, we have

$$\begin{aligned} \text{Var}[g_u(X)] &= \text{E}[g_u(X) - u]^2 - (u - K)^2 \\ &\le \text{E}[g_*(X) - u]^2 - (u - K)^2 = \text{Var}[g_*(X)], \end{aligned}$$

for any retention function g_* where $\text{E}[g_*(X)] = K$. This establishes the proposition.

16.2.5 Section 3.5.3. Show the Derivative of the Optimal ES

From Table 2.2, we have

$$
\begin{aligned}
&\partial_{d_*} ES(d_*, u_*) \\
&= \partial_d ES(d,u)|_{d=d_*,u=u_*} + \partial_u ES(d,u)|_{d=d_*,u=u_*} \times \partial_{d_*} u_* \\
&= \begin{cases} \frac{-1}{1-\alpha}[1-F(d_*)] + \frac{1}{1-\alpha}[1-F(u_*)]\frac{1-F(d_*)}{1-F(u_*)} & \text{if } \alpha < F(d_*) \\ -1 + \frac{1}{1-\alpha}[1-F(u_*)]\frac{1-F(d_*)}{1-F(u_*)} & \text{if } F(d_*) \le \alpha \le F(u_*) \\ -1 + \frac{1-F(d_*)}{1-F(u_*)} & \text{if } F(u_*) < \alpha \end{cases} \\
&= \begin{cases} 0 & \text{if } \alpha < F(d_*) \\ \frac{\alpha - F(d_*)}{1-\alpha} & \text{if } F(d_*) \le \alpha \le F(u_*) \\ \frac{F(u_*)-F(d_*)}{1-F(u_*)} & \text{if } F(u_*) < \alpha. \end{cases}
\end{aligned}
$$

This establishes that $\partial_{d_*} ES(d_*, u_*) \ge 0$.

16.3 Chapter Four Theory Development

16.3.1 Section 4.1.1.1. Verification of the Optimal Retention Proportions

Minimizing $\text{Var}(Y_{retained})$ subject to $\text{E}(Y_{reinsurer}) = RTC_{max}$ is a constrained optimization problem - we can use the method of Lagrange multipliers, Section 3.3.4, to solve this. To this end, define the Lagrangian

$$
\begin{aligned}
LA &= \text{Var}(Y_{retained}) + LME\ [\text{E}(Y_{reinsurer}) - RTC_{max}] \\
&= \textstyle\sum_{i=1}^{p} c_i^2\ \text{Var}(X_i) + LME\ [\sum_{i=1}^{p}(1-c_i)\ \text{E}(X_i) - RTC_{max}].
\end{aligned}
$$

Taking a partial derivative with respect to LME and setting this equal to zero simply means that the constraint, $\text{E}(Y_{reinsurer}) = RTC_{max}$, is enforced and we have to choose the proportions c_i to satisfy this constraint. Moreover, taking the partial derivative with respect to each proportion c_i yields

$$
\frac{\partial}{\partial c_i} LA = 2c_i\ \text{Var}(X_i) - LME\ \text{E}(X_i) = 0.
$$

Solving for c_i yields equation (4.1).

With our budget constraint, we may determine LME as the solution of

$$
RTC_{max} \quad = \textstyle\sum_{i=1}^{p}(1-c_i)\text{E}(X_i) = \sum_{i=1}^{p} \text{E}(X_i) - \frac{LME}{2} \sum_{i=1}^{p} \frac{[\text{E}(X_i)]^2}{\text{Var}(X_i)},
$$

that yields equation (4.2). Use this value of LME to determine the proportions.

16.3.2 Section 4.1.2.2. Verification of the Optimal Excess of Loss Limits

For excess of loss, recall that $Y_{retained} = S(u_1, \ldots, u_p) = \sum_{i=1}^{p} X_i \wedge u_i$. Minimizing $\text{Var}(Y_{retained})$ subject to $\text{E}(Y_{reinsurer}) = RTC_{max}$ is a constrained optimization problem - we can use the method of Lagrange multipliers to solve this. Define the Lagrangian

$$\begin{aligned} LA &= \text{Var}(Y_{retained}) + LME\{\text{E}(Y_{reinsurer}) - RTC_{max}\} \\ &= \sum_{i=1}^{p} \text{Var}(X_i \wedge u_i) + LME\{[\sum_{i=1}^{p} \text{E}(X_i) - \text{E}(X_i \wedge u_i)] - RTC_{max}\}. \end{aligned}$$

We first recall the relationships

$$\text{E}(X \wedge u) = \int_0^u [1 - F(x)]dx$$

and

$$\text{E}(X \wedge u)^2 = 2\int_0^u x[1 - F(x)]dx.$$

Taking a partial derivative with respect to LME and setting this equal to zero simply means that the constraint, $\text{E}(Y_{reinsurer}) = RTC_{max}$, is enforced and we have to choose the limits u_i to satisfy this constraint. Moreover, taking the partial derivative with respect to each limit u_i yields

$$\begin{aligned} \partial_{u_i} LA &= \partial_{u_i} \text{Var}(X_i \wedge u_i) - LME\ \partial_{u_i} E[X_i \wedge u_i] \\ &= \partial_{u_i} \left(E[X_i \wedge u_i]^2 - (E[X_i \wedge u_i])^2\right) - LME[1 - F_i(u_i)] \\ &= 2u_i[1 - F_i(u_i)] - 2E[X_i \wedge u_i][1 - F_i(u_i)] - LME[1 - F_i(u_i)]. \end{aligned}$$

Setting $\partial_{u_i}\ L = 0$ and solving for LME, we get for each i,

$$LME = 2(u_i - E[X_i \wedge u_i]).$$

16.3.3 Section 4.2.2. Check the Limited Cross Product Expectation Equation (4.6)

Taking conditional expectations of the left-hand side yields,

$$\text{E}(X_1 \wedge u_1)(X_2 \wedge u_2) = \text{E}\left\{(X_1 \wedge u_1)\text{E}_{X_1}(X_2 \wedge u_2)\right\},$$

where

$$\text{E}_{X_1}(X_2 \wedge u_2) = \int_0^{u_2} [1 - F(x_2|X_1)]dx_2.$$

From this,

$$\begin{array}{ll} \mathrm{E}(X_1 \wedge u_1)(X_2 \wedge u_2) & = \mathrm{E}\left\{(X_1 \wedge u_1) \int_0^{u_2} (1 - F(x_2|X_1))dx_2\right\} \\ & = \int_0^{u_1} x_1 \int_0^{u_2} [1 - F(x_2|x_1)]dx_2 \; f_1(x_1)dx_1 \\ & + \int_{u_1}^{\infty} u_1 \int_0^{u_2} [1 - F(x_2|x_1)]dx_2 \; f_1(x_1)dx_1. \end{array}$$

Because

$$F(x_2|x_1) = \int_0^{x_2} f(z|x_1)dz = \int_0^{x_2} \frac{f(x_1,z)}{f_1(x_1)}dz \quad \text{and} \quad \frac{\partial}{\partial x_1}F(x_1,x_2) = \int_0^{x_2} f(x_1,z)dz.$$

we have

$$f_1(x_1)(1 - F(x_2|x_1)) = \frac{\partial}{\partial x_1}\left[F_1(x_1) - F(x_1,x_2)\right].$$

With a change of integral,

$$\begin{array}{ll} \int_{u_1}^{\infty} u_1 \int_0^{u_2} & \{(1 - F(x_2|x_1)) \; f_1(x_1)\} \, dx_2 dx_1 \\ & = u_1 \int_0^{u_2} \int_{u_1}^{\infty} \{(1 - F(x_2|x_1)) \; f_1(x_1)\} \, dx_1 dx_2 \\ & = u_1 \int_0^{u_2} \int_{u_1}^{\infty} \left\{\frac{\partial}{\partial x_1}\left[F_1(x_1) - F(x_1,x_2)\right]\right\} dx_1 dx_2 \\ & = u_1 \int_0^{u_2} \left[1 - F_2(x_2) - (F_1(u_1) - F(u_1,x_2))\right] dx_2 \\ & = u_1 \, \mathrm{E}(X_2 \wedge u_2) - u_1 \int_0^{u_2} \left[F_1(u_1) - F(u_1,x_2)\right] dx_2. \end{array}$$

With a change of integral and integration by parts,

$$\begin{array}{ll} \int_0^{u_1} \int_0^{u_2} & x_1(1 - F(x_2|x_1))dx_2 \; f_1(x_1)dx_1 \\ & = \int_0^{u_2} \int_0^{u_1} x_1 \{(1 - F(x_2|x_1)) \; f_1(x_1)\} \, dx_1 dx_2 \\ & = \int_0^{u_2} \left[\int_0^{u_1} x_1 \left\{\frac{\partial}{\partial x_1}\left[F_1(x_1) - F(x_1,x_2)\right]\right\} dx_1\right] dx_2 \\ & = \int_0^{u_2} \left[x_1 \left[F_1(x_1) - F(x_1,x_2)\right]_0^{u_1} - \int_0^{u_1} \left[F_1(x_1) - F(x_1,x_2)\right] dx_1\right] dx_2 \\ & = \int_0^{u_2} \left[u_1 \left[F_1(u_1) - F(u_1,x_2)\right] - \int_0^{u_1} \left[F_1(x_1) - F(x_1,x_2)\right] dx_1\right] dx_2 \end{array}$$

Summing, yields the result in equation (4.6).

$$\begin{array}{l} \mathrm{E}(X_1 \wedge u_1)(X_2 \wedge u_2) \\ \quad = u_1 \, \mathrm{E}(X_2 \wedge u_2) - \int_0^{u_2} \int_0^{u_1} \left[F_1(x_1) - F(x_1,x_2)\right] dx_1 dx_2 \\ \quad = \; u_1 \, \mathrm{E}(X_2 \wedge u_2) + \; u_2 \, \mathrm{E}(X_1 \wedge u_1) - \; \int_0^{u_2} \int_0^{u_1} \left[1 - F(x_1,x_2)\right] dx_1 dx_2 \end{array}$$

16.3.4 Section 4.3.2. Quota Share with Parameter Constraints

Explicitly, we consider the risk retention problem

$$\begin{array}{lll} \text{minimize}_{\mathbf{c}} & f_0(\mathbf{c}) = \mathrm{Var}(Y_{retained}) = \mathbf{c'\Sigma c} \\ \text{subject to} & f_{in,1}(\mathbf{c}) = \mathrm{E}(Y_{reinsurer}) - RTC_{max} \\ & \qquad = (\mathrm{E}\,\mathbf{X})'(\mathbf{1} - \mathbf{c}) - RTC_{max} \leq 0 \\ & f_{in,j+1}(\mathbf{c}) = -c_j \leq 0, j = 1, \ldots, p. \end{array}$$

The Lagrangian is

$$LA(\mathbf{c}, \mathbf{LMI}) = \mathbf{c}'\mathbf{\Sigma}\mathbf{c} + LMI_1[(\mathrm{E}\ \mathbf{X})'(\mathbf{1} - \mathbf{c}) - RTC_{max}] - \sum_{j=1}^{p} LMI_{1+j}\ c_j.$$

Taking a partial derivative yields

$$\partial_{c_j} LA(\mathbf{c}, \mathbf{LMI}) \quad = 2\mathbf{c}'\mathbf{\Sigma}\mathbf{1}_j - LMI_1[\mathrm{E}(X_j)] - LMI_{j+1}, \quad j = 1, \ldots, p$$

where $\mathbf{1}_j$ is a column vector of zeros but with a one in the jth row.

Now, if $RTC_{max} > 0$ then there is at least one $c_j > 0$. Assume that $c_1 > 0$. By the *KKT* conditions, this means that the associated Lagrangian is zero, that is, $LMI_2 = 0$. This yields

$$0 = \partial_{c_1} LA(\mathbf{c}, \mathbf{LMI}) \quad = 2\mathbf{c}'\mathbf{\Sigma}\mathbf{1}_1 - LMI_1[\mathrm{E}(X_1)] \implies LMI_1 = \tfrac{2\mathbf{c}'\mathbf{\Sigma}\mathbf{1}_1}{\mathrm{E}(X_1)}$$

an explicit expression for the first Lagrange multiplier. Note that it is non-negative by the *KKT* conditions. Substituting this into the partial derivatives yields

$$0 = \partial_{c_j} LA(\mathbf{c}, \mathbf{LMI}) \quad = 2\mathbf{c}'\mathbf{\Sigma}\mathbf{1}_j - \tfrac{2\mathbf{c}'\mathbf{\Sigma}\mathbf{1}_1}{\mathrm{E}(X_1)}[\mathrm{E}(X_j)] - LMI_{j+1}, \quad j = 2, \ldots, p$$

and solving for the Lagrange multipliers shows

$$LMI_{j+1} \quad = 2\mathbf{c}'\mathbf{\Sigma}\mathbf{1}_j - \tfrac{2\mathbf{c}'\mathbf{\Sigma}\mathbf{1}_1}{\mathrm{E}(X_1)}[\mathrm{E}(X_j)], \quad j = 2, \ldots, p.$$

From the *KKT* conditions, we know that if $LMI_{j+1} > 0$ then $c_j = 0$. Equivalent to $LMI_{j+1} > 0$ is

$$\frac{\mathrm{Cov}(S(\mathbf{c}), X_j)}{\mathrm{E}(X_j)} = \frac{\mathbf{c}'\mathbf{\Sigma}\mathbf{1}_j}{\mathrm{E}(X_j)} > \frac{\mathbf{c}'\mathbf{\Sigma}\mathbf{1}_1}{\mathrm{E}(X_1)} = \frac{\mathrm{Cov}(S(\mathbf{c}), X_1)}{\mathrm{E}(X_1)}.$$

See Chapter 10 for further discussion of these risk transfer conditions.

16.3.5 Section 4.3.3. Verification of Equation (4.7)

We start with the Lagrangian

$$\begin{aligned} &LA(u_1, \ldots, u_p, LME) \\ &= \mathrm{Var}(Y_{retained}) + LME\ [\mathrm{E}(Y_{reinsurer}) - RTC_{max}]. \end{aligned}$$

From equation (4.6), we have

$$\begin{aligned} &\partial_{u_1} \mathrm{E}\{(X_1 \wedge u_1)(X_2 \wedge u_2)\} \\ &= \partial_{u_1} \{\textstyle\int_0^{u_2} \int_0^{u_1} [1 - F_1(x_1) - F_2(x_2) + F_{12}(x_1, x_2)]\ dx_1 dx_2\} \\ &= \textstyle\int_0^{u_2} \{1 - F_1(u_1) - F_2(x_2) + F_{12}(u_1, x_2)\}\ dx_2 \\ &= IT_{12}(u_1, u_2), \end{aligned}$$

where we define the asymmetric function

$$\begin{aligned} IT_{12}(a,b) &= \int_0^b \{1 - F_1(a) - F_2(z) + F_{12}(a,z)\}\ dz \\ &= \int_0^b \Pr(X_1 > a, X_2 > z)\ dz. \end{aligned}$$

Now, taking the variance of retained risks yields

$$\begin{aligned} &\partial_{u_1} \mathrm{Var}(Y_{retained}) \\ &= \partial_{u_1} \left\{ \sum_{i,j} \mathrm{E}[(X_i \wedge u_i)(X_j \wedge u_j)] - \left[\sum_{i=1}^p \mathrm{E}(X_i \wedge u_i)\right]^2 \right\} \\ &= \partial_{u_1} \mathrm{E}(X_1 \wedge u_1)^2 + 2 \sum_{j \neq 1} \{\partial_{u_1} \mathrm{E}(X_1 \wedge u_1)(X_j \wedge u_j)\} \\ &\quad -2 \left[\sum_{i=1}^p \mathrm{E}(X_i \wedge u_i)\right] \partial_{u_1} \mathrm{E}(X_1 \wedge u_1) \\ &= 2u_1[1 - F_1(u_1)] + 2 \sum_{j \neq 1} \{IT_{1j}(u_1, u_j)\} \\ &\quad -2 \left[\sum_{i=1}^p \mathrm{E}(X_i \wedge u_i)\right] [1 - F_1(u_1)] \\ &= 2 \left[u_1 - \sum_{i=1}^p \mathrm{E}(X_i \wedge u_i)\right] [1 - F_1(u_1)] + 2 \sum_{j \neq 1} \{IT_{1j}(u_1, u_j)\} \end{aligned}$$

Now, taking the derivative of the Lagrangian yields

$$\begin{aligned} &\partial_{u_1} LA(u_1, \ldots, u_p, LME) \\ &= \partial_{u_1} \mathrm{Var}(Y_{retained}) + \partial_{u_1} LME\ [\mathrm{E}(S) - \mathrm{E}(Y_{retained}) - RTC_{max}] \\ &= \partial_{u_1} \mathrm{Var}(Y_{retained}) - LME\ \partial_{u_1} \mathrm{E}(Y_{retained}) \\ &= 2 \left[u_1 - \sum_{i=1}^p \mathrm{E}(X_i \wedge u_i)\right] [1 - F_1(u_1)] + 2 \sum_{j \neq 1} \{IT_{1j}(u_1, u_j)\} \\ &\quad -LME\ \partial_{u_1} \mathrm{E}(X_1 \wedge u_1) \\ &= 2 \left[u_1 - \sum_{i=1}^p \mathrm{E}(X_i \wedge u_i) - \tfrac{LME}{2}\right] [1 - F_1(u_1)] + 2 \sum_{j \neq 1} \{IT_{1j}(u_1, u_j)\}. \end{aligned}$$

This verifies equation (4.7). Equation (4.8) is immediate from the budget constraint.

16.3.6 Section 4.3.3. Independence Case Details

In Section 4.1.2, we showed that $LME^* = 2\,(u_i^* - E(X_i \wedge u_i^*))$. Now, with the integrated distribution function, we have

$$H_{1i}(u_i) = \int_0^{u_i} F_i(z)dz = u_i - \int_0^{u_i} [1 - F_i(z)]dz = u_i - E(X_i \wedge u_i).$$

Thus, $\frac{LME}{2} = H_{1i}(u_i) \iff u_i = H_{1i}^{-1}(\frac{LME}{2})$. We may then determine LME^* as the solution of the equation

$$\begin{aligned} RTC_{max} &= \sum_{i=1}^p [E(X_i) - E(X_i \wedge u_i)] = ELoss - \sum_{i=1}^p \{u_i - H_{1i}(u_i)\} \\ &= ELoss - \sum_{i=1}^p \left\{H_{1i}^{-1}\left(\tfrac{LME}{2}\right) - \tfrac{LME}{2}\right\}. \end{aligned}$$

16.3.7 Section 4.4.1. Verification of the Convexity

Let $\mathbf{x}$ be a realization of the p risks. For each realization, define the function

$$g(\mathbf{x}, \mathbf{c}, z) = z + \tfrac{1}{1-\alpha}(\mathbf{c}'\mathbf{x} - z)_+ \ .$$

Take $\gamma \in (0,1)$ and let $(\mathbf{c}_1, z_1)$ and $(\mathbf{c}_2, z_2)$ be two sets of decision variables. Recall that $(a+b)_+ \leq a_+ + b_+$ for scalars a, b. With this, we have

$$\begin{aligned} g[\mathbf{x}, \quad & \gamma\mathbf{c}_1 + (1-\gamma)\mathbf{c}_2, \gamma z_1 + (1-\gamma)z_2] \\ & = \gamma z_1 + (1-\gamma)z_2 + \tfrac{1}{1-\alpha}(\gamma\mathbf{c}_1'\mathbf{x} + (1-\gamma)\mathbf{c}_2'\mathbf{x} - \gamma z_1 + (1-\gamma)z_2)_+ \\ & = \gamma z_1 + (1-\gamma)z_2 + \tfrac{1}{1-\alpha}(\gamma[\mathbf{c}_1'\mathbf{x} - z_1] + (1-\gamma)[\mathbf{c}_2'\mathbf{x} - z_2])_+ \\ & \leq \gamma z_1 + (1-\gamma)z_2 + \tfrac{1}{1-\alpha}(\gamma[\mathbf{c}_1'\mathbf{x} - z_1]_+ + (1-\gamma)[\mathbf{c}_2'\mathbf{x} - z_2]_+) \\ & = \gamma\ g[\mathbf{x}, \mathbf{c}_1, z_2] + (1-\gamma)\ g[\mathbf{x}, \mathbf{c}_2, z_2]. \end{aligned}$$

Thus, for each realization $\mathbf{x}$, the function $g(\mathbf{x}, \mathbf{c}, z)$ is convex in the decision variables $(\mathbf{c}, z)$. Further, we note that

$$\begin{aligned} CTE1(\mathbf{c}, z) & = z + \tfrac{1}{1-\alpha}\mathrm{E}[\mathbf{c}'\mathbf{X} - z]_+ \\ & = \mathrm{E}[g(\mathbf{X}, \mathbf{c}, z)]. \end{aligned}$$

This means that $CTE1(\mathbf{c}, z)$ is the nonnegative sum (or integral) of convex functions and so is convex (see, for example, Boyd and Vandenberghe (2004), Section 3.2.1).

16.3.8 Example 4.11. Verification of the Derivative of the Objective Function

For the objective function, we have

$$\begin{aligned} \partial_{u_j} f_0(u_1, u_2) & = \partial_{u_j} \mathrm{Var}[S(u_1, u_2)] \\ & = \partial_{u_j} \left\{ \mathrm{E}(X_1 \wedge u_1)^2 + \mathrm{E}(X_2 \wedge u_2)^2 + 2\mathrm{E}[(X_1 \wedge u_1)(X_2 \wedge u_2)] \right\} \\ & \quad -2[\mathrm{E}S(u_1, u_2)][1 - F_j(u_j)]. \end{aligned}$$

From equation (4.6), we have

$$\begin{aligned} \tfrac{\partial}{\partial u_1}\mathrm{E}(X_1 \wedge u_1)(X_2 \wedge u_2) & = \mathrm{E}(X_2 \wedge u_2) + \ u_2(1 - F_1(u_1)) - \ \int_0^{u_2} \ [1 - F(u_1, x_2)]\, dx_2 \\ & = \mathrm{E}(X_2 \wedge u_2) - \ \int_0^{u_2} \ [F_1(u_1) - F(u_1, x_2)]\, dx_2. \\ & = \int_0^{u_2} \ [1 - F_1(u_1) - F_2(x_2) + F(u_1, x_2)]\, dx_2. \end{aligned}$$

Thus,

$$\begin{aligned} & \partial_{u_1} f_0(u_1, u_2) \\ & = \partial_{u_1} \left\{ \mathrm{E}(X_1 \wedge u_1)^2 + \mathrm{E}(X_2 \wedge u_2)^2 + 2\mathrm{E}(X_1 \wedge u_1)(X_2 \wedge u_2) \right\} \\ & \quad -2[\mathrm{E}S(u_1, u_2)][1 - F_1(u_1)] \\ & = 2u_1(1 - F_1(u_1)) + 2\int_0^{u_2} \ [1 - F_1(u_1) - F_2(x_2) + F(u_1, x_2)]\, dx_2 \\ & \quad -2[\mathrm{E}S(u_1, u_2)][1 - F_1(u_1)] \\ & = 2\left\{ [u_1 - \mathrm{E}S(u_1, u_2)][1 - F_1(u_1)] + \int_0^{u_2} \ [1 - F_1(u_1) - F_2(x_2) + F(u_1, x_2)]\, dx_2 \right\} \\ & = 2\left\{ [u_1 + u_2 - \mathrm{E}S(u_1, u_2)][1 - F_1(u_1)] - \int_0^{u_2} \ [F_2(x_2) - F(u_1, x_2)]\, dx_2 \right\}. \end{aligned}$$

which establishes equation (4.10).

16.3.9 Example 4.13. Exponential Utility Functions

This is on the 'Development of the Exchange for an Exponential Utility Function'.

In the case of exponential utilities, we have $U_i'(x) = \exp(-x/\alpha_i)$. Let $Y_i = g_i(X_1, \ldots, X_n)$ be a payment after the exchange. From Borch's Theorem, we have for $i = 2, \ldots, n$

$$\begin{array}{rrl} & k_1 U_1'[w_1 - Y_1] & = k_i U_i'[w_i - Y_i] \\ \Longrightarrow & \exp[-(w_1 - Y_1)/\alpha_1] & = \frac{k_i}{k_1} \exp[-(w_i - Y_i)/\alpha_i] \\ \Longrightarrow & -(w_1 - Y_1)/\alpha_1 & = \ln \frac{k_i}{k_1} - (w_i - Y_i)/\alpha_i \\ \Longrightarrow & -\frac{\alpha_i}{\alpha_1}(w_1 - Y_1) & = Y_i + \alpha_i \ln \frac{k_i}{k_1} - w_i. \end{array}$$

Recall that the sum of total payments is $S = \sum_i X_i = \sum_i Y_i$. Summing the last line over $i = 1, \ldots, n$, we get

$$\begin{array}{rrl} & -(w_1 - Y_1) \sum_{i=1}^n \frac{\alpha_i}{\alpha_1} & = \sum_{i=1}^n \left\{ Y_i + \alpha_i \ln \frac{k_i}{k_1} - w_i \right\} \\ \Longrightarrow & (Y_1 - w_1) \frac{1}{c_1} & = S - \sum_{i=1}^n \left\{ w_i - \alpha_i \ln \frac{k_i}{k_1} \right\} \\ \Longrightarrow & Y_1 & = c_1 S + w_1 - c_1 \sum_{i=1}^n \left\{ w_i - \alpha_i \ln \frac{k_i}{k_1} \right\} \\ & & = c_1 S + P_1, \end{array}$$

as required for the first participant. The others follow through a relabeling of the indices.

16.4 Chapter Five Theory Development

16.4.1 Section 5.2.1. Verification of the Excess of Loss Distribution Function

Going back to the basics, we partition the distribution using the following four sets:

$$\begin{array}{ll} A_1 = \{X_1 > u_1, X_2 > u_2\} & A_2 = \{X_1 \le u_1, X_2 > u_2\} \\ A_3 = \{X_1 > u_1, X_2 \le u_2\} & A_4 = \{X_1 \le u_1, X_2 \le u_2\}. \end{array}$$

At a generic y,or the first set, we have

$$\begin{array}{rl} \Pr(S \le y, A_1) & = \Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \le y, X_1 > u_1, X_2 > u_2) \\ & = \Pr(u_1 + u_2 \le y, X_1 > u_1, X_2 > u_2) = I(u_1 + u_2 \le y) \Pr(A_1) \\ & = 0, \end{array}$$

because we exclude the case where $u_1 + u_2 \le y$.

For the second set,

$$\begin{aligned}
&\Pr(S \le y, A_2) \\
&= \Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \le y, X_1 \le u_1, X_2 > u_2) \\
&= \Pr(X_1 + u_2 \le y, X_1 \le u_1, X_2 > u_2) = \Pr(X_1 \le y - u_2, X_1 \le u_1, X_2 > u_2) \\
&= \Pr(X_1 \le \min(y - u_2, u_1), X_2 > u_2) = \Pr(X_1 \le y - u_2, X_2 > u_2) \\
&= \Pr(X_1 \le y - u_2) - \Pr(X_1 \le y - u_2, X_2 \le u_2) \\
&= F_1(y - u_2) - C[F_1(y - u_2), F_2(u_2)],
\end{aligned}$$

using copula notation. The third line is because $u_1 + u_2 > y$ implies $\min(y - u_2, u_1) = y - u_2$.

In the same way, for the third set we have

$$\Pr(S \le y, A_3) = \Pr(X_2 \le y - u_1, X_1 > u_1) = F_2(y - u_1) - C[F_1(u_1), F_2(y - u_1)].$$

For the fourth set,

$$\begin{aligned}
\Pr(S \le y, A_4) &= \Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \le y, X_1 \le u_1, X_2 \le u_2) \\
&= \Pr(X_1 + X_2 \le y, X_1 \le u_1, X_2 \le u_2) \\
&= \int_{-\infty}^{u_1} \int_{-\infty}^{u_2} I[x_1 + x_2 \le y] f(x_1, x_2) dx_1 dx_2 \\
&= \int_0^{F_1(u_1)} \int_0^{F_2(u_2)} I[F_1^{-1}(z_1) + F_2^{-1}(z_2) \le y] c(z_1, z_2)\ dz_2 dz_1.
\end{aligned}$$

This is sufficient for equation (5.1). For calculation purposes, we also use

$$\begin{aligned}
\Pr(S \le y, A_4) &= \Pr(X_1 + X_2 \le y, X_1 \le u_1, X_2 \le u_2) \\
&= \int_{-\infty}^{u_1} \Pr(X_2 \le y - x, X_2 \le u_2 | X_1 = x) f_1(x) dx \\
&= \int_{-\infty}^{u_1} \Pr(X_2 \le \min(y - x, u_2) | X_1 = x) f_1(x) dx \\
&= \int_{-\infty}^{u_1} C_1[F_1(x), F_2(\min(y - x, u_2)] f_1(x) dx,
\end{aligned}$$

using the relation

$$\begin{aligned}
\Pr(X_2 \le z | X_1 = x) &= \tfrac{\partial_x \Pr(X_2 \le z, X_1 \le x)}{f_1(x)} = \tfrac{\partial_x C[F_1(x), F_2(z)]}{f_1(x)} \\
&= C_1[F_1(x), F_2(z)].
\end{aligned}$$

See Appendix Chapter 13 for a discussion of derivatives and conditional distributions with copulas.

16.4.2 Section 5.2.2. Verification of the Excess of Loss Distribution Function Partial Derivative

Taking a partial derivative of the first line of equation (5.1), we have

$$\begin{aligned}
\tfrac{\partial}{\partial u_1} &\int_0^{F_1(u_1)} \int_0^{F_2(u_2)} I[F_1^{-1}(z_1) + F_2^{-1}(z_2) \le y] c(z_1, z_2)\ dz_2 dz_1 \\
&= f_1(u_1) \int_0^{F_2(u_2)} I[u_1 + F_2^{-1}(z_2) \le y]\ c[F_1(u_1), z_2]\ dz_2 \\
&= f_1(u_1) \int_0^{F_2(u_2) \wedge F_2(y - u_1)} c[F_1(u_1), z_2]\ dz_2 \\
&= f_1(u_1)\ C_1[F_1(u_1), F_2(y - u_1)],
\end{aligned}$$

with $y - u_1 < u_2$.

Taking a partial derivative of the second line of equation (5.1) yields

$$\begin{aligned}\partial_{u_1} \quad & \{F_2(y-u_1) - C[F_1(u_1), F_2(y-u_1)]\} \\ & = -f_2(y-u_1) - \partial_{u_1}\{C[F_1(u_1), F_2(y-u_1)]\} \\ & = -f_2(y-u_1) \\ & \quad -\{f_1(u_1)C_1[F_1(u_1), F_2(y-u_1)] - f_2(y-u_1)C_2[F_1(u_1), F_2(y-u_1)]\} \\ & = -f_1(u_1)C_1[F_1(u_1), F_2(y-u_1)] - f_2(y-u_1)\{1 - C_2[F_1(u_1), F_2(y-u_1)]\}.\end{aligned}$$

Adding these two results is sufficient for equation (5.2).

16.4.3 Section 5.3.1. Verification of the VaR Derivative Expressions

We assume local smoothness of $F(\boldsymbol{\theta}, y)$ in both arguments y and θ so that $constant = F[\boldsymbol{\theta}, VaR(\boldsymbol{\theta})]$ locally. Differentiating with respect to $\boldsymbol{\theta}$ yields

$$\begin{aligned}0 \quad & = \partial_{\boldsymbol{\theta}} F[\boldsymbol{\theta}, VaR(\boldsymbol{\theta})] \\ & = \partial_{\boldsymbol{\theta}} F[\boldsymbol{\theta}, y]|_{y=VaR(\boldsymbol{\theta})} + \partial_y F[\boldsymbol{\theta}, y]|_{y=VaR(\boldsymbol{\theta})} \times \partial_{\boldsymbol{\theta}} VaR(\boldsymbol{\theta}) \\ & = F_{\boldsymbol{\theta}}[\boldsymbol{\theta}, y]|_{y=VaR(\boldsymbol{\theta})} + F_y[\boldsymbol{\theta}, y]|_{y=VaR(\boldsymbol{\theta})} \times \partial_{\boldsymbol{\theta}} VaR(\boldsymbol{\theta}) \\ & = F_{\boldsymbol{\theta}}[\boldsymbol{\theta}, VaR(\boldsymbol{\theta})] + F_y[\boldsymbol{\theta}, VaR(\boldsymbol{\theta})] \times \partial_{\boldsymbol{\theta}} VaR(\boldsymbol{\theta}).\end{aligned}$$

This is sufficient for equation (5.5).

Taking partial derivatives again, for the first term, we have

$$\partial_{\boldsymbol{\theta}} F_{\boldsymbol{\theta}'}[\boldsymbol{\theta}, VaR(\boldsymbol{\theta})] = F_{\boldsymbol{\theta}\boldsymbol{\theta}'}[\boldsymbol{\theta}, VaR(\boldsymbol{\theta})] + \partial_{\boldsymbol{\theta}} VaR(\boldsymbol{\theta}) \times F_{y\boldsymbol{\theta}'}[\boldsymbol{\theta}, VaR(\boldsymbol{\theta})].$$

For the second term, we have

$$\begin{aligned}& \partial_{\boldsymbol{\theta}}\{F_y[\boldsymbol{\theta}, VaR(\boldsymbol{\theta})] \times \partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta})\} \\ & = \{\partial_{\boldsymbol{\theta}}\ F_y[\boldsymbol{\theta}, VaR(\boldsymbol{\theta})]\} \times \partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta}) + F_y[\boldsymbol{\theta}, VaR(\boldsymbol{\theta})] \times \partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta}) \\ & = \{F_{\boldsymbol{\theta}y}[\boldsymbol{\theta}, VaR(\boldsymbol{\theta})] + F_{yy}[\boldsymbol{\theta}, VaR(\boldsymbol{\theta})]\partial_{\boldsymbol{\theta}} VaR(\boldsymbol{\theta})\} \times \partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta}) \\ & \quad + F_y[\boldsymbol{\theta}, VaR(\boldsymbol{\theta})] \times \partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta}).\end{aligned}$$

Putting this together, the matrix of second derivatives of the quantile, or value at risk, is as in equation (5.6).

16.4.4 Section 5.3.1. Verification of the ES Derivative Expressions

For the auxiliary function, define $ES1_F(\boldsymbol{\theta}; y) = y + \frac{1}{1-\alpha}\{\mathrm{E}[g(X;\boldsymbol{\theta})] - \mathrm{E}[g(X;\boldsymbol{\theta}) \wedge y]\}$. Using equation (2.7), it is easy to see that $\partial_y \mathrm{E}[g(X;\boldsymbol{\theta}) \wedge y] = 1 - F(\boldsymbol{\theta}; y)$. Thus,

$$\partial_y\ ES1_F(\boldsymbol{\theta}; y) \quad = 1 - \tfrac{1}{1-\alpha}[1 - F(\boldsymbol{\theta}; y)].$$

Now, with the other set of partial derivatives

$$\partial_{\boldsymbol{\theta}}\ ES1_F(\boldsymbol{\theta};y) \quad = \tfrac{1}{1-\alpha}\left\{\partial_{\boldsymbol{\theta}}\mathrm{E}[g(X;\boldsymbol{\theta})] - \partial_{\boldsymbol{\theta}}\mathrm{E}[g(X;\boldsymbol{\theta})\wedge y]\right\},$$

we can take differentials to yield

$$\begin{aligned}
&\partial_{\boldsymbol{\theta}}\ ES_\alpha[g(\mathbf{X};\theta)]\\
&= \partial_{\boldsymbol{\theta}}\ ES1_F[\boldsymbol{\theta};VaR(\boldsymbol{\theta})]\\
&= \partial_{\boldsymbol{\theta}} ES1_F(\boldsymbol{\theta};y)|_{y=VaR(\boldsymbol{\theta})} + \partial_y ES1_F(\boldsymbol{\theta};y)|_{y=VaR(\boldsymbol{\theta})} \times \partial_{\boldsymbol{\theta}} VaR(\boldsymbol{\theta})\\
&= \tfrac{1}{1-\alpha}\left\{\partial_{\boldsymbol{\theta}}\mathrm{E}[g(X;\boldsymbol{\theta})] - \partial_{\boldsymbol{\theta}}\mathrm{E}[g(X;\boldsymbol{\theta})\wedge y]|_{y=VaR(\boldsymbol{\theta})}\right\}\\
&\qquad + \left(1 - \tfrac{1}{1-\alpha}\{1-F[\boldsymbol{\theta};VaR(\boldsymbol{\theta})]\}\right) \times \partial_{\boldsymbol{\theta}} VaR(\boldsymbol{\theta}),
\end{aligned}$$

as in equation (5.7).

Now consider partial second derivatives. First note that, because $\partial_y \mathrm{E}[g(X;\boldsymbol{\theta})\wedge y] = 1 - F(\boldsymbol{\theta};y)$, we have

$$\partial_y\partial_{\boldsymbol{\theta}}\ \mathrm{E}[g(X;\boldsymbol{\theta})\wedge y] \quad = -\partial_{\boldsymbol{\theta}} F(\boldsymbol{\theta};y).$$

With this, for the second term on the right hand side of equation (5.7), we have

$$\begin{aligned}
\partial_{\boldsymbol{\theta}} \quad & \partial_{\boldsymbol{\theta}'}\mathrm{E}[g(X;\boldsymbol{\theta})\wedge y]|_{y=VaR(\boldsymbol{\theta})}\\
&= \partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'}\mathrm{E}[g(X;\boldsymbol{\theta})\wedge y]|_{y=VaR(\boldsymbol{\theta})}\\
&\qquad + \partial_y\partial_{\boldsymbol{\theta}}\ \mathrm{E}[g(X;\boldsymbol{\theta})\wedge y]|_{y=VaR(\boldsymbol{\theta})} \times \partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta})\\
&= \partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'}\mathrm{E}[g(X;\boldsymbol{\theta})\wedge y]|_{y=VaR(\boldsymbol{\theta})}\\
&\qquad - \partial_{\boldsymbol{\theta}} F[\boldsymbol{\theta};y]|_{y=VaR(\boldsymbol{\theta})} \times \partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta}).
\end{aligned}$$

For the third term in equation (5.7),

$$\begin{aligned}
\partial_{\boldsymbol{\theta}} \quad & \left(1 - \tfrac{1}{1-\alpha}\{1-F[\boldsymbol{\theta};VaR(\boldsymbol{\theta})]\}\right) \times \partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta})\\
&= \left(1 - \tfrac{1}{1-\alpha}\{1-F[\boldsymbol{\theta};VaR(\boldsymbol{\theta})]\}\right) \times \partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta})\\
&\qquad + \left(\tfrac{1}{1-\alpha}\partial_{\boldsymbol{\theta}} F[\boldsymbol{\theta},VaR(\boldsymbol{\theta})]\right) \times \partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta}).
\end{aligned}$$

Putting these together yields

$$\begin{aligned}
&\partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'}\ ES_\alpha[g(\mathbf{X};\theta)]\\
&= \tfrac{1}{1-\alpha}\left\{\partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'}\mathrm{E}[g(X;\boldsymbol{\theta})] - \partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'}\mathrm{E}[g(X;\boldsymbol{\theta})\wedge y]|_{y=VaR(\boldsymbol{\theta})}\right.\\
&\qquad \left. - \ \partial_{\boldsymbol{\theta}} F[\boldsymbol{\theta};y]|_{y=VaR(\boldsymbol{\theta})} \times \partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta})\right\}\\
&\qquad + \left(1 - \tfrac{1}{1-\alpha}\{1-F[\boldsymbol{\theta};VaR(\boldsymbol{\theta})]\}\right) \times \partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta})\\
&\qquad + \tfrac{1}{1-\alpha}\partial_{\boldsymbol{\theta}} F[\boldsymbol{\theta},VaR(\boldsymbol{\theta})] \times \partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta}).\\
&= \tfrac{1}{1-\alpha}\left\{\partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'}\mathrm{E}[g(X;\boldsymbol{\theta})] - \partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'}\mathrm{E}[g(X;\boldsymbol{\theta})\wedge y]|_{y=VaR(\boldsymbol{\theta})}\right\}\\
&\qquad + \left(1 - \tfrac{1}{1-\alpha}\{1-F[\boldsymbol{\theta};VaR(\boldsymbol{\theta})]\}\right) \times \partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta})\\
&\qquad + \tfrac{1}{1-\alpha}\left(\partial_{\boldsymbol{\theta}} F[\boldsymbol{\theta},VaR(\boldsymbol{\theta})] - \partial_{\boldsymbol{\theta}} F[\boldsymbol{\theta};y]|_{y=VaR(\boldsymbol{\theta})}\right) \times \partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta})\\
&= \tfrac{1}{1-\alpha}\left\{\partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'}\mathrm{E}[g(X;\boldsymbol{\theta})] - \partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'}\mathrm{E}[g(X;\boldsymbol{\theta})\wedge y]|_{y=VaR(\boldsymbol{\theta})}\right\}\\
&\qquad + \left(1 - \tfrac{1}{1-\alpha}\{1-F[\boldsymbol{\theta};VaR(\boldsymbol{\theta})]\}\right) \times \partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta})\\
&\qquad + \tfrac{1}{1-\alpha}F_y[\boldsymbol{\theta},VaR(\boldsymbol{\theta})] \times \partial_{\boldsymbol{\theta}} VaR(\boldsymbol{\theta}) \times \partial_{\boldsymbol{\theta}'} VaR(\boldsymbol{\theta}),
\end{aligned}$$

where, in the last line, we recall that

$$\partial_{\boldsymbol{\theta}} F[\boldsymbol{\theta}, VaR(\boldsymbol{\theta})] = F_{\boldsymbol{\theta}}[\boldsymbol{\theta}, VaR(\boldsymbol{\theta})] + F_y[\boldsymbol{\theta}, VaR(\boldsymbol{\theta})] \times \partial_{\boldsymbol{\theta}} VaR(\boldsymbol{\theta}).$$

This is sufficient for equation (5.8).

16.4.5 Section 5.3.1. Verification of the ES Excess of Loss Derivative Expressions

$$\partial_{\boldsymbol{\theta}} \left\{ \mathrm{E}[g(X;\boldsymbol{\theta}) \wedge y] \right\} \quad = \partial_{\boldsymbol{\theta}} \left\{ \int_{-\infty}^{y} [1 - F(\boldsymbol{\theta}; z)] dz \right\} = - \int_{-\infty}^{y} \; F_{\boldsymbol{\theta}}(\boldsymbol{\theta}; z) dz.$$

For the first retention parameter, using equation (5.2), we have

$$\begin{aligned}
&\partial_{u_1} \left\{ \mathrm{E}[g(X;\boldsymbol{\theta}) \wedge y] \right\} \\
&= - \int_{-\infty}^{y} \; F_{u_1}(u_1, u_2; z) dz \\
&= \int_{-\infty}^{y} \; f_2(z - u_1) \left\{ 1 - C_2[F_1(u_1), F_2(z - u_1)] \right\} dz \\
&= F_2(y - u_1) - \int_{-\infty}^{y} \; \partial_z C[F_1(u_1), F_2(z - u_1)] dz \\
&= F_2(y - u_1) - C[F_1(u_1), F_2(y - u_1)] \\
&= \Pr(X_1 > u_1, X_2 \leq y - u_1).
\end{aligned}$$

This is sufficient for equation (5.9).

16.4.6 Section 5.5.2. Verification of the Bounds for Active Constraints

Starting with $0 \leq u_2 \leq \infty$, we have

$$\begin{aligned}
& 0 \leq AT_{1,RC}(u_1) = RC_2^{-1}[RTC_{max} - RC_1(u_1)] \leq \infty \\
\iff \quad & 0 = RC_2(\infty) \leq RTC_{max} - RC_1(u_1) \leq RC_2(0) \\
\iff \quad & -RC_2(0) \leq RC_1(u_1) - RTC_{max} \leq 0 \\
\iff \quad & RTC_{max} - RC_2(0) \leq RC_1(u_1) \leq RTC_{max}.
\end{aligned}$$

In the case that $RTC_{max} < RC_2(0)$, we can use 0 as a lower bound on $RC_1(u_1)$. This is sufficient for equation (5.14).

16.5 Chapter Six Theory Development

16.5.1 Section 6.5. Verification of Quantile Risk-Sharing Rule Properties

First recall that, for $h(\cdot)$ continuous and increasing, that $F^{-1}_{h(X)}(p) = h[F_X^{-1}(p)]$ for proportion p and a generic random variable X. That is, the quantile of a function of a random variable is the function of the quantile, assuming the transform is continuous and increasing.

Next, note that the function $S^c(u) = F_1^{-1}(u) + \cdots + F_n^{-1}(u)$ is continuous and increasing. Thus,

$$F_{S^c(U)}^{-1}(p) = S^c[F_U^{-1}(p)] = S^c[p] = \sum_i F_i^{-1}(p).$$

Thus,

$$\sum_i g_{3,i}(s) = \sum_i F_i^{-1}(p_s^c) = F_{S^c(U)}^{-1}(p_s^c) = s,$$

because $p_s^c = F_{S^c}(s)$ and the assumed continuity of the quantiles.

16.5.2 Section 6.5. Verification of Cash Back Properties

Because $g_i(s)$ is non-decreasing in s, we have

$$\begin{aligned} B_i &= [u_i - g_i(S)]_+ = [g_i(u_P) - g_i(S)]_+ \\ &= \begin{cases} 0 & u_P \le S \\ g_i(u_P) - g_i(S) & u_P > S \end{cases}. \end{aligned}$$

Note that the event $\{u_P > S\}$ does not depend on i. Thus, we can sum both sides over $i = 1, \ldots, n$ to get the result.

16.6 Chapter Seven Theory Development

16.6.1 Section 7.2.1. Verification of the VaR as a Solution of Display (7.2)

Treating this as a constrained minimization problem, we can write the Lagrangian as

$$LA(x, LMI, \boldsymbol{\theta}) = x - LMI \left(F_{g(X;\boldsymbol{\theta})}(x) - \alpha \right).$$

Here, the variables in $\boldsymbol{\theta}$ are exogenous, or auxiliary, to the main problem. Taking derivatives with respect to x and the Lagrange multiplier LMI yields

$$\begin{aligned} \partial_x LA(x, LMI, \boldsymbol{\theta}) &= 1 - LMI \; \partial_x F_{g(X;\boldsymbol{\theta})}(x) \\ \partial_{LMI} LA(x, LMI, \boldsymbol{\theta}) &= -(F_{g(X;\boldsymbol{\theta})}(x) - \alpha). \end{aligned}$$

Setting these equal to zero yields

$$\begin{aligned} LMI &= \tfrac{1}{f_{g(X;\boldsymbol{\theta})}(x^*)} \\ x^* &= F_{g(X;\boldsymbol{\theta})}^{-1}(\alpha), \end{aligned}$$

where $f_{g(X;\boldsymbol{\theta})}(\cdot)$ is the probability density function corresponding to the distribution function $F_{g(X;\boldsymbol{\theta})}(\cdot)$. From this, we see that we can calculate the value at risk as the solution of the constrained optimization problem in Display (7.2).

16.6.2 Section 7.2.2. Solution of the ES Minimization Problem

Recall from equation (2.7) that $\partial_x \, \mathrm{E}[Y \wedge x] = 1 - F(x)$. After a few calculations, we have

$$\begin{aligned} \partial_{z_0} \, ES1_F(z_0) &= \partial_{z_0} \left\{ z_0 + \tfrac{1}{1-\alpha} \left\{ \mathrm{E}[Y] - \mathrm{E}[Y \wedge z_0] \right\} \right\} \\ &= 1 - \tfrac{1}{1-\alpha} \left\{ 1 - F(z_0) \right\}. \end{aligned}$$

Setting this to zero yields the best value of z_0 to be $F^{-1}(\alpha)$, as claimed. Moreover,

$$\begin{aligned} ES1_F[F^{-1}(\alpha)] &= F^{-1}(\alpha) + \tfrac{1}{1-\alpha} \left\{ \mathrm{E}[Y] - \mathrm{E}[Y \wedge F^{-1}(\alpha)] \right\} \\ &= ES \end{aligned}$$

from equation (2.3).

16.6.3 Section 7.3.1. Confirm Display (7.15)

To ease notation, for the moment drop the superscript s used to denote the rescaling. Now, define the Lagrangian

$$LA \quad = \textstyle\sum_{j=1}^{p} \, \mathrm{Var}(X_j \wedge u_j) + LME[\sum_{j=1}^{p} \, \{\mathrm{E}(X_j) - \mathrm{E}(X_j \wedge u_j)] - RTC_{max}\}.$$

We first recall the relationships $\mathrm{E}(X \wedge u) = \int_0^u \, [1 - F(x)]dx$ and $\mathrm{E}(X \wedge u)^2 = 2 \int_0^u \, x[1 - F(x)]dx$.

Taking a partial derivative with respect to LME and setting this equal to zero simply means that the constraint, $\mathrm{E}(Y_{reinsurer}) = RTC_{max}$, is enforced and we have to choose the limits u_j to satisfy this constraint. Moreover, taking the partial derivative with respect to each limit u_j yields

$$\begin{aligned} \partial_{u_j} LA &= \partial_{u_j} \, \mathrm{Var}(X_j \wedge u_j) - LME \, \partial_{u_j} \, E[X_j \wedge u_j] \\ &= \partial_{u_j} \left(E[X_j \wedge u_j]^2 - (E[X_j \wedge u_j])^2 \right) - LME[1 - F_j(u_j)] \\ &= 2u_j[1 - F_j(u_j)] - 2E[X_j \wedge u_j][1 - F_j(u_j)] - LME[1 - F_j(u_j)] \\ &= (2u_j - 2E[X_j \wedge u_j] - LME) \, [1 - F_j(u_j)]. \end{aligned}$$

Setting this partial derivative equal to zero is sufficient for Display (7.15).

16.6.4 Section 7.3.1. Confirm Equation (7.16)

As before, to ease notation, for the moment drop the superscript s used to denote the rescaling. From Display (7.15), we have that $LME = 2 \, (u_j - E(X_j \wedge u_j))$. Now, with the integrated distribution function, we have

$$H_j(u_j) = \int_0^{u_j} F_j(z)dz = u_j - \int_0^{u_j} [1 - F_j(z)]dz = u_j - E(X_j \wedge u_j).$$

Thus, $\frac{LME}{2} = H_j(u_j) \iff u_j = H_j^{-1}(\frac{LME}{2})$. We may then determine LME as the solution of the equation

$$\begin{array}{rl} RTC_{max} & = \sum_{i=1}^p [E(X_j) - E(X_j \wedge u_j)] = \sum_{j=1}^p \mathrm{E}(X_j) - \sum_{i=1}^p \{u_j - H_j(u_j)\} \\ & = \sum_{j=1}^p \mathrm{E}(X_j) - \sum_{i=1}^p \left\{H_j^{-1}\left(\frac{LME}{2}\right) - \frac{LME}{2}\right\}, \end{array}$$

that suffices for equation (7.16).

16.6.5 Section 7.3.1. Confirm the Scaling Factor to Maintain the Budget Constraint

Maintain the Budget Constraint

Multiplying the average scaling factor by the rescaled maximal risk transfer cost yields

$$\begin{array}{rl} \overline{sd} \times RTC_{max}^s = & \left(\frac{1}{p}\sum_{j=1}^p sd_j\right)\left(\sum_{j=1}^p \mathrm{E}(X_j^s) - \mathrm{E}(X_j^s \wedge u_j^s)\right) \\ & = \left(\sum_{j=1}^p sd_j\right)(\mathrm{E}(X_1^s) - \mathrm{E}(X_1^s \wedge u_1^s)) \\ & = \left(\sum_{j=1}^p sd_j\left[\mathrm{E}(X_j^s) - \mathrm{E}(X_j^s \wedge u_j^s)\right]\right) \\ & = \sum_{j=1}^p \left[\mathrm{E}(X_j) - \mathrm{E}(X_j \wedge u_j^{rs})\right] \\ & = RTC_{max}. \end{array}$$

16.7 Chapter Nine Theory Development

16.7.1 Example 9.7. Check the VaR Sensitivity

For this problem, one minimizes over x and so we can take $\theta_k = a$ to be an auxiliary variable. For partial derivatives, we use

- $f_{0\theta_k}(x, \boldsymbol{\theta}) = \partial_{\theta_k} x = 0$,
- $f_{eq,\theta_k}(x, \boldsymbol{\theta}) = \partial_{\theta_k} f_{eq}(x, \boldsymbol{\theta}) = -\partial_{\theta_k}[F_{g(X;\boldsymbol{\theta})}(x) - \alpha] = -\partial_{\theta_k} F_{g(X;\boldsymbol{\theta})}(x)$, and
- $f_{eq,1}(x, \boldsymbol{\theta}) = -\partial_x[F_{g(X;\boldsymbol{\theta})}(x) - \alpha] = -f_{g(X;\boldsymbol{\theta})}(x)$.

Thus, with (9.3), we have for any $k = 1, \ldots, p$

$$\begin{array}{rl} \partial_{\theta_k} VaR_\alpha(g(X;\boldsymbol{\theta})) & = \partial_{\theta_k} f_0(x^*, \theta_k) \\ & = f_{0\theta_k}(x^*, \theta_k) + LME^* \ f_{eq,\theta_k}(x^*, \theta_k) \\ & = 0 + LME^* \ f_{eq,\cdot a}(x^*(a), a) \\ & = \frac{-1}{f_{g(X;\boldsymbol{\theta})}[F^{-1}_{g(X;\boldsymbol{\theta})}(\alpha)]} \ \partial_{\theta_k} F_{g(X;\boldsymbol{\theta})}(x)\big|_{x=F^{-1}_{g(X;\boldsymbol{\theta})}(\alpha)}, \end{array}$$

that confirms equation (9.10).

16.7.2 Section 9.4.1. Verification of Asset Allocation Sensitivities

The Lagrangian is

$$\begin{aligned} LA &= f_0(\mathbf{c}, \mathbf{\Sigma}) + LME_1 \ f_{con}(\mathbf{c}) \\ &= -(\mathbf{c}'\boldsymbol{\mu} - \lambda_{Port} \ \mathbf{c}'\mathbf{\Sigma}\mathbf{c}) + LME_1(\mathbf{c}'\mathbf{1} - 1). \end{aligned}$$

To determine the optimal choice variables $\mathbf{c}$, we take partial derivatives

$$\begin{aligned} \partial_{\mathbf{c}} LA &= -\boldsymbol{\mu} + 2\lambda_{Port} \ \mathbf{\Sigma}\mathbf{c} + LME_1 \ \mathbf{1} = \mathbf{0} \\ \Longrightarrow \quad \mathbf{c}^* &= \tfrac{1}{2\lambda_{Port}} \ \mathbf{\Sigma}^{-1}(\boldsymbol{\mu} - LME_1^* \ \mathbf{1}). \end{aligned}$$

In addition, from the constraint $\mathbf{c}'\mathbf{1} = 1$, we have

$$\begin{aligned} & 1 = \mathbf{c}^{*\prime}\mathbf{1} = \tfrac{1}{2\lambda_{Port}}(\boldsymbol{\mu} - LME_1^* \ \mathbf{1})'\mathbf{\Sigma}^{-1}\mathbf{1} \\ \Longrightarrow \quad & LME_1^* = [\boldsymbol{\mu}'\mathbf{\Sigma}^{-1}\mathbf{1} - 2\lambda_{Port}]/\mathbf{1}'\mathbf{\Sigma}^{-1}\mathbf{1}. \end{aligned}$$

Taking a partial derivative with respect to μ is sufficient for equation (9.12).

In the same way, taking a partial derivative with respect to μ of the optimal decision variables yields

$$\partial_\mu \mathbf{c}^* = \tfrac{1}{2\lambda_{Port}} \ \mathbf{\Sigma}^{-1}(\partial_\mu \boldsymbol{\mu} - \mathbf{1} \ \partial_\mu LME_1^*(\mu)),$$

which is sufficient for equation (9.11).

16.8 Chapter Eleven Theory Development

16.8.1 Example 11.1. Univariate Risk Retention Conditions

This if for the 'Verification that the Optimal d=0'.

As with equation (3.10), the associated Lagrangian is

$$\begin{aligned} LA(\boldsymbol{\theta}, \mathbf{LMI}) = ES_\alpha[g(X); \boldsymbol{\theta}] \quad & + LMI_1[RTC(\boldsymbol{\theta}) - RTC_{max}] \\ & - LMI_2 \ d + LMI_3(d - u). \end{aligned}$$

Taking partial derivatives, we have

$$\begin{aligned} \partial_d LA(\boldsymbol{\theta}, \mathbf{LMI}) &= \partial_d ES_\alpha[g(X); \boldsymbol{\theta}] + LMI_1 \ \partial_d RTC(\boldsymbol{\theta}) + LMI_3 - LMI_2 \\ \partial_u LA(\boldsymbol{\theta}, \mathbf{LMI}) &= \partial_u ES_\alpha[g(X); \boldsymbol{\theta}] + LMI_1 \ \partial_u RTC(\boldsymbol{\theta}) - LMI_3, \end{aligned}$$

where explicit expressions for $\partial_\theta ES_\alpha(g(X)$ and $\partial_\theta RTC(\boldsymbol{\theta})$ are in Table 2.2. Henceforth, we evaluate expressions at the optimum and, for simplicity, drop the asterisk (*) superscript on $\boldsymbol{\theta}$ and **LMI**.

From equation (2.7), we can express the risk transfer cost as

$$\begin{aligned} RTC(d, c=1, u) &= \mathrm{E}(X) - \mathrm{E}[g(X; d, c=1, u])] \\ &= \mathrm{E}(X) - \int_d^u [1 - F(x)]dx \\ &= \int_0^d [1 - F(x)]dx + \int_u^\infty [1 - F(x)]dx. \end{aligned}$$

Now, suppose that $LMI_1 = 0$ so that, from the second partial derivative at the optimum, we have

$$\partial_u LA\,(\boldsymbol{\theta}, \mathbf{LMI}) = \partial_u ES_\alpha[g(X); \boldsymbol{\theta}] - LMI_3 < 0,$$

using Table 2.2. This violates the KKT conditions, so $LMI_1 > 0$.

Because $LMI_1 > 0$, the requirement that $RTC(\boldsymbol{\theta}) = RTC_{max}$ is binding. Further, because $RTC(d, c=1, u) = RTC_{max} < E(X)$, this means that $u > d$ (because $RTC(d, c, u = d) = \mathrm{E}(X)$). Thus, from the second KKT condition, we have $LMI_3 = 0$.

Because $u > d$ and with the KKT conditions, we have $\partial_u LA\,(\boldsymbol{\theta}, LMI_1) = 0$. So, we can solve for LMI_1 as

$$\begin{aligned} 0 &= \partial_u LA\,(\boldsymbol{\theta}, \mathbf{LMI}) = \partial_u ES_\alpha[g(X); \boldsymbol{\theta}] - LMI_1[1 - F(u)] \\ \Rightarrow \quad LMI_1 &= \tfrac{\partial_u ES_\alpha[g(X); \boldsymbol{\theta}]}{1 - F(u)}. \end{aligned}$$

From Table 2.2,

$$LMI_1 = \begin{cases} \frac{1}{1-\alpha} & F_\alpha^{-1} < d \\ \frac{1}{1-\alpha} & d \le F_\alpha^{-1} < u \\ \frac{1}{1-F(u)} & u \le F_\alpha^{-1}. \end{cases}$$

Thus,

$$\begin{aligned} \partial_d LA\,(\boldsymbol{\theta}, \mathbf{LMI}) &= -\tfrac{1-F(d)}{1-\alpha} + LMI_1[1 - F(d)] - LMI_2 \\ &= -LMI_2 + \begin{cases} -\frac{1-F(d)}{1-\alpha} + \frac{1}{1-\alpha}[1 - F(d)] & F_\alpha^{-1} < d \\ -1 + \frac{1}{1-\alpha}[1 - F(d)] & d \le F_\alpha^{-1} < u \\ -1 + \frac{1}{1-F(u)}[1 - F(d)] & u \le F_\alpha^{-1}. \end{cases} \\ &= -LMI_2 + \begin{cases} 0 & F_\alpha^{-1} < d \\ \frac{\alpha - F(d)}{1-\alpha} & d \le F_\alpha^{-1} < u \\ \frac{F(u) - F(d)}{1-F(u)} & u \le F_\alpha^{-1}. \end{cases} \end{aligned}$$

From the KKT conditions, at the optimum we have $\partial_d LA\,(\boldsymbol{\theta}, LMI_1) = 0$. Thus, $LMI_2 > 0$ except when $d > F_\alpha^{-1}$, a case ruled out by the problem set-up where $d \le d_{max} < F_\alpha^{-1}$, so $\partial_d LA\,(\boldsymbol{\theta}, LMI_1) > 0$. Because $LMI_2 > 0$, from the second KKT condition we have a binding constraint and so the optimal $d = 0$, as claimed.

16.8.2 Section 11.2. Proof the Single Risk Retention Results

From equation (3.10), the associated Lagrangian is

$$\begin{aligned} LA\left(\boldsymbol{\theta}, \mathbf{LMI}\right) &= RM[g(X); \boldsymbol{\theta}] + LMI_1(RTC(\boldsymbol{\theta}) - RTC_{max}) + LMI_2(c-1) \\ &\quad -LMI_3 d - LMI_4 c + LMI_5(d-u). \end{aligned}$$

Taking partial derivatives, we have

$$\begin{aligned} \partial_d LA\left(\boldsymbol{\theta}, \mathbf{LMI}\right) &= \partial_d RM[g(X); \boldsymbol{\theta}] + LMI_1 \ \partial_d RTC(\boldsymbol{\theta}) - LMI_3 + LMI_5 \\ \partial_c LA\left(\boldsymbol{\theta}, \mathbf{LMI}\right) &= \partial_c RM[g(X); \boldsymbol{\theta}] + LMI_1 \ \partial_c RTC(\boldsymbol{\theta}) - LMI_4 \\ \partial_u LA\left(\boldsymbol{\theta}, \mathbf{LMI}\right) &= \partial_u RM[g(X); \boldsymbol{\theta}] + LMI_1 \ \partial_u RTC(\boldsymbol{\theta}) - LMI_5. \end{aligned}$$

Henceforth, we evaluate expressions at the optimum and, for this proof, drop the asterisk (*) suffix on $\boldsymbol{\theta}$ and LMI.

The assumption $RTC_{max} < RTC(d, c, u = d)$ means that $u > d$. Thus, from the KKT conditions, we have $LMI_5 = 0$. With this and the third expression in Display (16.8.2), we have

$$0 = \partial_u LA\left(\boldsymbol{\theta}, \mathbf{LMI}\right) = \partial_u RTC(\boldsymbol{\theta})(LMI_1 - RM_u^2) \Rightarrow LMI_1 = RM_u^2,$$

with our assumption that $\partial_u RTC(\boldsymbol{\theta}) < 0$.

From the first two expressions in Display (16.8.2), we have

$$\begin{aligned} \partial_d LA\left(\boldsymbol{\theta}, \mathbf{LMI}\right) &= (-RM_d^2 + RM_u^2) \ \partial_d RTC(\boldsymbol{\theta}) - LMI_3 \\ \partial_c LA\left(\boldsymbol{\theta}, \mathbf{LMI}\right) &= (-RM_c^2 + RM_u^2) \ \partial_c RTC(\boldsymbol{\theta}) + LMI_2 - LMI_4. \end{aligned}$$

As in equation (11.3), assume that $RM_u^2 > RM_d^2$. Then, with $\partial_d LA\left(\boldsymbol{\theta}, \mathbf{LMI}\right) = 0$, we have

$$LMI_3 = (RM_u^2 - RM_d^2) \ \partial_d RTC(\boldsymbol{\theta}) > 0$$

By the KKT conditions, this means that $d = 0$, the desired result.

Next, as in equation (11.4), assume that $RM_u^2 > RM_c^2$. Then, with $\partial_c LA\left(\boldsymbol{\theta}, \mathbf{LMI}\right) = 0$, we have

$$LMI_2 = (RM_c^2 - RM_u^2) \ \partial_c RTC(\boldsymbol{\theta}) + LMI_4 > 0$$

By the KKT conditions, this means that $c = 1$, the desired result.

16.8.3 Example 11.2. Multivariate Excess of Loss with Variance as a Risk Measure.

This is to 'Confirm the RM^2'

Start with the partial derivative of the variance,

$$\begin{aligned}\partial_{u_j} RM[S(\mathbf{u})] &= \partial_{u_j} \tfrac{1}{2}\mathrm{Var}[S(\mathbf{u})] \\ &= \mathrm{E}[S(\mathbf{u})I(X_j > u_j)] - \mathrm{E}[S(\mathbf{u})]\{\mathrm{E}[I(X_j > u_j)]\} \\ &= \mathrm{E}[S(\mathbf{u})I(X_j > u_j)] - \mathrm{E}[S(\mathbf{u})]\Pr(X_j > u_j).\end{aligned}$$

For simplicity, we use the fair risk transfer cost $RTC(\mathbf{u}) = \mathrm{E}[S(\infty)] - \mathrm{E}[S(\mathbf{u})]$. Thus, $\partial_{u_j} RTC(\mathbf{u}) = -\partial_{u_j}\mathrm{E}[X_j \wedge u_j] = -\Pr(X_j > u_j)$. From this, we have

$$\begin{aligned}RM_j^2 &= -\frac{\partial_{u_j} RM[S(\mathbf{u})]}{\partial_{u_j} RTC(\mathbf{u})} = \frac{\mathrm{E}[S(\mathbf{u})I(X_j>u_j)]-\mathrm{E}[S(\mathbf{u})]\Pr(X_j>u_j)}{\Pr(X_j>u_j)} \\ &= \frac{\mathrm{E}[S(\mathbf{u})I(X_j>u_j)]}{\Pr(X_j>u_j)} - \mathrm{E}[S(\mathbf{u})] \\ &= \mathrm{E}[S(\mathbf{u})|X_j > u_j] - \mathrm{E}[S(\mathbf{u})] \\ &= \textstyle\sum_{i=1}^{p} \{\mathrm{E}[X_i \wedge u_i|X_j > u_j] - \mathrm{E}[X_i \wedge u_i]\} \\ &= u_j - \mathrm{E}[X_j \wedge u_j] + \textstyle\sum_{i\neq j}^{p} \{\mathrm{E}[X_i \wedge u_i|X_j > u_j] - \mathrm{E}[X_i \wedge u_i]\},\end{aligned}$$

because $E[X_i \wedge u_i|X_i > u_i] = u_i$. This confirms equation (11.9).

16.8.4 Example 11.3. Bivariate Excess of Loss with VaR as a Risk Measure

This is to 'Check the VaR Balance'.

Assuming fair risk transfer cost,

$$RTC(u_1, u_2) = \mathrm{E}[S(\infty, \infty)] - \mathrm{E}[S(u_1, u_2)] = \mathrm{E}(X_1) + \mathrm{E}(X_2) - \mathrm{E}(X_1 \wedge u_1) - \mathrm{E}(X_2 \wedge u_2).$$

With this, we have

$$\partial_{u_1}\ RTC(u_1, u_2) = -\partial_{u_1}\mathrm{E}(X_1 \wedge u_1) = -[1 - F_1(u_1)],$$

and similarly for u_2. With this, the risk measure relative marginal change is

$$RM_i^2 = \frac{\partial_{u_i} RM[S(u_1, u_2)]}{1 - F_i(u_i)}.$$

First consider the value at risk for the risk measure so $RM[S(u_1, u_2)] = F_S^{-1}(\alpha)$. From equation (5.5), the quantile sensitivity is

$$\partial_{u_i}\ F_S^{-1}(\alpha) = -\frac{\partial_{u_i} F_S\left[F_S^{-1}(\alpha)\right]}{f_S\left[F_S^{-1}(\alpha)\right]}.$$

So, if the optimal $u_1^* > 0$ and $u_2^* > 0$, by the balance among retention parameters (11.8), we have

$$\frac{\partial_{u_1} F_S\left[F_S^{-1}(\alpha)\right]\big|_{u_1=u_1^*}}{1-F_1(u_1^*)} = \frac{\partial_{u_2} F_S\left[F_S^{-1}(\alpha)\right]\big|_{u_2=u_2^*}}{1-F_2(u_2^*)},$$

assuming the density f_S at the quantile $F_S^{-1}(\alpha)$ is positive (that cancels out of the expression).

From equation (5.2), we have when $u_1^* + u_2^* > z_0^*$ that

$$\begin{aligned}&\partial_{u_1} \Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \leq z_0^*)|_{u_1=u_1^*, u_2=u_2^*}\\ &\quad = -I(u_1 < z_0^*) f_2(z_0^* - u_1^*)\left[1 - C_1(F_2(z_0^* - u_1^*), F_1(u_1^*))\right].\end{aligned}$$

Here, C_1 is a copula derivative defined in Appendix Section 14.1. Note that when $u_1^* + u_2^* \leq z_0^*$, the probability must equal one resulting in no variation in the choice of parameters.

16.8.5 Example 11.5. Multivariate Deductible with Variance as a Risk Measure.

This is to 'Confirm the RM^2 for Example 11.5.'

This yields an expression for the partial derivative of the variance,

$$\begin{aligned}\partial_{d_j} RM[S(\mathbf{d})] &= \partial_{u_j} \tfrac{1}{2}\mathrm{Var}[S(\mathbf{u})]\\ &= -\mathrm{E}[\{S(\boldsymbol{\infty}) - S(\mathbf{d})\} I(X_j > d_j)] - \mathrm{E}[\{S(\boldsymbol{\infty}) - S(\mathbf{d})\}\{-\mathrm{E}[I(X_j > d_j)]\}\\ &= -\mathrm{E}[S(\mathbf{d}) I(X_j > d_j)] + \mathrm{E}[S(\mathbf{d})]\Pr[X_j > d_j].\end{aligned}$$

For simplicity, we use the fair risk transfer cost $RTC(\mathbf{d}) = \mathrm{E}[S(\mathbf{d})]$. Thus, $\partial_{d_j} RTC(\mathbf{d}) = \partial_{d_j}\mathrm{E}[X_j \wedge u_d] = \Pr(X_j > d_j)$. From this, we have

$$\begin{aligned}RM_j^2(\mathbf{d}) &= -\tfrac{\partial_{d_j} RM[S(\mathbf{d})]}{\partial_{d_j} RTC(\mathbf{d})} = \tfrac{\mathrm{E}[S(\mathbf{d}) I(X_j > d_j)] - \mathrm{E}[S(\mathbf{d})]\Pr(X_j > d_j)}{\Pr(X_j > d_j)}\\ &= \mathrm{E}[S(\mathbf{d})|X_j > u_j] - \mathrm{E}[S(\mathbf{d})]\\ &= d_j - \mathrm{E}[X_i \wedge d_j] + \textstyle\sum_{i\neq j}^{p}\left\{\mathrm{E}[X_i \wedge d_i | X_j > d_j] - \mathrm{E}[X_i \wedge d_i]\right\},\end{aligned}$$

as in [Example 10.3.1]. This confirms equation (11.10).

16.8.6 Section 11.4.1. Confirm the RM^2 for the Simulation Variance Risk Measure

The risk transfer cost is

$$RTC_R(\boldsymbol{\theta}) = \frac{1}{R}\sum_{r=1}^{R}\left\{\left[\sum_{j=1}^{p} X_{rj}\right] - g(\mathbf{X}_r; \boldsymbol{\theta}_j)\right\}.$$

Taking a partial derivative yields

$$\begin{aligned}\partial_{\theta_j} RTC_R(\boldsymbol{\theta}) &= \tfrac{1}{R}\sum_{r=1}^{R} \partial_{\theta_j}\left\{\left[\sum_{j=1}^{p} X_{rj}\right] - g(\mathbf{X}_r;\boldsymbol{\theta}_j)\right\} \\ &= \tfrac{-1}{R}\sum_{r=1}^{R} g_{rj} = -\bar{g}_j.\end{aligned}$$

Further, taking a partial derivative of the variance of retained risks yields

$$\begin{aligned}\partial_{\theta_j}\ RM_{var}(\boldsymbol{\theta}) &= \tfrac{1}{R}\sum_{r=1}^{R} g(\mathbf{X}_r;\boldsymbol{\theta}) g_{rj} - \left\{\overline{g(\mathbf{X}_R;\boldsymbol{\theta})}\right\} \partial_{\theta_j}\left\{\overline{g(\mathbf{X}_R;\boldsymbol{\theta})}\right\} \\ &= \tfrac{1}{R}\sum_{r=1}^{R} g(\mathbf{X}_r;\boldsymbol{\theta}) g_{rj} - \left\{\overline{g(\mathbf{X}_R;\boldsymbol{\theta})}\right\} \bar{g}_j.\end{aligned}$$

This yields

$$\begin{aligned}RM^2_{var,j}(\boldsymbol{\theta}) &= -\tfrac{\partial_{\theta_j}\ RM_{var}(\boldsymbol{\theta})}{\partial_{\theta_j} RTC_R(\boldsymbol{\theta})} \\ &= \tfrac{-1}{\bar{g}_j}\left(\tfrac{1}{R}\sum_{r=1}^{R} g(\mathbf{X}_r;\boldsymbol{\theta}) g_{rj} - \left\{\overline{g(\mathbf{X}_R;\boldsymbol{\theta})}\right\}\bar{g}_j\right) \\ &= \overline{g(\mathbf{X}_R;\boldsymbol{\theta})} - \tfrac{\mathrm{E}_R[g(\mathbf{X}_R;\boldsymbol{\theta}) g_{Rj}]}{\bar{g}_j},\end{aligned}$$

which confirms equation (11.12).

16.8.7 Section 11.4.2. Confirm the Partial Derivative of RTC

We have

$$\begin{aligned}\partial_{\theta_j}\ RTC_R(\boldsymbol{\theta}) &= \partial_{\theta_j}\mathrm{E}_R\left\{\left(\sum_{j=1}^{p} X_j\right) - g(\mathbf{X}_R;\boldsymbol{\theta})\right\} \\ &= -\partial_{\theta_j}\int \mathrm{E}_R\left\{g(\mathbf{X};\boldsymbol{\theta}) + bz\right\}\ k(z)dz \\ &= -\int \mathrm{E}_R\left\{g_{rj}\right\}\ k(z)dz \\ &= -\bar{g}_j,\end{aligned}$$

as in the case of unweighted risk transfer costs.

16.8.8 Section 11.4.2. Confirm the Expected Shortfall RM^2

From Appendix Section 10.5.3, the ES sensitivity can be expressed as

$$\begin{aligned}\partial_{\theta_j}\ ES_{Rk}[g(\mathbf{X};\theta)] &= \tfrac{1}{R(1-\alpha)}\sum_{r=1}^{R}\left\{1 - K\left(\tfrac{VaR_{Rk} - g(\mathbf{X}_r;\boldsymbol{\theta})}{b}\right)\right\}\ g_{rj} \\ &\quad + [1 - \tfrac{1}{1-\alpha}\{1 - F_{Rk}(VaR_{Rk})\}] \times \partial_{\theta_j} VaR_{Rk}(\boldsymbol{\theta}).\end{aligned}$$

Assuming no discreteness at the quantile and using the notation $\tilde{k}_{r\theta} = 1 - K\left([VaR_{Rk} - g(\mathbf{X}_r;\boldsymbol{\theta})]/b\right)$, we have

$$\begin{aligned}\partial_{\theta_j}\ ES_{Rk}[g(\mathbf{X};\theta)] &= \tfrac{1}{R(1-\alpha)}\sum_{r=1}^{R}\left\{\tilde{k}_{r\theta}\ g_{rj}\right\} \\ &= \tfrac{1}{1-\alpha}\mathrm{E}_R\left\{\tilde{k}_{R\theta}\ g_{Rj}\right\}.\end{aligned}$$

This and $\partial_{\theta_j}\ RTC_R(\boldsymbol{\theta}) = -\bar{g}_j$ are sufficient for the result.

16.8.9 Section 11.5.3.1. Confirm the Justification of *KKT* Conditions for a Single Equality Contstraint

Sketch of Proof:

Consider a point $\mathbf{z}$ that is in the feasible set. Now, suppose that it is not optimal so that $\mathbf{z} \neq \mathbf{z}^*$. Then we can find many directions s so that $f_0(\mathbf{z}+s) < f_0(\mathbf{z})$. Choose a direction s that is small and use Taylor series expansion to write

$$f_0(\mathbf{z}+s) \approx f_0(\mathbf{z}) + \nabla f_0(\mathbf{z})'s \Rightarrow \nabla f_0(\mathbf{z})'s < 0.$$

Restrict considerations to those directions s so that $\mathbf{z}+s$ is also in the feasible set. In the same way, with a Taylor series expansion, we have

$$0 = f_{con,1}(\mathbf{z}+s) \approx f_{con,1}(\mathbf{z}) + \nabla f_{con,1}(\mathbf{z})'s \Rightarrow \nabla f_{con,1}(\mathbf{z})'s = 0.$$

Is such a restriction sensible? Yes, for example, we might choose

$$s = -\epsilon \left(\mathbf{I} - \frac{\nabla f_{con,1}(\mathbf{z}) \nabla f'_{con,1}(\mathbf{z})}{||\nabla f_{con,1}(\mathbf{z})||^2} \right) \nabla f'_0(\mathbf{z}),$$

for a small positive constant ϵ. With this choice, one can check that we still have $\nabla f_{con,1}(\mathbf{z})'s = 0$ and $\nabla f_0(\mathbf{z})'s < 0$.

This establishes that a given feasible point $\mathbf{z}$ is not optimal if we can find a small step s that both retains feasibility and decreases the objective function f to first order.

Thus, we cannot find a constant LME_1 such that $\nabla f_0(\mathbf{z}) + LME_1 \nabla f_{con,1}(\mathbf{z}) = 0$ (that is, the gradients are proportional to one another). If this were true, then

$$\left[\nabla f_0(\mathbf{z}) + LME_1 \nabla f_{con,1}(\mathbf{z})\right]' s = 0,$$

contradicting prior arguments. That is, if we can find a constant, say LME_1 so that $\nabla f_0(\mathbf{z}) + LME_1 \nabla f_{con,1}(\mathbf{z}) = 0$, then this point is optimal $\mathbf{z} = \mathbf{z}^*$.

16.8.10 Section 11.5.3.2. Confirm the Justification of *KKT* Conditions for a Single Inequality Contstraint

Sketch of Proof:

As in the Section 11.5.3 sketch of the proof, we first seek conditions so that a small step s decreases the objective function while retaining feasibility. As before, for decreasing the objective function, we have $f_0(\mathbf{z}+s) < f_0(\mathbf{z})$. With a Taylor series expansion, this means

$$f_0(\mathbf{z}+s) \approx f_0(\mathbf{z}) + \nabla f_0(\mathbf{z})'s \Rightarrow \nabla f_0(\mathbf{z})'s < 0.$$

For feasibility, we now require that $f_{con,1}(\mathbf{z}) \leq 0$ and $f_{con,1}(\mathbf{z}+s) \leq 0$. Using a Taylor series expansion, we have

$$f_{con,1}(\mathbf{z}+s) \approx f_{con,1}(\mathbf{z}) + \nabla f_{con,1}(\mathbf{z})'s.$$

Taken together, we seek

$$\nabla f_0(\mathbf{z})'s < 0, \quad f_{con,1}(\mathbf{z}) + \nabla f_{con,1}(\mathbf{z})'s \leq 0. \tag{16.1}$$

Case 1. Begin by assuming that the constraint is inactive so that $f_{con,1}(\mathbf{z}) < 0$. From this, any direction s that is sufficiently small satisfies the feasibility constraint

$$f_{con,1}(\mathbf{z}) + \nabla f_{con,1}(\mathbf{z})'s \leq 0.$$

Thus, if $\mathbf{z}$ is an optimal point then $\nabla f_0(\mathbf{z})' = 0$. Otherwise, the point could be improved upon and would not be optimal. A proportionality constant is not required and so we may define $LMI = 0$ without imposing any restrictions.

Case 2. Now assume that the constraint is active at $\mathbf{z}$ so that $f_{con,1}(\mathbf{z}) = 0$. Then, equation (16.1) becomes

$$\nabla f_0(\mathbf{z})'\ s < 0, \quad \nabla f_{con,1}(\mathbf{z})'\ s \leq 0. \tag{16.2}$$

Suppose that we cannot find a s such that equation (16.2) holds (assuming an active constraint). Then, as in the Section 11.5.3 proof sketch, $\mathbf{z}$ is an optimal point, $\mathbf{z} = \mathbf{z}^*$, and this can only occur when $\nabla f_0(\mathbf{z})$ is proportional to $\nabla f_{con,1}(\mathbf{z})$. Writing this proportionality constraint as

$$\nabla f_0(\mathbf{z}) + LMI\ \nabla f_{con,1}(\mathbf{z}) = 0,$$

for some constant LMI, we see that we now also require that $LMI > 0$. If $LMI < 0$, then we could choose a direction s so that both the objective function and the constraint function would decrease without bound (both $\nabla f_0(\mathbf{z})'s < 0$ and $\nabla f_{con,1}(\mathbf{z})'\ s < 0$).

Using these constructions for LMI, we can define the Lagrangian as in equation (11.18). With this,

- if the constraint is inactive, then $LMI = 0$ and $\nabla LA(\mathbf{z}) = \nabla f_0(\mathbf{z}) = 0$ at the optimum. Thus, the KKT conditions are satisfied.
- if the constraint is active, then $f_{con,1}(\mathbf{z}) = 0$, $\nabla LA(\mathbf{z}) = 0$, and $LMI > 0$. Thus, the KKT conditions are satisfied.

This is sufficient for the result.

16.9 Chapter Twelve Theory Development

16.9.1 Section 12.3.2. Development of the Quantile Sensitivity

As in Section 5.3.1, we assume local smoothness of $F(y,\sigma)$ in both arguments y and σ so that $constant = F(VaR(\sigma);\sigma)$ locally. Then, differentiate to get

$$0 = \partial_\sigma\ F(y;\sigma)|_{y=VaR(\sigma)} + \partial_y\ F(y;\sigma)|_{y=VaR(\sigma)}\ \partial_\sigma VaR(\sigma).$$

For additional notation, define

$$\partial_y F(y;\sigma) = F_y(y;\sigma) \quad \text{and} \quad \partial_\sigma F(y;\sigma) = F_\sigma(y;\sigma).$$

Thus,

$$0 = F_\sigma[VaR(\sigma);\sigma] + F_y[VaR(\sigma);\sigma]\ \partial_\sigma VaR(\sigma),$$

which is sufficient for equation (12.3).

16.9.2 Section 12.3.2. Development of the RVaR Sensitivity

Using a change of variables ($z = F^{-1}(\gamma)$) and an integration by parts, we have

$$\begin{aligned}
\int_\alpha^{\alpha+\beta} F^{-1}(\gamma)\ d\gamma &= \int_{F^{-1}(\alpha)}^{F^{-1}(\alpha+\beta)} zf(z)dz \\
&= -z[1-F(z)]|_{F^{-1}(\alpha)}^{F^{-1}(\alpha+\beta)} + \int_{F^{-1}(\alpha)}^{F^{-1}(\alpha+\beta)}[1-F(z)]dz \\
&= F^{-1}(\alpha)(1-\alpha) - F^{-1}(\alpha+\beta)(1-\alpha-\beta) \\
&\quad +\{\mathrm{E}[g(\mathbf{X}) \wedge F^{-1}(\alpha+\beta)] - \mathrm{E}[g(\mathbf{X}) \wedge F^{-1}(\alpha)]\}.
\end{aligned}$$

Taking a partial derivative and using Leibniz's rule, we have

$$\begin{aligned}
\partial_\sigma\ \mathrm{E}[g(\mathbf{X}) \wedge F^{-1}(a)] &= \partial_\sigma \int_0^{F^{-1}(a)}[1-F(z)]dz \\
&= (1-a)\partial_\sigma F^{-1}(a) - \int_0^{F^{-1}(a)} F_\sigma(z)\ dz.
\end{aligned}$$

So, for $0 < \beta \le 1-\alpha$, we have

$$\begin{aligned}
\partial_\sigma \int_\alpha^{\alpha+\beta} F^{-1}(\gamma)\ d\gamma &= (1-\alpha)\partial_\sigma F^{-1}(\alpha) - (1-\alpha-\beta)\partial_\sigma F^{-1}(\alpha+\beta) \\
&\quad +\partial_\sigma\{\mathrm{E}[g(\mathbf{X}) \wedge F^{-1}(\alpha+\beta)] - \mathrm{E}[g(\mathbf{X}) \wedge F^{-1}(\alpha)]\} \\
&= (1-\alpha)\partial_\sigma F^{-1}(\alpha) - (1-\alpha-\beta)\partial_\sigma F^{-1}(\alpha+\beta) \\
&\quad +[1-(\alpha+\beta)]\partial_\sigma F^{-1}(\alpha+\beta) - \int_0^{F^{-1}(\alpha+\beta)} F_\sigma(z)dz \\
&\quad -(1-\alpha)\partial_\sigma F^{-1}(\alpha) + \int_0^{F^{-1}(\alpha)} F_\sigma(z)dz \\
&= -\int_{F^{-1}(\alpha)}^{F^{-1}(\alpha+\beta)} F_\sigma(z)dz,
\end{aligned}$$

as desired.

16.9.3 Example 12.5. Expected Shortfall for an Insurance Portfolio

With equation (12.4), we have

$$\begin{aligned}
\int_{F^{-1}(\alpha)}^{\infty} F_\sigma(t)dt &= \tfrac{1}{2}\int_{F^{-1}(\alpha)}^{\infty} \text{Cov}\left\{I[h(\mathbf{X}) \le t],\ W_\sigma(\mathbf{X})\right\} dt \\
&= \tfrac{-1}{2}\int_{F^{-1}(\alpha)}^{\infty} \text{Cov}\left\{I[h(\mathbf{X}) > t],\ W_\sigma(\mathbf{X})\right\} dt \\
&= \tfrac{-1}{2}\text{E}\left[\int_{F^{-1}(\alpha)}^{\infty} \left\{W_\sigma(\mathbf{X}) - \text{E}[W_\sigma(\mathbf{X})]\right\} I[h(\mathbf{X}) > t]\right] dt \\
&= \tfrac{-1}{2}\text{E}\left[\left\{W_\sigma(\mathbf{X}) - \text{E}[W_\sigma(\mathbf{X})]\right\} \left[h(\mathbf{X}) - F^{-1}(\alpha)\right]\ I[h(\mathbf{X}) > F^{-1}(\alpha)]\right] \\
&= \tfrac{-1}{2}\text{Cov}\left\{W_\sigma(\mathbf{X}), \left[h(\mathbf{X}) - F^{-1}(\alpha)\right]\ I[h(\mathbf{X}) > F^{-1}(\alpha)]\right\}.
\end{aligned}$$

This and equation (12.6) are sufficient for the result.

16.9.4 Section 12.4.3. Development of Pool Contagion Weights

Suppose there are $P = \sum_{i=1}^{N} p_i$ in risks in the pool and define the $P \times P$ patterned matrix

$$\begin{aligned}
\boldsymbol{\Sigma} = \text{Var}(\mathbf{y}) &= \text{Var}\begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_N \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{D}_1\mathbf{A}\mathbf{D}_1' + \boldsymbol{\Omega}_1 & \mathbf{D}_1\mathbf{A}\mathbf{D}_2' & \cdots & \mathbf{D}_1\mathbf{A}\mathbf{D}_N' \\ \mathbf{D}_2\mathbf{A}\mathbf{D}_1' & \mathbf{D}_2\mathbf{A}\mathbf{D}_2' + \boldsymbol{\Omega}_2 & \cdots & \mathbf{D}_2\mathbf{A}\mathbf{D}_N' \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{D}_N\mathbf{A}\mathbf{D}_1' & \mathbf{D}_N\mathbf{A}\mathbf{D}_2' & \cdots & \mathbf{D}_N\mathbf{A}\mathbf{D}_N' + \boldsymbol{\Omega}_N \end{pmatrix} \\
&= \mathbf{D}\mathbf{A}\mathbf{D}' + \boldsymbol{\Omega},
\end{aligned}$$

where $\mathbf{D} = (\mathbf{D}_1', \mathbf{D}_2', \ldots, \mathbf{D}_N')'$ has dimensions $P \times p$ and $\boldsymbol{\Omega} = blkdiag(\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2, \ldots, \boldsymbol{\Omega}_N)$.

Using a standard result on the inverse of partitioned matrices, we may write

$$\begin{aligned}
\boldsymbol{\Sigma}^{-1} &= \left[\mathbf{D}\mathbf{A}\mathbf{D}' + \boldsymbol{\Omega}\right]^{-1} \\
&= \boldsymbol{\Omega}^{-1} - \boldsymbol{\Omega}^{-1}\mathbf{D}\left[\mathbf{A}^{-1} + \mathbf{D}'\boldsymbol{\Omega}^{-1}\mathbf{D}\right]^{-1}\mathbf{D}\boldsymbol{\Omega}^{-1}.
\end{aligned}$$

In this, note that

$$\mathbf{D}'\boldsymbol{\Omega}^{-1}\mathbf{D} = \sum_{i=1}^{N} \mathbf{D}_i'\boldsymbol{\Omega}_i^{-1}\mathbf{D}_i$$

is a $p \times p$ matrix that is typically easy to invert. Let $\mathbf{ns}_\star(\mathbf{X})' = (\mathbf{ns}_\star(\mathbf{X})_1', \ldots, \mathbf{ns}_\star(\mathbf{X})_N')'$.

$$\begin{aligned}\mathbf{ns}_\star(\mathbf{X})'\boldsymbol{\Sigma}^{-1}\mathbf{D} &= \mathbf{ns}_\star(\mathbf{X})'\left\{\boldsymbol{\Omega}^{-1}\mathbf{D} - \boldsymbol{\Omega}^{-1}\mathbf{D}\left[\mathbf{A}^{-1} + \mathbf{D}'\boldsymbol{\Omega}^{-1}\mathbf{D}\right]^{-1}\mathbf{D}'\boldsymbol{\Omega}^{-1}\mathbf{D}\right\} \\ &= \left\{\mathbf{ns}_\star(\mathbf{X})'\boldsymbol{\Omega}^{-1}\mathbf{D}\right\}\left\{\mathbf{I} - \left[\mathbf{A}^{-1} + \mathbf{D}'\boldsymbol{\Omega}^{-1}\mathbf{D}\right]^{-1}\mathbf{D}'\boldsymbol{\Omega}^{-1}\mathbf{D}\right\} \\ &= \left\{\textstyle\sum_{i=1}^{N}\mathbf{ns}_\star(\mathbf{X})_i'\boldsymbol{\Omega}_i^{-1}\mathbf{D}_i\right\}\left\{\mathbf{I} - \left[\mathbf{I} + \mathbf{A}\mathbf{D}'\boldsymbol{\Omega}^{-1}\mathbf{D}\right]^{-1}\mathbf{A}\mathbf{D}'\boldsymbol{\Omega}^{-1}\mathbf{D}\right\}.\end{aligned}$$

Thus, with equation (12.2), we have the weights

$$\begin{aligned}W_\sigma(\mathbf{X}) &= \mathbf{ns}_\star(\mathbf{X})'\boldsymbol{\Sigma}^{-1}(\partial_a\boldsymbol{\Sigma})\boldsymbol{\Sigma}^{-1}\mathbf{ns}_\star(\mathbf{X}) \\ &= \left[\mathbf{ns}_\star(\mathbf{X})'\boldsymbol{\Sigma}^{-1}\mathbf{D}\right](\partial_a\mathbf{A})\left[\mathbf{D}'\boldsymbol{\Sigma}^{-1}\mathbf{ns}_\star(\mathbf{X})\right].\end{aligned}$$

17

Selected Exercise Solutions

17.1 Chapter One Exercise Solutions

17.1.1 Exercise 1.1. Trade-off between Price and Uncertainty, including the Effects of Dependence

```
#  Set-up for Exercise 1.1
risk1shape <- 2
risk1scale <- 5000
risk2shape <- 3
risk2scale <- 2000
```

```
# Exercise 1.1 Solution, Part a
set.seed(2017)
nsim      <- 100000
Q.def     <- 2.25*(ERisk1fun()+ERisk2fun())
uparam    <- seq(from = 0.0, to = q2(0.99), length.out = 11)
RTC11 <-  actuar::mpareto(order=1, shape = risk2shape, scale = risk2scale) -
          actuar::levpareto(limit = uparam,shape = risk2shape, scale = risk2scale)
u.compare <- 9
u.def     <- uparam[u.compare]
rhoparam  <- (-9:9)/10
rho.ind   <- 10; rho.lower <- 10-3; rho.upper <- 10+3
QuanS     <- matrix(0,length(uparam),length(rhoparam)) -> CTE
dfS       <- matrix(0,length(uparam),length(rhoparam))
# Generate Dist Function at Q.def and Quantiles
# for several dependence parameters
for (j in 1:length(rhoparam)) {
  FrankParam <- copula::iRho(copula::frankCopula(param=1, dim = 2), rho=rhoparam[j])
  nc <- copula::frankCopula(param = FrankParam, dim = 2)
  U  <- copula::rCopula(nsim, nc)
  X1 <- qgamma(U[,1],shape = risk1shape, scale = risk1scale)
  X2 <- actuar::qpareto(U[,2],shape = risk2shape, scale = risk2scale)
  #  Loop over several upper limits
  for (i in 1:length(uparam)) {
  u1 <- uparam[i]
  Sretained <- pmin(X1) + pmin(X2,u1)
  if ((j == rho.ind)  *(i==u.compare))    {Sret.ind   <- Sretained }
  if ((j == rho.lower)*(i==u.compare))    {Sret.lower <- Sretained }
  if ((j == rho.upper)*(i==u.compare))    {Sret.upper <- Sretained }
```

```
  df         <- ecdf(Sretained)
  dfS[i,j]   <- df(Q.def)
  QuanS[i,j] <- quantile(Sretained, probs = c(0.95))
  CTE[i,j]   <- sum(Sretained*(Sretained>= QuanS[i,j]))/
                    (1+sum(1*(Sretained>= QuanS[i,j])))
  }
}
Qaxis <- Q.def*seq(from = 0.8, to = 1.4, length.out = 11)
df <- ecdf(Sret.ind);   dfQ.ind   <- df(Qaxis)
df <- ecdf(Sret.lower); dfQ.lower <- df(Qaxis)
df <- ecdf(Sret.upper); dfQ.upper <- df(Qaxis)
Quanaxis    <- seq(0.92, 1, length.out = 11)
QuanS.ind   <- quantile(Sret.ind,   probs = Quanaxis)
QuanS.lower <- quantile(Sret.lower, probs = Quanaxis)
QuanS.upper <- quantile(Sret.upper, probs = Quanaxis)

save(Q.def, u.def, RTC11, dfS, dfQ.ind, dfQ.lower, dfQ.upper,
     QuanS, QuanS.ind,  QuanS.lower,  QuanS.upper,
#     Sret.ind, Sret.lower,  Sret.upper,
     rho.ind, rho.lower, rho.upper, rhoparam,
     CTE,
     file = "../ChapTablesData/Chap1/Exercise11.RData")
```

```
# Exercise 1.1 Solution,  Plot Figure 1.6
load(file = "ChapTablesData/Chap1/Exercise11.RData")

Q.def     <- 2.25*(ERisk1fun()+ERisk2fun())
Qaxis     <- Q.def*seq(from = 0.8, to = 1.4, length.out = 11)
uparam    <- seq(from = 0.0, to = q2(0.99), length.out = 11)
u.compare <- 9
u.def     <- uparam[u.compare]

par(mfrow = c(2,2))
plot(Qaxis, dfQ.ind, type = "l", ylab="Distribution Function",
     xlab = "Capital", main=paste("u=",round(u.def,digits=0)))
lines(Qaxis,dfQ.lower,  col=FigBlue, lty=2)
lines(Qaxis,dfQ.upper,  col=FigRed, lty=3)

plot(uparam,dfS[,rho.ind], type = "l", ylab=paste("Dist Fct at Q=", Q.def),
     xlab = "Limit Parameter u", ylim=c(.94,.96), xlim=c(0, 3000))
lines(uparam,dfS[,rho.lower],  col=FigBlue, lty=2)
lines(uparam,dfS[,rho.upper],  col=FigRed, lty=3)

plot(RTC11,QuanS.ind, type = "l", ylab="VaR",
     xlab = "Risk Transfer Cost", ylim = c(23600,25600))
lines(RTC11,QuanS.lower,  col=FigBlue, lty=2)
lines(RTC11,QuanS.upper,  col=FigRed, lty=3)

plot(RTC11,CTE[,rho.ind], type = "l", ylab="ES",
     xlab = "Risk Transfer Cost", ylim = c(29400,31800))
```

```
lines(RTC11,CTE[,rho.lower],  col=FigBlue, lty=2)
lines(RTC11,CTE[,rho.upper],  col=FigRed, lty=3)
```

- **b.** The upper left hand panel shows the distribution function $\Pr(S \leq Q)$ as a function of the capital amount Q. For this demonstration, we use $u = 5827$ to limit the retention for the second risk (this is above the 98th percentile). As anticipated, the distribution function increases as the capital Q increases. For each Q, the probability is smaller for positive dependence, the dotted red line that corresponds to a Spearman correlation of 0.3, and larger for negative dependence, the dashed blue line that corresponds to a Spearman correlation of -0.3.
- **c.** The upper right hand panel presents the distribution function, fixing the capital level at $Q = 24750$ and allowing the retention limit u to vary. Note that as the retention limit increases, the effects of dependence become more prominent.
- **d.** The lower left hand panel presents the quantile, also known as the value at risk (VaR), at the 95th confidence level, compared to the offload price, $\mathrm{E}[X_2 - X_2 \wedge u]$, over different choices of u.
  - When the limit is $u = 0$, the offload price is at its maximum, $E\ X_2 = 1000$.
  - As the limit increases to infinity, that is becomes unlimited, the offload price approaches zero.
  - This is the trade-off that the portfolio manager faces, a smaller price paid to offload the risk means a greater need for capital.
  - Note that the offload price does not change with the dependence in contrast to the capital requirement.
  - For each value of u/offload price, the VaR for positively associated risks is greater than the independence case and is lower for negatively associated risks. This is consistent with the information presented in the upper right hand panel and conforms to our intuition.
- **e.** The lower right hand panel presents the same type of information as in the lower left but for the ES, or expected shortfall, another widely used measure of capital adequacy. Qualitatively, it is consistent with the behavior of VaR.

17.1.2 Exercise 1.2. Solution

```
# Exercise 1.2 Solution, Plot Figure 1.7
load(file="ChapTablesData/Chap1/Exercise11.RData")
par(mfrow = c(1,2))
plot(rhoparam,dfS[6,], ylab=paste("Dist Fct at Q=",Q.def), xlim = c(-1,1) , cex=0.8,
     xlab="Spearman's Rho", type = "b", main=paste("u=", round(u.def,digits=0)))
plot(rhoparam,QuanS[6,], ylab="VaR", xlim = c(-1,1) , cex=0.8,
     xlab="Spearman's Rho", type = "b", main=paste("u=",round(u.def,digits=0)))
```

- **a.** The left hand panel presents the distribution function at Q= 24750 and u= 5827 but allowing the dependence, as quantified through Spearman's correlation, to vary. For a given retention level, the probability of retained risks being less than or equal to a given level of capital decreases as the level of association increases.
- **b.** The right hand panel presents the value at risk, 95th confidence level, at u= 5827 but allowing the dependence, as quantified through Spearman's correlation, to vary. The required capital increases as the level of association increases.

For both panels, the greater the dependence, the greater is the amount of capital required.

17.2 Chapter Two Exercise Solutions

17.2.1 Exercise 2.1. Comparing Parametric Value at Risk Measures

```
# Exercise 2.1 Illustrative Code
# Identify Pareto parameters by quantile matching
alpha1 = 0.2
alpha2 = 0.8
qPareto1 <- actuar::qpareto(p=alpha1, shape =0.999, scale = 2300)
qPareto2 <- actuar::qpareto(p=alpha2, shape =0.999, scale = 2300)

model <- function(MParms) {
  c(F1=qgamma(p=alpha1, shape=exp(MParms[1]),
              scale =exp(MParms[2])) -qPareto1,
    F2=qgamma(p=alpha2, shape=exp(MParms[1]),
              scale =exp(MParms[2])) -qPareto2)}

ss <- rootSolve::multiroot(f = model, start = c(log(0.5), log(20000)))
shape.gamma <- exp(ss$root[1])
scale.gamma <- exp(ss$root[2])
```

```
# Exercise 2.1 Figure 2.7
prob.seq   <- seq(0.50, 0.99, by=0.001)
VaR.Pareto <- qpareto(p=prob.seq, shape =0.999, scale = 2300)
VaR.gamma  <- qgamma(p=prob.seq, shape =shape.gamma, scale = scale.gamma)

par(mfrow=c(1, 2))
  plot(prob.seq, VaR.Pareto, xlab = 'Confidence Level',
     ylab = "VaR", type = 'l')
    lines(prob.seq,VaR.gamma, col = FigBlue, lty = 2)
  plot(prob.seq, VaR.Pareto, xlab = 'Confidence Level',
     log = "y", ylab = "VaR", type = 'l')
    lines(prob.seq,VaR.gamma, col = FigBlue, lty = 2)
```

17.2.2 Exercise 2.2. Comparing Parametric $RVaR$ Measures

```
# Exercise 2.2 Illustrative Code
# Identify Pareto parameters by quantile matching
alpha1 <- 0.2
alpha2 <- 0.8
qPareto1 <- actuar::qpareto(p=alpha1, shape =0.999, scale = 2300)
```

```
qPareto2 <- actuar::qpareto(p=alpha2, shape =0.999, scale = 2300)

model <- function(MParms) {
  c(F1=qgamma(p=alpha1, shape=exp(MParms[1]),
              scale =exp(MParms[2])) - qPareto1,
    F2=qgamma(p=alpha2, shape=exp(MParms[1]),
              scale =exp(MParms[2])) - qPareto2)}

ss <- rootSolve::multiroot(f = model, start = c(log(0.5), log(20000)) )
shape.gamma <- exp(ss$root[1])
scale.gamma <- exp(ss$root[2])
alpha.seq   <- seq(0.50, 0.80, by = 0.02)
beta.seq    <- seq(0.01, 0.19, by = 0.01)
RVaR.gamma  <- matrix(0, length(alpha.seq), length(beta.seq))
RVaR.Pareto <- matrix(0, length(alpha.seq), length(beta.seq))
qPareto.x   <- function(x){actuar::qpareto(p=x,
                                shape =0.999,       scale = 2300)}
qgamma.x    <- function(x){qgamma(p=x,
                                shape =shape.gamma, scale = scale.gamma)}

for (ia in (1:length(alpha.seq))) {
  for (ib in (1:length(beta.seq))){
    RVaR.gamma[ia,ib]= (1/beta.seq[ib])*
      integrate(qgamma.x,  lower = alpha.seq[ia],
                upper = alpha.seq[ia]+beta.seq[ib])$value
    RVaR.Pareto[ia,ib]= (1/beta.seq[ib])*
      integrate(qPareto.x, lower = alpha.seq[ia],
                upper = alpha.seq[ia]+beta.seq[ib])$value
  }}
```

```
# Exercise 2.2 Figure 2.8
par(mfrow=c(1, 2))
  persp(alpha.seq, beta.seq, RVaR.gamma, theta = 30, phi = 10, col = FigLBlue,
        xlab = expression(alpha), cex.axis = 0.4,
        ylab = expression(beta),
        zlab = expression(RVaR) , ticktype='detailed',nticks = 1,
        expand = 0.6, cex.lab=0.7)
  persp(alpha.seq, beta.seq, RVaR.Pareto,
        theta = 30, phi = 10, col = FigLGreen,
        xlab = expression(alpha), cex.axis = 0.4,
        ylab = expression(beta),
        zlab = "RVaR" , ticktype='detailed',nticks = 1,
        expand = 0.6, cex.lab = 0.7)
```

17.2.3 Exercise 2.3. Risk Measures for the Sum of Two Risks

```
# Exercise 2.3 Illustrative Code
# Part a
```

```
gamma.shape   <- 2
gamma.scale   <- 5000
Pareto.shape  <- 3
Pareto.scale  <- 2000
alpha         <- 0.95
SumDfConv <- function(y){
 integrandConv <- function(z){
    dgamma(z,shape = gamma.shape, scale = gamma.scale)*
    actuar::ppareto(y-z,shape = Pareto.shape, scale = Pareto.scale)
    }
 integrate(integrandConv, lower =0, upper = y)$value
 }
SumDfConv(18000)
# Part b
SumDfConv.a <- function(y){SumDfConv(y) - alpha}
(VaRSum.a   <- uniroot(SumDfConv.a, lower = 0, upper = 1e6)$root)
# Part c
SumSfConv   <- function(y){1-SumDfConv(y)}
SumSfConv   <- Vectorize(SumSfConv)
LimExpValue.a  <- integrate(SumSfConv, lower =0, upper = VaRSum.a)$value
(CTESum.a   <- VaRSum.a + (1/(1-alpha))*(10000 + 1000 - LimExpValue.a  ) )
```

17.2.4 Exercise 2.4. Risk Retention for the Sum of Two Risks

```
# Exercise 2.4 Illustrative Code
d <-100;c <-0.8;u<-30000
alpha <- 0.95
# Part a. Use equation (2.11)
( VaRg.95 <- c*pmax(0,pmin(VaRSum.a,u) - d) )
# Part b. Use equation (2.12)
LimExpValue.a  <- integrate(SumSfConv, lower =0, upper = VaRSum.a)$value
LimExpValue.d <- integrate(SumSfConv, lower =0, upper = d)$value
LimExpValue.u <- integrate(SumSfConv, lower =0, upper = u)$value
b.1 <- (c/(1-alpha))*(LimExpValue.u-LimExpValue.d)
b.2 <- (c/(1-alpha))*(LimExpValue.u-LimExpValue.a  +
                        (1-alpha)*(VaRSum.a-d))
b.3 <- c*(u-d)
( CTEg.a <- b.1*(VaRSum.a<d) + b.2*(VaRSum.a>d)*(VaRSum.a<u) +
            b.3*(VaRSum.a>u) )
# Part c. Use equation (2.13)
beta <- 0.04
SumDfConva.b<- function(y){SumDfConv(y) - alpha - beta}
VaRSum.a.b <- uniroot(SumDfConva.b, lower = 0, upper = 1e6)$root
LimExpValue.a.b <- integrate(SumSfConv, lower =0,
                             upper = VaRSum.a.b)$value
c.2 <- (c/beta)* ( (1-alpha-beta)*(d-VaRSum.a.b) +
                    (LimExpValue.a.b-LimExpValue.d) )
c.3 <- (c/beta)* ( (1-alpha-beta)*(d-u) +(LimExpValue.u-LimExpValue.d) )
c.4 <- (c/beta)* ( (1-alpha-beta)*(VaRSum.a-u) +
```

```
                    beta*(u-d) +(LimExpValue.u-LimExpValue.a) )
c.5 <- (c/beta)* ( (1-alpha-beta)*(VaRSum.a-VaRSum.a.b) +
                    beta*(VaRSum.a.b-d) +(LimExpValue.a.b-LimExpValue.a) )
c.6 <- c*(u-d)
#VaRSum.a;VaRSum.a.b;d;u
( RVaRg.a <-   c.2*(VaRSum.a<d)*(d<VaRSum.a.b)*(VaRSum.a.b<u) +
               c.3*(VaRSum.a<d)*(d<u)*(u<VaRSum.a.b) +
               c.4*(d<VaRSum.a)*(VaRSum.a<u)*(u<VaRSum.a.b) +
               c.5*(d<VaRSum.a)*(VaRSum.a.b<u) +
               c.6*(u<VaRSum.a)   )
# Part d.
CTESum.a.b <- VaRSum.a.b + (1/(1-alpha-beta))*
               (10000 + 1000 - LimExpValue.a.b )
( GlueVar  <- (VaRSum.a + CTESum.a + CTESum.a.b)/3     )
```

17.2.5 Exercise 2.5. Portfolio Management and Simulation

```
# Exercise 2.5 Set Up
# For the gamma distributions, use
alpha1 <- 2;      theta1 <- 100
alpha2 <- 2;      theta2 <- 200
# For the Pareto distributions, use
alpha3 <- 2;      theta3 <- 1000
alpha4 <- 3;      theta4 <- 2000
```

```
# Exercise 2.5 Illustrative Code
nSim <- 10000  #number of simulations
set.seed(2017) #set seed to reproduce work
X1 <- rgamma(nSim, alpha1, scale = theta1)
X2 <- rgamma(nSim, alpha2, scale = theta2)
# For the Pareto Distribution, use
X3 <- actuar::rpareto(nSim,scale=theta3,shape=alpha3)
X4 <- actuar::rpareto(nSim,scale=theta4,shape=alpha4)
c <- 0.8
d <- 100
u <- 5000
# Portfolio Risks
S          <- X1 + X2 + X3 + X4
g.S <- c*( pmin(S,u) - pmin(S,d) )
summary(g.S)
# Part a
plot(density(g.S), main="")
# Part b
mean(g.S)
sd(g.S)
# Part c
probSim <- c(0.80, 0.90, 0.95)
(VaRSim <- quantile(g.S, probs = probSim))
```

```
# Part d
CTESim <- 0*VaRSim
  for (i in 1:length(CTESim)) {
  excess <- pmax(g.S-VaRSim[i],0)
  CTESim[i] <- VaRSim[i] + mean(excess)/(1-probSim[i])
  }
CTESim
```

17.2.6 Exercise 2.6. Risk Retention Changes for the Sum of Two Risks

```
# Exercise 2.6 Illustrative Code
# First, run the code from Exercises 2.3 and 2.4.
# Part a - VaR
( VaRg.base <- c*pmax(0,pmin(VaRSum.a,u)     - d) )
( VaRg.d500 <- c*pmax(0,pmin(VaRSum.a,u)     - (d+500)) )
( VaRg.u500 <- c*pmax(0,pmin(VaRSum.a,u+500) - d) )
# Part a - ES
CTESumg.u.d <- function(d,u){
  LimExpValue.u <- integrate(SumSfConv, lower =0, upper = u)$value
  LimExpValue.d <- integrate(SumSfConv, lower =0, upper = d)$value
  b.1 <- (c/(1-alpha))*(LimExpValue.u-LimExpValue.d)
  b.2 <- (c/(1-alpha))*(LimExpValue.u-LimExpValue.a  +
                        (1-alpha)*(VaRSum.a-d) )
  b.3 <- c*(u-d)
  CTEg.u.d <- b.1*(VaRSum.a<d) + b.2*(VaRSum.a>d)*(VaRSum.a<u) +
              b.3*(VaRSum.a>u)
  return(CTEg.u.d)
}
CTESumg.u.d(d=100,    u=30000)
CTESumg.u.d(d=100+500,u=30000)
CTESumg.u.d(d=100,    u=30000+500)

# Part b
VaRg.d10  <- c*pmax(0,pmin(VaRSum.a,u) - (d+10))
VaRg.d50  <- c*pmax(0,pmin(VaRSum.a,u) - (d+50))
VaRg.d100 <- c*pmax(0,pmin(VaRSum.a,u) - (d+100))
ratiosVaR <- c((VaRg.d10 - VaRg.base)/10,
(VaRg.d50  - VaRg.base)/50,
(VaRg.d100 - VaRg.base)/100,
(VaRg.d500 - VaRg.base)/500)
ratiosVaR
ratiosCTE <- c(
(CTESumg.u.d(d=100+10,u=30000)  - CTESumg.u.d(d=100,u=30000))/10,
(CTESumg.u.d(d=100+50,u=30000)  - CTESumg.u.d(d=100,u=30000))/50,
(CTESumg.u.d(d=100+100,u=30000) - CTESumg.u.d(d=100,u=30000))/100,
(CTESumg.u.d(d=100+500,u=30000) - CTESumg.u.d(d=100,u=30000))/500)
ratiosCTE

# Part c
```

```
( diffVaR = -c*(VaRSum.a>d) )
( diffCTE = -(c/(1-alpha))*(1-SumDfConv(d))*(VaRSum.a<d)- c*(VaRSum.a>d)   )
```

17.3 Chapter Three Exercise Solutions

17.3.1 Exercise 3.1. Ridge and Lasso Regressions

a. With both problems, there are no equality constraints and $m_{iin} = 1$ inequality constraints. So equation (3.10) becomes $LA(\boldsymbol{\beta}, LMI) = f_0(\boldsymbol{\beta}) + LMI\ f_{in,1}(\boldsymbol{\beta})$, with the regression coefficients $\boldsymbol{\beta} = \boldsymbol{\theta}$ being the parameters of interest. The objective function is $f_0(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i'\boldsymbol{\beta})^2$. The constraints are $f_{in,1}(\boldsymbol{\beta}) = \sum_{j=1}^{p}|\beta_j|^s$. We can drop the "$c$" constants in the Lagrangian as they do not affect the determination of the optimal parameters.

b. Taking a derivative with respect to the jth regression coefficient of the Lagrangian yields

$$\begin{array}{rl} \partial_{\beta_j} LA(\boldsymbol{\beta}, LMI) & = \frac{-2}{n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i'\boldsymbol{\beta})\partial_{\beta_j}(\mathbf{x}_i'\boldsymbol{\beta}) + LMI\ \partial_{\beta_j}\sum_{j=1}^{p}\beta_j^2 \\ & = \frac{-2}{n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i'\boldsymbol{\beta})x_{ij} + 2LMI\ \beta_j \end{array}$$

Stacking this over $j = 1, \ldots, p$ yields

$$\begin{array}{rl} \partial_{\boldsymbol{\beta}} LA(\boldsymbol{\beta}, LMI) & = \frac{-2}{n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i'\boldsymbol{\beta})\mathbf{x}_i + 2LMI\ \boldsymbol{\beta} \\ & = \frac{-2}{n}\left(\mathbf{y}'\mathbf{X} - \boldsymbol{\beta}\mathbf{X}'\mathbf{X}\right) + 2LMI\ \boldsymbol{\beta} \end{array}$$

because

$$\sum_{i=1}^{n} y_i\mathbf{x}_i = (\mathbf{x}_1', \ldots, \mathbf{x}_n')\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \mathbf{X}'\mathbf{y}$$

and

$$\sum_{i=1}^{n}\mathbf{x}_i'\boldsymbol{\beta}\mathbf{x}_i = \left(\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i'\right)\boldsymbol{\beta} = \mathbf{X}'\mathbf{X}.$$

Setting this vector of partial derivatives to zero yields

$$\begin{array}{rl} \mathbf{0} & = \frac{-2}{n}\left(\mathbf{X}'\mathbf{y} - \hat{\boldsymbol{\beta}}^{ridge}\mathbf{X}'\mathbf{X} - n\ LMI\ \hat{\boldsymbol{\beta}}^{ridge}\right) \\ & = \frac{-2}{n}\left(\mathbf{X}'\mathbf{y} - \hat{\boldsymbol{\beta}}^{ridge}\left[\mathbf{X}'\mathbf{X} + n\ LMI\ \mathbf{I}\right]\right) \end{array} \quad \Longrightarrow\ \hat{\boldsymbol{\beta}}^{ridge} = \left[\mathbf{X}'\mathbf{X} + n\ LMI\ \mathbf{I}\right]^{-1}\mathbf{X}'\mathbf{y}.$$

c. When $LMI = 0$, the ridge regression estimator reduces to the ordinary least squares estimator, so $\hat{\boldsymbol{\beta}}^{ols} = \left[\mathbf{X}'\mathbf{X}\right]^{-1}\mathbf{X}'\mathbf{y}$. With the expression for the ridge regression estimator and $\mathbf{X}'\mathbf{X}\ \hat{\boldsymbol{\beta}}^{ols} = \mathbf{X}'\mathbf{y}$, we have

$$\begin{aligned}\hat{\boldsymbol{\beta}}^{ridge} &= [\mathbf{X}'\mathbf{X} + n\ LMI\ \mathbf{I}]^{-1}\mathbf{X}'\mathbf{y} \\ &= [\mathbf{X}'\mathbf{X} + n\ LMI\ \mathbf{I}]^{-1}\mathbf{X}'\mathbf{X}\ \hat{\boldsymbol{\beta}}^{ols}.\end{aligned}$$

Defining the weight matrix $\mathbf{W} = [\mathbf{X}'\mathbf{X} + n\ LMI\ \mathbf{I}]^{-1}\mathbf{X}'\mathbf{X}$ is sufficient for the proof.

$$\hat{\boldsymbol{\beta}}^{ridge} = \mathbf{W}\hat{\boldsymbol{\beta}}^{ols} + (\mathbf{I} - \mathbf{W})\mathbf{0},$$

d. The concentric ellipsoids in the upper right are contours of the goodness of fit; when travelling about the contours, the goodness of fit remains the same. One would like to get the contours as close as possible to the center, denoted by $\hat{\beta}$. However, the betas must fall within the constraint regions, the regions of permissible values. To do so, one expands the contours until it touches the constraint region.

In the left-hand panel of Figure 3.9, we see that it is possible to do so at the vertex of the diamond where the first beta is zero. In contrast, for the right-hand panel, one is close to but not exactly at β_1 equal to zero. This demonstrates graphically how the lasso may result in model simplification in contrast to the ridge.

17.3.2 Exercise 3.2. Upper Limit and $RVaR$

To see that the range value at risk $RVaR$ is also monotone increasing in u, from equation (2.10), we have the $VaR_\alpha(X \wedge u) = F_\alpha^{-1} \wedge u$ for the value at risk. Assume that $u_1 \leq u_2$. Then, from this equation (or the middle panel of Figure 2.5), you can check that $VaR_\alpha(X \wedge u_1) \leq VaR_\alpha(X \wedge u_2)$. Now, using the definition of the range value at risk in equation (2.10), if $\beta > 0$ then we have

$$\begin{aligned}RVaR_{(\alpha,\beta)}(X \wedge u_1) &= \tfrac{1}{\beta}\int_\alpha^{\alpha+\beta} VaR_a(X \wedge u_1)da \\ &\leq \tfrac{1}{\beta}\int_\alpha^{\alpha+\beta} VaR_a(X \wedge u_2)da \\ &= RVaR_{(\alpha,\beta)}(X \wedge u_2).\end{aligned}$$

This is sufficient to establish the required result, that the $RVaR_{(\alpha,\beta)}(X \wedge u)$ is monotone increasing in u.

17.3.3 Exercise 3.3. Alternative Training and Testing Periods for Portfolio of Insurance Stock Returns

```
# Exercise 3.3
load(file= "../ChapTablesData/Chap1/InsurSectorPrices2023.Rdata")
Return <- (prices/stats::lag(prices) - 1)[-1]
p.num  <- ncol(Return)  # number of stocks

#We now divide the data into a training set and test set:
# Training 2015-01-05 to 2021-05-31
# Testing 2021-06-01 to 2022-05-31
```

```
Return_trn <- Return[1:1665, ]
Return_tst <- Return[1666:1927, ]

T_trn <- nrow(Return_trn)
mu    <- colMeans(Return_trn, na.rm = TRUE)
Sigma <- cov(Return_trn, use = "pairwise.complete.obs")

portolioMinVar <- function(Sigma) {
  c.prop <- Variable(nrow(Sigma))
  prob   <- Problem(Minimize(quad_form(c.prop, Sigma)),
                 constraints = list(c.prop >= 0,
                                    sum(c.prop) == 1))
  result <- solve(prob)
  c.prop <- as.vector(result$getValue(c.prop))
  names(c.prop) <- colnames(Sigma)
  return(c.prop)
}
portolioMarkowitz <- function(mu, Sigma, lmd = 10) {
  c.prop <- Variable(nrow(Sigma))
  prob   <- Problem(Maximize(t(mu) %*% c.prop -
                               lmd*quad_form(c.prop, Sigma)),
                 constraints = list(c.prop >= 0,
                                    sum(c.prop) == 1))
  result <- solve(prob)
  c.prop <- as.vector(result$getValue(c.prop))
  names(c.prop) <- colnames(Sigma)
  return(c.prop)
}
c_Markowitz <- portolioMarkowitz(mu, Sigma)
c_MinVar    <- portolioMinVar(Sigma)
c_Equal     <- rep(1/8,8)
c_all       <- cbind("MinVar" = c_MinVar,
                "Markowitz"    = c_Markowitz,
                "Equal"        = c_Equal)

# compute returns of all portfolios
ret_all     <- xts(Return     %*% c_all, index(Return    ))
ret_all_trn <- xts(Return_trn %*% c_all, index(Return_trn))
ret_all_tst <- xts(Return_tst %*% c_all, index(Return_tst))

Lambda.vec <- c(seq(from=0, to = 3, length.out = 5),
                seq(from=3.5, to = 12, length.out = 5),
                seq(from=12.5, to = 100, length.out = 5))

Portmu     <- rep(0,length(Lambda.vec)) -> Portstd
PortPerf   <- matrix(0,2,length(Lambda.vec))
for (iVec in 1:length(Lambda.vec)) {
   MarkPort <-  portolioMarkowitz(mu, Sigma, lmd = Lambda.vec[iVec])
   ret_MarkPort <- xts(Return_trn %*% MarkPort, index(Return_trn))
   PortPerf[,iVec] <- t(table.AnnualizedReturns(ret_MarkPort)[1:2,1])
   }
```

```
temp1     <- t(table.AnnualizedReturns(ret_all_trn))
temp2     <- t(table.AnnualizedReturns(ret_all_tst))
PortSumm  <- cbind(as.matrix(temp1), as.matrix(temp2))
colnames(PortSumm) <- c("Train Ann Return", "Train Std Dev",
                        "Train Sharpe (Rf=0)",
                        "Test Ann Return", "Test Std Dev",
                        "Test Sharpe (Rf=0)")
save(PortSumm, file="../ChapTablesData/Chap3/Exercise33.Rdata")
```

```
# Exercise 3.3, Table 3.2
load(file="ChapTablesData/Chap3/Exercise33.Rdata")
colnames(PortSumm) <- c("Train Ann Return", "Train Std Dev",
                        "Train Sharpe (Rf=0)",
                        "Test Ann Return", "Test Std Dev",
                        "Test Sharpe (Rf=0)")

TableGen1(TableData=PortSumm,
 TextTitle='Portfolio Performance Measures for
            Alternative Training and Testing Periods',
         Align='cccccc', Digits=3, ColumnSpec=1:6,
         BorderRight= 1, ColWidth = ColWidth6)
```

Comparing training periods for the benchmark Table 3.1 and the new Table 3.2, we see little difference in annual return, and increase in the standard deviations that results in a decrease in the Sharpe ratios. For both tables, the Markowitz is the best portfolio based on the Sharpe ratio.

For the test periods, there is a marked decrease in the annualized returns when moving from the benchmark Table 3.1 to the new Table 3.2, with the standard deviations staying about the same, resulting in a marked decrease in the Sharpe ratios. In the new table, the Markowitz is a distant third with the minimum variance and equally weighted portfolios similar.

17.4 Chapter Four Exercise Solutions

17.4.1 Exercise 4.1. Portfolio Management

```
# Exercise 4.1
#In preparation, here is the code needed to set the parameters.
# For the gamma distributions, use
alpha1 <- 2;      theta1 <- 100
alpha2 <- 2;      theta2 <- 200
# For the Pareto distributions, use
alpha3 <- 2;      theta3 <- 1000
alpha4 <- 3;      theta4 <- 2000
```

```
# Limits
M1     <- 100
M2     <- 200

# Simulate the risks
nSim      <- 10000  #number of simulations
set.seed(2017) #set seed to reproduce work
X1        <- rgamma(nSim,alpha1,scale = theta1)
X2        <- rgamma(nSim,alpha2,scale = theta2)
# For the Pareto Distribution, use
library(actuar)
X3        <- actuar::rpareto(nSim,scale=theta3,shape=alpha3)
X4        <- actuar::rpareto(nSim,scale=theta4,shape=alpha4)
# Portfolio Risks
X         <- X1 + X2 + X3 + X4
Yretained <- pmin(X1,M1) + pmin(X2,M2)
YReinsurer<- X - Yretained

#a. Here is the code for the expected claim amounts.
ExpVec    <- t(as.matrix(c(mean(Yretained),mean(YReinsurer),mean(X))))
colnames(ExpVec) <- c("Retained", "Insurer","Total")

#b. Here is the code for the quantiles.
quantMat <- rbind(
                quantile(Yretained,  probs=c(0.80, 0.90, 0.95, 0.99)),
                quantile(YReinsurer, probs=c(0.80, 0.90, 0.95, 0.99)),
                quantile(X       ,   probs=c(0.80, 0.90, 0.95, 0.99)) )
rownames(quantMat) <- c("Retained", "Insurer","Total")

#c. Here is the code for the density plots of the retained, reinsurer,
#          and total portfolio risk.
par(mfrow=c(1,3))
plot(density(X), xlim=c(0,15000), main="Total Portfolio Risk", xlab="Loss")
plot(density(Yretained), xlim=c(0,500), main="Retained Portfolio Risk",
     xlab="Loss (Note the different horizontal scale)",
     ylab = "Density (Note different vertical scale)")
plot(density(YReinsurer), xlim=c(0,15000),
     main="Reinsurer Portfolio Risk", xlab="Loss")
```

17.4.2 Exercise 4.2. Sensitivity of Optimal Quota Share Proportions

Recall a basic relationship from vector calculus,

$$\frac{\partial}{\partial\sigma}\mathbf{\Sigma}^{-1} = \partial_\sigma\mathbf{\Sigma}^{-1} = -\mathbf{\Sigma}^{-1}\frac{\partial\mathbf{\Sigma}}{\partial\sigma}\mathbf{\Sigma}^{-1}.$$

As will be described in Section 9.4, both parameters $\mathbf{c} = \mathbf{c}(\mathbf{\Sigma})$ and $LME = LME(\mathbf{\Sigma})$ change when $\mathbf{\Sigma}$ changes. Specifically,

$$\begin{aligned} 0 &= \partial_\sigma RTC_{max} = -\left(\partial_\sigma LME\right) \tfrac{1}{2}\ \mathrm{E}(\mathbf{X})'\ \boldsymbol{\Sigma}^{-1}\ \mathrm{E}(\mathbf{X}) - \tfrac{LME}{2}\ \mathrm{E}(\mathbf{X})'\ \left(\partial_\sigma \boldsymbol{\Sigma}^{-1}\right)\ \mathrm{E}(\mathbf{X}) \\ &= -\left(\partial_\sigma LME\right) \tfrac{1}{2}\ \mathrm{E}(\mathbf{X})'\ \boldsymbol{\Sigma}^{-1}\ \mathrm{E}(\mathbf{X}) + \tfrac{LME}{2}\ \mathrm{E}(\mathbf{X})'\ \boldsymbol{\Sigma}^{-1} \left(\partial_\sigma \boldsymbol{\Sigma}\right) \boldsymbol{\Sigma}^{-1}\ \mathrm{E}(\mathbf{X}) \\ \Longrightarrow \quad \partial_\sigma LME &= LME\ \tfrac{\mathrm{E}(\mathbf{X})'\ \boldsymbol{\Sigma}^{-1}(\partial_\sigma \boldsymbol{\Sigma})\boldsymbol{\Sigma}^{-1}\ \mathrm{E}(\mathbf{X})}{\mathrm{E}(\mathbf{X})'\ \boldsymbol{\Sigma}^{-1}\ \mathrm{E}(\mathbf{X})} \end{aligned}$$

and

$$\begin{aligned} \partial_\sigma \mathbf{c} &= \left(\partial_\sigma LME\right) \tfrac{1}{2}\ \boldsymbol{\Sigma}^{-1}\ \mathrm{E}(\mathbf{X}) + \tfrac{LME}{2}\ \left(\partial_\sigma \boldsymbol{\Sigma}^{-1}\right)\ \mathrm{E}(\mathbf{X}) \\ &= \left(\partial_\sigma LME\right) \tfrac{1}{2}\ \boldsymbol{\Sigma}^{-1}\ \mathrm{E}(\mathbf{X}) - \tfrac{LME}{2}\ \boldsymbol{\Sigma}^{-1} \left(\partial_\sigma \boldsymbol{\Sigma}\right) \boldsymbol{\Sigma}^{-1}\ \mathrm{E}(\mathbf{X}). \end{aligned}$$

This is sufficient for equations (4.13) and (4.14).

17.4.3 Exercise 4.3. Markowitz Investment Optimal Allocations

Define the Lagrangian

$$\begin{aligned} LA &= \tfrac{1}{2}\mathrm{Var}[S(\mathbf{c})] + LME_1(Req_0 - \mathrm{E}[S(\mathbf{c})]) + LME_2(1 - \mathbf{c}'\mathbf{1}) \\ &= \tfrac{1}{2}\mathbf{c}'\boldsymbol{\Sigma}\mathbf{c} + LME_1[Req_0 - \mathbf{c}'\ \mathrm{E}(\mathbf{R})] + LME_2(1 - \mathbf{c}'\mathbf{1}). \end{aligned}$$

Taking partial derivatives, we have

$$\begin{aligned} \partial_{\mathbf{c}}\ LA &= \boldsymbol{\Sigma}\mathbf{c} - LME_1[\mathrm{E}(\mathbf{R})] - LME_2\mathbf{1} \\ \partial_{LME_1} LA &= Req_0 - \mathbf{c}'\ \mathrm{E}(\mathbf{R}) \\ \partial_{LME_2} LA &= 1 - \mathbf{c}'\mathbf{1}. \end{aligned}$$

Equating the first set of partial derivatives to zero, we can write the set of solutions as in equation (4.15).

Equating the last two set of partial derivatives to zero, we can insert this set of solutions to get

$$\begin{array}{llll} Req_0 & = \mathbf{c}'\ \mathrm{E}(\mathbf{R}) & = \left(LME_1\ \boldsymbol{\Sigma}^{-1}[\mathrm{E}(\mathbf{R})] + LME_2\ \boldsymbol{\Sigma}^{-1}\mathbf{1}\right)'\ \mathrm{E}(\mathbf{R}) & = A_1 LME_1 + A_2 LME_2 \\ 1 & = \mathbf{c}'\mathbf{1} & = \left(LME_1\ \boldsymbol{\Sigma}^{-1}[\mathrm{E}(\mathbf{R})] + LME_2\ \boldsymbol{\Sigma}^{-1}\mathbf{1}\right)'\mathbf{1} & = A_2 LME_1 + A_3 LME_2 \end{array}$$

Solving these two equations in terms of LME_1 and LME_2 yields expressions in equation (4.16).

17.4.4 Exercise 4.4. Risk Sharing Does not Always work with Heterogeneous Risks

Check: a. Using the independence of risks, sufficient for part (a) is

$$\begin{aligned} 1 &= \mathrm{Var}(X_2) \leq \mathrm{Var}(Y_2) = \tfrac{1}{2}\left(1 + 4\right) = 2.5 \\ &= \mathrm{Var}(Y_1) \leq \mathrm{Var}(X_1) = 4. \end{aligned}$$

b. We have

$$\begin{array}{ll} & \text{Var}(Y_j) = \left(\frac{1}{n}\sum_{j=1}^{n} X_j\right) = \frac{1}{n^2}\sum_{j=1}^{n}\sigma_j^2 \leq \sigma_2^2 \\ \iff & \sigma_1^2 + (n-1)\sigma_2^2 \leq n^2\sigma_2^2 \\ \iff & \sigma_1^2 \leq \left(n^2 - n + 1\right)\sigma_2^2, \end{array}$$

as desired.

c. For the first risk position to improve, we require

$$\begin{array}{ll} & \text{Var}(Y_1) = c^2\sigma_1^2 + (1-c)^2\sigma_2^2 \leq \sigma_1^2 \\ \iff & (1-c)^2\sigma_2^2 \leq [1-c^2]\sigma_1^2 \\ \iff & k_c = \frac{1-c}{1+c} = \frac{(1-c)^2}{1-c^2} \leq \frac{\sigma_1^2}{\sigma_2^2}. \end{array}$$

In the same way, for the second, we need

$$\begin{array}{ll} & \text{Var}(Y_2) = (1-c)^2\sigma_1^2 + c^2\sigma_2^2 \leq \sigma_2^2 \\ \iff & (1-c)^2\sigma_1^2 \leq [1-c^2]\sigma_2^2 \\ \iff & \frac{\sigma_1^2}{\sigma_2^2} \leq \frac{1+c}{1-c} = \frac{1}{k_c} \end{array}$$

So, we require $\sigma_2^2 k_c \leq \sigma_1^2 \leq \sigma_2^2/k_c$, as desired.

17.4.5 Exercise 4.5. Optimal Linear Exchange Based on a Clearing Exchange Condition

Check: For the minimization, we can use the method of Lagrange multipliers. The Lagrangian is

$$LA(\mathbf{C}) = trace(\mathbf{C\Sigma C}') + \boldsymbol{\lambda}_2'(\mathbf{C}'\mathbf{1} - \mathbf{1}).$$

Taking derivatives (here is a reminder), we get

$$\begin{array}{ll} \mathbf{0} = \frac{\partial\, LA(\mathbf{C})}{\partial\, \mathbf{C}} & = \frac{\partial}{\partial\, \mathbf{C}} trace(\mathbf{C\Sigma C}') + \frac{\partial}{\partial\, \mathbf{C}}\boldsymbol{\lambda}_2'\mathbf{C}'\mathbf{1} \\ & = 2\mathbf{C\Sigma} + \mathbf{1}\boldsymbol{\lambda}_2'. \end{array}$$

Pre-multiply this expression by $\mathbf{1}'$ to get

$$\begin{array}{ll} 0 & = 2\mathbf{1}'\mathbf{C\Sigma} + \mathbf{1}'\mathbf{1}\boldsymbol{\lambda}_2' \\ & = 2\mathbf{1}'\mathbf{\Sigma} + n\boldsymbol{\lambda}_2'. \end{array}$$

Solving for the Lagrange multiplier yields

$$\boldsymbol{\lambda}_2 = -\frac{2}{n}\mathbf{\Sigma 1}.$$

From this, the set of optimal allocation parameters is

$$\begin{aligned}\mathbf{C}_{opt} &= -\tfrac{1}{2}\mathbf{1}\boldsymbol{\lambda}_{2,opt}'\boldsymbol{\Sigma}^{-1} \\ &= -\tfrac{1}{2}\mathbf{1}(-\tfrac{2}{n}\boldsymbol{\Sigma}\mathbf{1})'\boldsymbol{\Sigma}^{-1} \\ &= \tfrac{1}{n}\mathbf{1}\mathbf{1}'.\end{aligned}$$

that verifies equation (4.17).

b. For the variance, we have $\text{Var}(\mathbf{Y}_{opt}) = \mathbf{C}_{opt}\Sigma\mathbf{C}_{opt}'$. With this, we have

$$\begin{aligned}\mathbf{C}_{opt}\boldsymbol{\Sigma}\mathbf{C}_{opt}' &= \tfrac{1}{n^2}\mathbf{1}\mathbf{1}'\boldsymbol{\Sigma}\mathbf{1}\mathbf{1}' \\ &= \tfrac{k_\Sigma}{n^2}\mathbf{1}\mathbf{1}',\end{aligned}$$

verifying the variance expression.

17.4.6 Exercise 4.6 Optimal Linear Exchange with Clearing Exchange and No Profits Conditions

Check: For the minimization, we again use the method of Lagrange multipliers. The Lagrangian is

$$LA(\mathbf{C}) = trace(\mathbf{C}\boldsymbol{\Sigma}\mathbf{C}') + \boldsymbol{\lambda}_1'(\mathbf{C}\boldsymbol{\mu} - \boldsymbol{\mu}) + \boldsymbol{\lambda}_2'(\mathbf{C}'\mathbf{1} - \mathbf{1}).$$

Taking derivatives, we get

$$\begin{aligned}\mathbf{0} = \tfrac{\partial\ LA(\mathbf{C})}{\partial\ \mathbf{C}} &= \tfrac{\partial}{\partial\ \mathbf{C}}trace(\mathbf{C}\boldsymbol{\Sigma}\mathbf{C}') + \tfrac{\partial}{\partial\ \mathbf{C}}\boldsymbol{\lambda}_1'\mathbf{C}\boldsymbol{\mu} + \tfrac{\partial}{\partial\ \mathbf{C}}\mathbf{1}'\mathbf{C}\boldsymbol{\lambda}_2 \\ &= 2\mathbf{C}\boldsymbol{\Sigma} + \boldsymbol{\lambda}_1\boldsymbol{\mu}' + \mathbf{1}\boldsymbol{\lambda}_2'.\end{aligned}$$

Thus,

$$\mathbf{C}_{opt} = -\tfrac{1}{2}\boldsymbol{\lambda}_{1,opt}\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1} - \tfrac{1}{2}\mathbf{1}\boldsymbol{\lambda}_{2,opt}'\boldsymbol{\Sigma}^{-1}$$

Now, pre-multiply the expression in equation (17.4.6) by $\mathbf{1}'$ to get

$$\begin{aligned}\mathbf{0} &= 2\mathbf{1}'\mathbf{C}\boldsymbol{\Sigma} + \mathbf{1}'\boldsymbol{\lambda}_1\boldsymbol{\mu}' + \mathbf{1}'\mathbf{1}\boldsymbol{\lambda}_2'. \\ &= 2\mathbf{1}'\boldsymbol{\Sigma} + \mathbf{1}'\boldsymbol{\lambda}_1\boldsymbol{\mu}' + n\boldsymbol{\lambda}_2'. \\ \rightarrow\ \boldsymbol{\lambda}_2 &= -\tfrac{1}{n}\left(\boldsymbol{\mu}\boldsymbol{\lambda}_1'\mathbf{1} + 2\boldsymbol{\Sigma}\mathbf{1}\right)\end{aligned}$$

Post-multiply the expression in equation (17.4.6) by $\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ to get

$$\begin{aligned}0 &= (2\mathbf{C}\boldsymbol{\Sigma} + \boldsymbol{\lambda}_1\boldsymbol{\mu}' + \mathbf{1}\boldsymbol{\lambda}_2')\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ &= 2\mathbf{C}\boldsymbol{\mu} + k_\mu\boldsymbol{\lambda}_1 - \tfrac{1}{n}\mathbf{1}(2\mathbf{1}'\boldsymbol{\Sigma} + \mathbf{1}'\boldsymbol{\lambda}_1\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ &= 2\boldsymbol{\mu} + k_\mu\boldsymbol{\lambda}_1 - \tfrac{1}{n}2\mathbf{1}\mathbf{1}'\boldsymbol{\mu} - \tfrac{1}{n}\mathbf{1}\mathbf{1}'\boldsymbol{\lambda}_1 k_\mu \\ &= 2(\boldsymbol{\mu} - \mathbf{1}\bar{\mu}) + k_\mu(\boldsymbol{\lambda}_1 - \mathbf{1}\tfrac{1}{n}\mathbf{1}'\boldsymbol{\lambda}_1) \\ \rightarrow\ \boldsymbol{\lambda}_1 &= -\tfrac{2}{k_\mu}\boldsymbol{\mu}.\end{aligned}$$

Inserting this into equation (17.4.6) yields

$$\begin{aligned}\boldsymbol{\lambda}_2 &= -\tfrac{1}{n}\left(\boldsymbol{\mu}\boldsymbol{\lambda}_1'\mathbf{1} + 2\boldsymbol{\Sigma}\mathbf{1}\right) \\ &= \tfrac{2}{nk_\mu}\left(\boldsymbol{\mu}\boldsymbol{\mu}'\mathbf{1} - k_\mu\boldsymbol{\Sigma}\mathbf{1}\right).\end{aligned}$$

Summarizing, we have

$$\begin{aligned}\mathbf{C}_{opt} &= -\tfrac{1}{2}\boldsymbol{\lambda}_{1,opt}\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1} - \tfrac{1}{2}\mathbf{1}\boldsymbol{\lambda}_{2,opt}'\boldsymbol{\Sigma}^{-1} \\ \boldsymbol{\lambda}_{1,opt} &= -\tfrac{2}{k_\mu}\boldsymbol{\mu} \\ \boldsymbol{\lambda}_{2,opt} &= \tfrac{2}{nk_\mu}\left(\boldsymbol{\mu}\boldsymbol{\mu}' - k_\mu\boldsymbol{\Sigma}\right)\mathbf{1}\end{aligned}$$

With this, we have

$$\begin{aligned}\mathbf{C}_{opt} &= -\tfrac{1}{2}\boldsymbol{\lambda}_{1,opt}\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1} - \tfrac{1}{2}\mathbf{1}\boldsymbol{\lambda}_{2,opt}'\boldsymbol{\Sigma}^{-1} \\ &= \tfrac{1}{k_\mu}\boldsymbol{\mu}\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1} - \tfrac{1}{nk_\mu}\mathbf{1}\mathbf{1}'\left[\boldsymbol{\mu}\boldsymbol{\mu}' - k_\mu\boldsymbol{\Sigma}\right]\boldsymbol{\Sigma}^{-1} \\ &= \tfrac{1}{k_\mu}\boldsymbol{\mu}\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1} - \tfrac{1}{k_\mu}\tfrac{\mathbf{1}'\boldsymbol{\mu}}{n}\mathbf{1}\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1} + \tfrac{1}{n}\mathbf{1}\mathbf{1}' \\ &= \tfrac{1}{k_\mu}(\boldsymbol{\mu} - \bar{\mu}\mathbf{1})\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1} + \tfrac{1}{n}\mathbf{1}\mathbf{1}'.\end{aligned}$$

With $\boldsymbol{\mu}^* = \boldsymbol{\mu} - \bar{\mu}\mathbf{1}$, this is sufficient for equation (4.18).

As a check, we can see that the solution satisfies the constraints, and so

$$\begin{aligned}\mathbf{C}_{opt}\boldsymbol{\mu} &= \tfrac{1}{k_\mu}(\boldsymbol{\mu} - \bar{\mu}\mathbf{1})\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \tfrac{1}{n}\mathbf{1}\mathbf{1}'\boldsymbol{\mu} \\ &= \tfrac{1}{k_\mu}(\boldsymbol{\mu} - \bar{\mu}\mathbf{1})k_\mu + \bar{\mu}\mathbf{1} = \boldsymbol{\mu}.\end{aligned}$$

For the other constraint, we have

$$\begin{aligned}\mathbf{1}'\mathbf{C}_{opt} &= \tfrac{1}{k_\mu}\mathbf{1}'(\boldsymbol{\mu} - \bar{\mu}\mathbf{1})\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1} + \tfrac{1}{n}\mathbf{1}'\mathbf{1}\mathbf{1}' \\ &= \mathbf{0} + \tfrac{1}{n}n\mathbf{1}' = \mathbf{1}'.\end{aligned}$$

---

b. Now, define $\text{Var}(\mathbf{Y}_{opt}) = \mathbf{C}_{opt}\Sigma\mathbf{C}_{opt}'$. With this, we have

$$\begin{aligned}\mathbf{C}_{opt}\boldsymbol{\Sigma}\mathbf{C}_{opt}' &= \left[\tfrac{1}{n}\mathbf{1}\mathbf{1}' + \tfrac{1}{k_\mu}\boldsymbol{\mu}^*\boldsymbol{\mu}'\ \boldsymbol{\Sigma}^{-1}\right]\boldsymbol{\Sigma}\mathbf{C}_{opt}' \\ &= \left[\tfrac{1}{n}\mathbf{1}\mathbf{1}'\boldsymbol{\Sigma} + \tfrac{1}{k_\mu}\boldsymbol{\mu}^*\boldsymbol{\mu}'\right]\left[\tfrac{1}{n}\mathbf{1}\mathbf{1}' + \tfrac{1}{k_\mu}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\boldsymbol{\mu}^{*\prime}\right] \\ &= \tfrac{1}{n^2}k_\Sigma\mathbf{1}\mathbf{1}' + \tfrac{1}{k_\mu}\bar{\mu}\left[\boldsymbol{\mu}^*\mathbf{1}' + \mathbf{1}\boldsymbol{\mu}^{*\prime}\right] + \tfrac{1}{k_\mu}\boldsymbol{\mu}^*\boldsymbol{\mu}^{*\prime}\end{aligned}$$

This is sufficient for the result.

17.4.7 Exercise 4.7 Optimal Linear Exchange - Special Case

```
# Exercise 4.7
library(CVXR)
num.row  <- 3
ones.vec <- as.vector(rep(1,num.row))
ones.mat <- matrix (rep(1,num.row^2),nrow = num.row)
X.sigma  <- matrix(c(10, -4 , -1 , -4 , 8 , 1,
                     -1 ,  1 , 1), nrow = num.row)
X.sigma.inv <- solve(X.sigma)
k.sigma     <- sum(X.sigma)
mu          <- as.vector(c( 20, 3, 10))
#mu = as.vector(c( 4, 6, 1))
Results.mat <- matrix(0,4,4)
Results.mat[1:3,1] <- diag(X.sigma)
Results.mat[4,1]   <- sum(Results.mat[1:3,1])

## Exchange Clearing
C.vec       <- Variable(3)
#objective  <- quad_form(C.vec, X.sigma)
objective   <- k.sigma*(C.vec[1]**2 +
                        C.vec[2]**2 +C.vec[3]**2 )
constraints <- list(sum(C.vec) == 1)
prob        <- Problem(Minimize(objective), constraints)
result      <- solve(prob)

C.opt.1     <- result$getValue(C.vec)
Summarize.Result <-function(Condnum,C.opt){
  Y.sigma <- (C.opt %*% ones.vec) %*%
              X.sigma %*% t(C.opt %*% ones.vec)
  Results.mat[1:3,Condnum+1] <- diag(Y.sigma)
  Results.mat[4,Condnum+1]   <- sum(Results.mat[1:3,Condnum+1])
  return(Results.mat)
  }
Results.mat <- Summarize.Result(Condnum=1,C.opt=C.opt.1)


## Add Limits
C.vec       <- Variable(3)
#objective  <- quad_form(C.vec, X.sigma)
objective   <- k.sigma*(C.vec[1]**2 +
                        C.vec[2]**2 +C.vec[3]**2 )
constraints <- list(sum(C.vec) == 1,  C.vec >= 0, C.vec <= 1)
prob        <- Problem(Minimize(objective), constraints)
result      <- solve(prob)
C.opt.3     <- result$getValue(C.vec)
Results.mat <- Summarize.Result(Condnum=2,C.opt=C.opt.3)

## Add  Risk Improvement

C.vec       <- Variable(3)
#objective  <- quad_form(C.vec, X.sigma)
objective   <- k.sigma*(C.vec[1]**2 + C.vec[2]**2 +C.vec[3]**2 )
```

```
constraints <- list(sum(C.vec) == 1,  C.vec >= 0, C.vec <= 1,
                    k.sigma*C.vec[1]**2 <= X.sigma[1,1],
                    k.sigma*C.vec[2]**2 <= X.sigma[2,2],
                    k.sigma*C.vec[3]**2 <= X.sigma[3,3])
prob        <- Problem(Minimize(objective), constraints)
result      <- solve(prob)
C.opt.4     <- result$getValue(C.vec)
Results.mat <- Summarize.Result(Condnum=3,C.opt=C.opt.4)

# Check short selling
C.opt.mat <- cbind(C.opt.1,C.opt.3,C.opt.4)

rownames(Results.mat) <- c("Agent 1", "Agent 2",
                           "Agent 3","Total")
colnames(Results.mat) <- c("Original",  "Clear Exchange",
                           "Short Sell", "Risk Improvement")

TableGen1(TableData=Results.mat,
         TextTitle='Ex 4.5 Risk Sharing  Variances',
         Align='rrrrr', Digits=4, ColumnSpec=1:4,
         BorderRight=1 )
```

```
# Exercise 4.7, Table 4.11
colnames(C.opt.mat) <- c("Clear Exchange", "Short Sell",
                         "Risk Improvement")
rownames(C.opt.mat) <- c("Agent 1", "Agent 2", "Agent 3")

TableGen1(TableData=C.opt.mat,
         TextTitle='Ex 4.5 Risk Sharing  Coefficients',
         Align='rrrrr', Digits=4, ColumnSpec=1:3,
         BorderRight=1 )
```

17.4.8 Exercise 4.8 Solution

```
# Exercise 4.8
library(CVXR)
timestamp <- Sys.time()
num.row   <- 20
set.seed(202122)
ones.vec  <- as.vector(rep(1,num.row))
ones.mat  <- matrix (rep(1,num.row^2),nrow = num.row)
X.Low     <- matrix(runif(num.row*num.row), nrow = num.row, ncol =num.row)
X.Low[upper.tri(X.Low, diag = FALSE)] <- 0
#round(X.Low,digits=3)
X.sigma   <- X.Low %*% t(X.Low)
#round(X.sigma,digits=3)

mu        <- as.vector(runif(num.row))
```

```
Results.mat              <- matrix(0,num.row+1,5)
Results.mat[1:num.row,1] <- diag(X.sigma)
Results.mat[num.row+1,1] <- sum(Results.mat[1:num.row,1])

## Exchange Clearing
C.mat     <- Variable(num.row,num.row)
objective <- quad_form(t(C.mat[1,]), X.sigma)
for (j in 2:num.row){
  objective <- objective + quad_form(t(C.mat[j,]), X.sigma)
  }
constraints <- list(t(C.mat) %*% ones.vec == ones.vec)
prob        <- Problem(Minimize(objective), constraints)
result      <- solve(prob)

C.opt.1     <- result$getValue(C.mat)
Summarize.Result <-function(Condnum,C.opt){
  Y.sigma <- C.opt %*% X.sigma %*% t(C.opt)
  Results.mat[1:num.row,Condnum+1] <- diag(Y.sigma)
  Results.mat[num.row+1,Condnum+1] <-
    sum(Results.mat[1:num.row,Condnum+1])
  return(Results.mat)
  }
Results.mat <- Summarize.Result(Condnum=1,C.opt=C.opt.1)

##  Exchange Clearing  and the No Profits Conditions
constraints <- list(t(C.mat) %*%
                      ones.vec == ones.vec, C.mat %*% mu == mu)
prob        <- Problem(Minimize(objective), constraints)
result      <- solve(prob)
C.opt.2     <- result$getValue(C.mat)
Results.mat <- Summarize.Result(Condnum=2,C.opt=C.opt.2)

## Add Limits
constraints <- list(t(C.mat) %*% ones.vec == ones.vec,
                    C.mat %*% mu == mu, C.mat >= 0, C.mat <= 1)
prob        <- Problem(Minimize(objective), constraints)
result      <- solve(prob)
C.opt.3     <- result$getValue(C.mat)
Results.mat <- Summarize.Result(Condnum=3,C.opt=C.opt.3)

## Add  Risk Improvement
cons        <-  X.sigma[1,1] - quad_form(t(C.mat[1,]), X.sigma)
  for (j in 2:num.row){
  cons <- min_elemwise(cons,X.sigma[j,j] -
                         quad_form(t(C.mat[j,]), X.sigma) )
  }
constraints <- list(t(C.mat) %*% ones.vec == ones.vec,
                    C.mat %*% mu == mu, C.mat >= 0, C.mat <= 1,
                    cons >= 0)
prob        <- Problem(Minimize(objective), constraints)
```

```
result      <- solve(prob)
C.opt.4     <- result$getValue(C.mat)
Results.mat <- Summarize.Result(Condnum=4,C.opt=C.opt.4)

# Check equality constraints
# t(C.opt.4) %*% ones.vec
# C.opt.4 %*% mu - mu
# Check short selling
#C.opt.3.mat <- cbind(C.opt.1[,3],C.opt.2[,3],
#                     C.opt.3[,3],C.opt.4[,3])

colnames(Results.mat) <- c("Original",  "Clear Exchange",
                           "No Profits","Short Sell",
                           "Risk Improvement") -> xlab1
#kableExtra::kable_styling(knitr::kable(Results.mat, digits=4,
#                   caption="Risk Sharing Model Variances", align = "rrrrr"))

plot(1:5, Results.mat[1,], type = "b", ylim = c(0,8),
               xlab="Type of Model", ylab=
              "Model Variances", xaxt = "n", yaxt = "n")
axis(side=1, at=1:5, labels=xlab1)
axis(side=2, las=2)
if (HtmlEval) {
for (j in 2:num.row){lines(1:5, Results.mat[j,], type = "b", col=j) }
  } else {
for (j in 2:num.row){lines(1:5, Results.mat[j,], type = "b") }
   }

#Sys.time() - timestamp # n=10 Time difference of 13.05655 secs
# n=20 Time difference of 22.74492 secs (ylim = c(0,8))
# n= 50 Time difference of 3.576095 mins (ylim = c(0,20))
```

17.4.9 Exercise 4.9. Conditional Mean Risk-Sharing Rules

Because $g_{i,n}(s)$ is non-decreasing in s, we have

$$\begin{aligned} B_{i,n} &= (w_{i,n} - g_{i,n}(S_n))_+ = (g_{i,n}(w_n)) - g_{i,n}(S_n))_+ \\ &= \begin{cases} 0 & w_n \leq S_n \\ g_{i,n}(w_n)) - g_{i,n}(S_n) & w_n > S_n \end{cases} . \end{aligned}$$

Note that the event $\{w_n > S_n\}$ does not depend on i. Thus, we can sum both sides over $i = 1, \ldots, n$ to get the result.

17.5 Chapter Five Exercise Solutions

17.5.1 Exercise 5.1. Lack of Convexity for Value at Risk

```
#  Exercise 5.1 Illustrative Code
risk1shape <- 2;           risk1scale <- 5000
risk2shape <- 5         ;  risk2scale <- 25000
rho = -0.3
alpha = 0.9
# Set up Two Risks, gamma and a Pareto

f1 <- function(x){dgamma(x=x,shape = risk1shape, scale = risk1scale)}
F1 <- function(x){pgamma(q=x,shape = risk1shape, scale = risk1scale)}
q1 <- function(x){qgamma(p=x,shape = risk1shape, scale = risk1scale)}
f2 <- function(x){actuar::dpareto(x=x,shape = risk2shape, scale = risk2scale)}
F2 <- function(x){actuar::ppareto(q=x,shape = risk2shape, scale = risk2scale)}
q2 <- function(x){actuar::qpareto(p=x,shape = risk2shape, scale = risk2scale)}
Prob.fct <- function(u1,u2,rho,y){
  u1       <- u1*(u1>0)
  u2       <- u2*(u2>0)
  norm.cop <- copula::normalCopula(param=rho, dim = 2,'dispstr' ="un")
  ProbA <- 0 -> ProbB -> ProbC -> ProbD1 -> ProbD2
  if (y>u1) { ProbB <- F2(y-u1) - copula::pCopula(c(F1(u1),F2(y-u1)), norm.cop) }
  if (y>u2) { ProbC <- F1(y-u2) - copula::pCopula(c(F1(y-u2),F2(u2)), norm.cop) }
  ProbD1.x <- function(x){VineCopula::BiCopHfunc1(F1(x),F2(y-x), family=1, par=rho)*f1(x)}
  ProbD1.x.vec <- Vectorize(ProbD1.x)
  ProbD1   <-  cubature::cubintegrate(ProbD1.x.vec, lower = max(0,y-u2), upper = min(y,u1),
                                relTol =1e-4, absTol =1e-8, method = "hcubature")$integral
  if (y>u2) { ProbD2 <-  copula::pCopula(c(F1(y-u2),F2(u2)), norm.cop) }
  ProbTot <- ProbA + ProbB + ProbC + ProbD1 + ProbD2
  ProbTot <- ProbTot + (1-ProbTot)*(u1+u2<y)
  ProbTot <- ProbTot*(y>0)
  return(ProbTot)
}

cVec <- seq(0, 1, length.out=5)
u1.L <- qgamma(p=0.70,shape = risk1shape, scale = risk1scale)
u1.R <- qgamma(p=0.90,shape = risk1shape, scale = risk1scale)

u2.L <- actuar::qpareto(p=0.70,shape = risk2shape, scale = risk2scale)
u2.R <- actuar::qpareto(p=0.90,shape = risk2shape, scale = risk2scale)

u1Vec.a <- u1.L*(1-cVec) + u1.R*cVec
u1Vec.b <- u1.R*(1-cVec) + u1.L*cVec
u2Vec   <- u2.L*(1-cVec) + u2.R*cVec

y.Vec <- function(par){par[1]}

Var.fct <- function(u1,u2){
```

```
Prob.y.Vec <- function(par){
  y=par[1]
  return(1e6*(Prob.fct(u1,u2,rho,y) - alpha))
}
return(uniroot(Prob.y.Vec, lower = 1, upper = 2e5)$root)
}

VarVec.a <- 0 * cVec -> VarVec.b
for (kindex in 1:length(cVec)) {
  VarVec.a[kindex] <- Var.fct(u1Vec.a[kindex],u2Vec[kindex])
  VarVec.b[kindex] <- Var.fct(u1Vec.b[kindex],u2Vec[kindex])
}

par(mfrow=c(1,2))
  plot(cVec,VarVec.a, type = "l", ylab = "Value at Risk",
       xlab = "c", ylim = c(17000,23000))
    abline(a=VarVec.a[1],b=VarVec.a[length(cVec)]-VarVec.a[1],
       lty = "dashed", col =FigRed)
  plot(cVec,VarVec.b, type = "l", ylab = "Value at Risk",
       xlab = "c", ylim = c(17000,23000))
    abline(a=VarVec.b[1],b=VarVec.b[length(cVec)]-VarVec.b[1],
       lty = "dashed", col =FigRed)
```

17.5.2 Exercise 5.2. Lack of Convexity for Expected Shortfall

```
#  Exercise 5.2 Illustrative Code
nsim <- 200000
set.seed(202020)
risk1shape <- 2;           risk1scale <- 5000
risk2shape <- 5         ;  risk2scale <- 25000

rho   <- -0.3
alpha <- 0.9

#  Normal, or Gaussian, Copula
norm.cop <- copula::normalCopula(param=rho, dim = 2,'dispstr' ="un")
UCop     <- copula::rCopula(nsim, norm.cop)

#  Marginal Distributions
X1   <- qgamma(p=UCop[,1],shape = risk1shape, scale = risk1scale)
u1.L <- qgamma(p=0.70,shape = risk1shape, scale = risk1scale)
u1.R <- qgamma(p=0.90,shape = risk1shape, scale = risk1scale)

X2   <- actuar::qpareto(p=(UCop[,2]),shape = risk2shape, scale = risk2scale)
u2.L <- actuar::qpareto(p=0.70,shape = risk2shape, scale = risk2scale)
u2.R <- actuar::qpareto(p=0.90,shape = risk2shape, scale = risk2scale)

cVec    <- seq(0, 1, length.out=200)
u1Vec.a <- u1.L*(1-cVec) + u1.R*cVec
```

```
u1Vec.b <- u1.R*(1-cVec) + u1.L*cVec
u2Vec   <- u2.L*(1-cVec) + u2.R*cVec

#  Objective Function
f0 <- function(u1,u2){
    S <- pmin(X1,u1) +  pmin(X2,u2)
    quant <- quantile(S, probs=alpha)
    excess      <- pmax(S - quant,0)
    CTE         <- quant + mean(excess)/(1-alpha)
    return(CTE)
    }
CTEVec.a <- 0 * cVec -> CTEVec.b
for (kindex in 1:length(cVec)) {
  CTEVec.a[kindex] <- f0(u1Vec.a[kindex],u2Vec[kindex])
  CTEVec.b[kindex] <- f0(u1Vec.b[kindex],u2Vec[kindex])
}

save(cVec,CTEVec.a, CTEVec.b,
    file = "../ChapTablesData/Chap5/Exer512.Rdata")
```

17.5.3 Exercise 5.3. Special Case. Excess of Loss Distribution Function for Independent Uniformly Distributed Losses

Part (a).

Case I. Take $y < u_2$. Then

$$\Pr(X_1 \wedge u_1 + \quad X_2 \wedge u_2 \le y) = \Pr(X_1 + X_2 \le y) = \begin{cases} y^2/2 & \text{if } y < 1 \\ 1 - \frac{1}{2}(1-y)^2/2 & \text{if } y \ge 1. \end{cases}$$

Case II. Take $u_2 < y < u_1$. Then

$$\begin{aligned} &\Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \le y) \\ &= \Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \le y, X_1 \le u_1) + \Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \le y, X_1 \le u_1) \\ &= \Pr(X_1 + X_2 \wedge u_2 \le y, X_1 \le u_1) + \Pr(u_1 + X_2 \wedge u_2 \le y, X_1 \le u_1) \\ &= \Pr(X_1 + X_2 \wedge u_2 \le y, X_1 \le u_1), \end{aligned}$$

because $y < u_1$.

$$\begin{aligned} &\Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \le y) \\ &= \Pr(X_1 + X_2 \wedge u_2 \le y, X_1 \le u_1, X_2 \le u_2) + \Pr(X_1 + X_2 \wedge u_2 \le y, X_1 \le u_1, X_2 > u_2) \\ &= \Pr(X_1 + X_2 \le y, X_1 \le u_1, X_2 \le u_2) + \Pr(X_1 + u_2 \le y, X_1 \le u_1, X_2 > u_2) \end{aligned}$$

Because $u_1 + u_2 < y$, we have $\min(y - u_2, u_1) - y - u_2$. Thus,

$$\begin{aligned} &\Pr(X_1 + u_2 \le y, X_1 \le u_1, X_2 > u_2) = \Pr(X_1 \le \min(y - u_2, u_1), X_2 > u_2) \\ &= \Pr(X_1 \le y - u_2, X_2 > u_2) = (y - u_2)(1 - u_2). \end{aligned}$$

$$
\begin{aligned}
&= \Pr(X_1 + X_2 \le y, X_1 \le u_1, X_2 \le u_2) \\
&= \textstyle\int_0^{u_2} \Pr(X_1 + z \le y, X_1 \le u_1 | X_2 = z) f_{X_2}(z) dz \\
&= \textstyle\int_0^{u_2} \Pr[X_1 \le \min(y - z, u_1)] dz = \int_0^{u_2} \min(y - z, u_1) dz \\
&= \textstyle\int_0^{u_2} (y - z) dz = u_2(y - u_2/2).
\end{aligned}
$$

Thus, for Case II, we have

$$
\begin{aligned}
&\Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \le y) \\
&= u_2(y - u_2/2) + (y - u_2)(1 - u_2) \\
&= u_2^2/2 + y - u_2.
\end{aligned}
$$

Case III. Take $u_2 < y$. Now, split the domain into four components, $A_1 = \{X_1 > u_1, X_2 > u_2\}$,$A_2 = \{X_1 > u_1, X_2 \le u_2\}$,$A_3 = \{X_1 \le u_1, X_2 > u_2\}$,and $A_4 = \{X_1 \le u_1, X_2 \le u_2\}$. On A_1, we have

$$
\begin{aligned}
&\Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \le y, A_1) \\
&= \Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \le y, X_1 > u_1, X_2 > u_2) \\
&= \Pr(u_1 + u_2 \le y, X_1 > u_1, X_2 > u_2) \\
&= I(u_1 + u_2 \le y)(1 - u_1)(1 - u_2).
\end{aligned}
$$

On A_2, we have

$$
\begin{aligned}
&\Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \le y, A_2) \\
&= \Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \le y, X_1 > u_1, X_2 \le u_2) \\
&= \Pr(X_2 \le y - u_1, X_1 > u_1, X_2 \le u_2) \\
&= \min(u_2, y - u_1)(1 - u_1) = (y - u_1)(1 - u_1).
\end{aligned}
$$

In the same way, on A_3,

$$
\begin{aligned}
&\Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \le y, A_3) \\
&= (y - u_2)(1 - u_2).
\end{aligned}
$$

For A_4, we have

$$
\begin{aligned}
&\Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \le y, A_4) \\
&= \Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \le y, X_1 \le u_1, X_2 \le u_2) \\
&= \Pr(X_1 + X_2 \le y, X_1 \le u_1, X_2 \le u_2) \\
&= u_1 u_2 - \tfrac{1}{2}(u_1 + u_2 - y)^2
\end{aligned}
$$

A graph helps one see the last equality.

Altogether this yields for Case III,

$$
\begin{aligned}
&\Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \le y) \\
&= u_1 u_2 - \tfrac{1}{2}(u_1 + u_2 - y)^2 + (y - u_1)(1 - u_1) + (y - u_2)(1 - u_2)
\end{aligned}
$$

This is sufficient for part (a).

For part (b), we have the following code.

```
# Exercise 5.3 Illustrative Code, Fig 5.17
dfFunc = function(u1,u2,y) {
  PartA <- y^2/2*(y<1) + (1-(2-y)^2/2)*(y>=1)*(y<=2)
  PartB <- u2^2/2 + y - u2
  PartC <- u1*u2 - (0.5)*(u1 + u2 -y)^2 +
          (y-u1)*(1-u1) + (y-u2)*(1-u2)
  df <- PartA * (y < u2) + PartB * (u2 < y)*(y < u1) +
      PartC * (u1 < y)
  df <- df + (1-df)*(u1+u2 <= y)
  return(df)
}

u1 <- 0.80
u2 <- 0.40
y.arg  <- seq(from=0, to = 2, length.out=200)
dfLine <- dfFunc(u1,u2, y = y.arg)
plot(y.arg,dfLine, ylim = c(0,1), type = "l",
     ylab = "Dist Fct", xlab = "y")
```

```
# Exercise 5.3 Illustrative Code, Fig 5.17
dfFunc = function(u1,u2,y) {
  PartA <- y^2/2*(y<1) + (1-(2-y)^2/2)*(y>=1)*(y<=2)
  PartB <- u2^2/2 + y - u2
  PartC <- u1*u2 - (0.5)*(u1 + u2 -y)^2 +
          (y-u1)*(1-u1) + (y-u2)*(1-u2)
  df <- PartA * (y < u2) + PartB * (u2 < y)*(y < u1) +
      PartC * (u1 < y)
  df <- df + (1-df)*(u1+u2 <= y)
  return(df)
  }

u1 <- 0.80
u2 <- 0.40
y.arg  <- seq(from=0, to = 2, length.out=200)
dfLine <- dfFunc(u1,u2, y = y.arg)
plot(y.arg,dfLine, ylim = c(0,1), type = "l",
     ylab = "Dist Fct", xlab = "y")
```

17.5.4 Exercise 5.4. Special Case. Excess of Loss Value at Risk for Independent Uniformly Distributed Losses

```
# Exercise 5.4 Illustrative Code
dfFunc = function(u1,u2,y) {
  PartA <- y^2/2*(y<1) + (1-(2-y)^2/2)*(y>=1)*(y<=2)
  PartB <- u2^2/2 + y - u2
  PartC <- u1*u2 - (0.5)*(u1 + u2 -y)^2 +
```

```
          (y-u1)*(1-u1) + (y-u2)*(1-u2)
  df <- PartA * (y < u2) + PartB * (u2 < y)*(y < u1) +
      PartC * (u1 < y)
  df <- df + (1-df)*(u1+u2 <= y)
  return(df)
}

u1 <- 0.80
u2 <- 0.40
alpha.seq <- seq(from=0.00001, to = 0.7, length.out=200)
VaR.seq <- alpha.seq*0

for (ialpha in 1:length(alpha.seq)) {
  dfFuncu1u2 <- function(y){dfFunc(u1,u2,y) - alpha.seq[ialpha]}
  VaR.seq[ialpha] <- uniroot(dfFuncu1u2, lower = 0, upper = 2)$root
}
```

17.5.5 Exercise 5.5. Risk Sensitivity the Sum of Two Risks

```
# Exercise 5.5 Illustrative Code
# First, run the code from Exercises 2.3, 2.4, and 2.6.

gamma.shape   <- 2
gamma.scale   <- 5000
Pareto.shape  <- 3
Pareto.scale  <- 2000
alpha         <- 0.95
SumDfConv <- function(y){
 integrandConv <- function(z){
    dgamma(z,shape = gamma.shape, scale = gamma.scale)*
    actuar::ppareto(y-z,shape = Pareto.shape, scale = Pareto.scale)
    }
 integrate(integrandConv, lower =0, upper = y)$value
 }
SumDfConv.a <- function(y){SumDfConv(y) - alpha}
VaRSum.a   <- uniroot(SumDfConv.a, lower = 0, upper = 1e6)$root


# Part a - Recall that the quantile of the retained risk is:
d=100;c=0.8;u=30000
( VaRg.base <- c*pmax(0,pmin(VaRSum.a,u) - d) )

# From equation(2.9)
SumDfRet <- function(par){
  z=par[1];d=par[2];c=par[3];u=par[4]
  SumDfConv(d)*(z==0) +
    SumDfConv(z/c+d)*(z>0)*(z<c*(u-d)) +
    1*(z>=c*(u-d))
}
```

```
#SumDfRet(c(VaRg.base,d,c,u))
# SumDfConv(VaRg.base)
# SumDfRet(c(VaRg.base,d=100,c=0.8,u=30000))
Derivates <- numDeriv::grad(f=SumDfRet,
                            x = c(VaRg.base,d,c,u), method = "simple")
( QuanSen <- -Derivates[2:4] / Derivates[1] )

# Part b

LimExpValueg.fct <- function(par){
  z=par[1];d=par[2];c=par[3];u=par[4]
  SumSfRet <- function(y){1-SumDfRet(c(y,d,c,u))}
  SumSfRet.vec <- Vectorize(SumSfRet)
  LimExp <- integrate(SumSfRet.vec, lower =0, upper = z)$value
  return(LimExp)
  }

partb.1 <- (1/(1-alpha))*(
           numDeriv::grad(f=LimExpValueg.fct,
                 x = c(z=1e7,d=100,c=0.8,u=30000), method = "simple") -
           numDeriv::grad(f=LimExpValueg.fct,
                 x = c(z=VaRg.base,d=100,c=0.8,u=30000), method = "simple")
         )
partb.1 <- partb.1[2:4]
CTE1Prime <- 1 - (1/(1-alpha))*(1-SumDfRet(c(VaRg.base,d,c,u)))
( Partb <- partb.1 + CTE1Prime* QuanSen )

save(VaRg.base,QuanSen, partb.1,
    file = "../ChapTablesData/Chap5/Exer531.Rdata")
```

17.5.6 Exercise 5.6. Single Variable Upper Limit - VaR

For this exercise, the parameter of interest is the upper limit $\theta = u$ and the distribution function is $F(\theta; z) = F_{X \wedge u}(u; z)$. Recall the distribution function,

$$F_{X \wedge u}(z) = \begin{cases} 1 & \text{if } u \leq z \\ F(z) & \text{if } u > z \end{cases} .$$

From equation (2.10), the quantile is

$$VaR_\alpha[X \wedge u] = VaR_\alpha(u) = \begin{cases} u & \text{if } u \leq F_\alpha^{-1} \\ F_\alpha^{-1} & \text{if } u > F_\alpha^{-1} \end{cases} .$$

Thus, $\partial_u VaR_\alpha(u) = 1$ if $u \leq F_\alpha^{-1}$ and 0 otherwise. The discreteness induced by the upper limit parameter had been suggested by Figure 2.3.

For equation (5.5)

$$\partial_{\boldsymbol{\theta}} VaR(\boldsymbol{\theta}) \quad = \frac{-F_{\boldsymbol{\theta}}[\boldsymbol{\theta}, y]}{F_y[\boldsymbol{\theta}, y]} \Big|_{y = VaR(\boldsymbol{\theta})} .$$

$$F_y[\boldsymbol{\theta}, y] = \partial_y F_{X \wedge u}(y) = f(u) I(y \geq u)$$

and

$$F_{\boldsymbol{\theta}}[\boldsymbol{\theta}, y] = \partial_u F_{X \wedge u}(y) = 0.$$

This ratio is clearly not equal to the value from Table 2.2.

17.5.7 Exercise 5.7. Single Variable Upper Limit - ES

For this exercise, the parameter of interest is the upper limit $\theta = u$ and the distribution function is $F_g(\theta; z) = F_{X \wedge u}(u; z)$. Recall the distribution function,

$$F_{X \wedge u}(z) = \begin{cases} 1 & \text{if } u \leq z \\ F(z) & \text{if } u > z \end{cases}.$$

From equation (2.10), the quantile is

$$VaR_\alpha[X \wedge u] = VaR_\alpha(u) = \begin{cases} u & \text{if } u \leq F_\alpha^{-1} \\ F_\alpha^{-1} & \text{if } u > F_\alpha^{-1} \end{cases}.$$

Thus, $\partial_u VaR_\alpha(u) = 1$ if $u \leq F_\alpha^{-1}$ and 0 otherwise. From equation (5.7), we have

$$\begin{aligned} \partial_u ES_\alpha[X \wedge u] &= \tfrac{1}{1-\alpha} \left\{ \partial_u \mathrm{E}[X \wedge u] - \partial_u \mathrm{E}[X \wedge u \wedge z]|_{z=VaR_\alpha(u)} \right\} \\ &\quad + \{1 - \tfrac{1}{1-\alpha}(1 - F_{X \wedge u}[u; VaR_\alpha(u)])\} \times \partial_u VaR_\alpha(u). \end{aligned}$$

As we have seen before, $\mathrm{E}(X \wedge u) = \int^u [1 - F(z)]dz$. Thus, $\partial_u \mathrm{E}(X \wedge u) = 1 - F(u)$ and

$$\partial_u \mathrm{E}(X \wedge u \wedge z) = \begin{cases} \partial_u \mathrm{E}(X \wedge u) = 1 - F(u) & \text{if } u \leq z \\ \partial_u \mathrm{E}(X \wedge z) = 0 & \text{if } u > z \end{cases}.$$

Thus,

$$\partial_u \mathrm{E}(X \wedge u \wedge z)|_{z=F^{-1}_{X \wedge u}(\alpha)} = \begin{cases} 1 - F(u) & \text{if } u \leq F_\alpha^{-1} \\ 0 & \text{if } u > F_\alpha^{-1} \end{cases}.$$

Putting this together, first consider the case where $u > F_\alpha^{-1}$. Thus, $\partial_u VaR_\alpha(u) = 0$ and

$$\begin{aligned} \partial_u ES_\alpha[X \wedge u] &= \tfrac{1}{1-\alpha} \left\{ \partial_u \mathrm{E}[X \wedge u] - \partial_u \mathrm{E}[X \wedge u \wedge z]|_{z=VaR_\alpha(u)} \right\} + 0 \\ &= \tfrac{1}{1-\alpha} \{1 - F(u) - 0\} = \tfrac{1-F(u)}{1-\alpha}, \end{aligned}$$

as desired. Now, consider the case where $u \leq F_\alpha^{-1}$. Now, Thus, $\partial_u VaR_\alpha(u) = 1$ and

$$\begin{aligned}\partial_u ES_\alpha[X\wedge u] &= \tfrac{1}{1-\alpha}\left\{\partial_u \mathrm{E}[X\wedge u] - \partial_u \mathrm{E}[X\wedge u\wedge z]|_{z=VaR_\alpha(u)}\right\}\\ &\quad +\{1-\tfrac{1}{1-\alpha}(1-F_{X\wedge u}[u;VaR_\alpha(u)])\}\times 1\\ &= \tfrac{1}{1-\alpha}\{1-F(u)-[1-F(u)]\}\\ &\quad +\{1-\tfrac{1}{1-\alpha}(1-F_{X\wedge u}[u;VaR_\alpha(u)])\}\\ &= 1-\tfrac{1}{1-\alpha}(1-F_{X\wedge u}[u;VaR_\alpha(u)]).\end{aligned}$$

When $u \leq F_\alpha^{-1}$, we have $F_{X\wedge u}[F^{-1}_{X\wedge u}(\alpha)] = F_{X\wedge u}[u] = 1$. Thus,

$$\begin{aligned}\partial_u\ ES_\alpha[X\wedge u] &= 1-\tfrac{1}{1-\alpha}(1-F_{X\wedge u}[u;VaR_\alpha(u)])\\ &= 1-\tfrac{1}{1-\alpha}(1-1) = 1,\end{aligned}$$

which is sufficient for the result.

17.5.8 Exercise 5.8. Special Case. Excess of Loss Value at Risk for Independent Uniformly Distributed Losses

For a continuous distribution, the VaR is that value y^* such that $\Pr(X_1\wedge u_1 + X_2\wedge u_2 \leq y^*) = \alpha$.

For the first case in [Exercise 5.2.1] where $y < u_2 < u_1$, $Pr[S(u_1,u_2)\leq y] = y^2/2 \times I(y\leq 1) + \{1-(2-y)^2/2\}\times I(1\leq y\leq 2)$. For small values of α, we have $y^* = VaR = \sqrt{2\alpha}$. For large values of α, we have

$$\begin{aligned}&\alpha = 1-(2-y^*)^2/2\\ \implies\ &2(1-\alpha) = (2-y^*)^2\\ \implies\ &y^* = 2-\sqrt{2(1-\alpha)}.\end{aligned}$$

Summarizing,

$$VaR = y^* = \begin{cases}\sqrt{2\alpha} & \alpha\leq 0.5\\ 2-\sqrt{2(1-\alpha)} & \alpha > 0.5.\end{cases}$$

From this, if $VaR < u_2$, then the VaR does not depend on u_1 nor u_2. This means that all derivatives are zero.

For the second case where $u_2 < y < u_1$, $Pr[S(u_1,u_2)\leq y] = u_2^2/2 + y - u_2$. Thus, we have $\alpha = u_2^2/2 + y^* - u_2$, so $VaR = y^* = \alpha + u_2 - u_2^2/2$. This does not depend on u_1. Further, we see that $\partial_{u_2} VaR = 1-u_2$ and $\partial^2_{u_2} VaR = -1$, so this is concave in u_2.

For the third case where $u_2 < u_1 < y$, $Pr[S(u_1,u_2)\leq y] = u_1u_2 - \frac{1}{2}(u_1+u_2-y)^2 + (y-u_1)(1-u_1) + (y-u_2)(1-u_2)$. Thus, we have

$$\begin{aligned}\partial_{u_1}\alpha &= \partial_{u_1}\left\{u_1u_2 - \tfrac{1}{2}(u_1+u_2-y^*)^2 + (y^*-u_1)(1-u_1) + (y^*-u_2)(1-u_2)\right\}\\ \implies 0 &= u_2 - (u_1+u_2-y^*)(1-\partial_{u_1}y^*)\\ &\qquad +(\partial_{u_1}y^*-1)(1-u_1) + (y^*-u_1)(-1) + \partial_{u_1}y^*(1-u_2)\\ &= \partial_{u_1}y^*\left[(u_1+u_2-y^*) + (1-u_1) + (1-u_2)\right] + u_2 - (u_1+u_2-y^*) - (1-u_1) - (y^*-u_1)\\ &= \partial_{u_1}y^*\left[2-y^*\right] + u_1 - 1\\ \implies\\ &\partial_{u_1}VaR = \tfrac{1-u_1}{2-y^*} > 0.\end{aligned}$$

For second derivatives, we have

$$\begin{aligned} 0 &= \partial_{u_1} y^* [2 - y^*] + u_1 - 1 \\ \implies 0 &= \partial_{u_1} \{\partial_{u_1} y^* [2 - y^*] + u_1 - 1\} \\ &= \partial^2_{u_1} y^* [2 - y^*] - (\partial_{u_1} y^*)^2 + 1 \\ \implies \partial^2_{u_1} VaR &= \tfrac{(\partial_{u_1} y^*)^2 - 1}{2 - y^*}. \end{aligned}$$

Using $\partial_{u_1} y^* = \frac{1-u_1}{2-y^*}$, we have

$$\begin{aligned} \partial^2_{u_1} VaR &= \tfrac{(1-u_1)^2 - (2-y^*)^2}{(2-y^*)^3} \\ &= \tfrac{(3-u_1-y^*)(y^*-u_1-1)}{(2-y^*)^3} < 0. \end{aligned}$$

Because $y^* - u_1 - 1 < (u_1 + u_2) - u_1 - 1 = u_2 - 1 < 0$. In addition,

$$\begin{aligned} 0 &= \partial_{u_1} y^* [2 - y^*] + u_1 - 1 \\ \implies 0 &= \partial_{u_2} \{\partial_{u_1} y^* [2 - y^*] + u_1 - 1\} \\ &= (\partial_{u_1} \partial_{u_2} y^*) [2 - y^*] - (\partial_{u_1} y^*)(\partial_{u_2} y^*) \\ &= (\partial_{u_1} \partial_{u_2} y^*) [2 - y^*] - (\tfrac{1-u_1}{2-y^*})(\tfrac{1-u_2}{2-y^*}) \\ \implies \partial_{u_1} \partial_{u_2} VaR &= \tfrac{(1-u_1)(1-u_2)}{(2-y^*)^3}. \end{aligned}$$

Consider the matrix

$$\begin{aligned} &\begin{pmatrix} \partial^2_{u_1} VaR & \partial_{u_1} \partial_{u_2} VaR \\ \partial_{u_1} \partial_{u_2} VaR & \partial^2_{u_2} VaR \end{pmatrix} \\ &= \begin{pmatrix} \frac{(3-u_1-y^*)(y^*-u_1-1)}{(2-y^*)^3} & \frac{(1-u_1)(1-u_2)}{(2-y^*)^3} \\ \frac{(1-u_1)(1-u_2)}{(2-y^*)^3} & \frac{(3-u_2-y^*)(y^*-u_2-1)}{(2-y^*)^3} \end{pmatrix} \\ &= \tfrac{1}{(2-y^*)^3} \begin{pmatrix} (3-u_1-y^*)(y^*-u_1-1) & (1-u_1)(1-u_2) \\ (1-u_1)(1-u_2) & (3-u_2-y^*)(y^*-u_2-1) \end{pmatrix}. \end{aligned}$$

17.5.9 Exercise 5.9. Constrained Optimization with an Equality Constraint

Start with equation (5.13). Taking derivatives, we get

$$\begin{aligned} AT_{1RC}(u) &= \partial_u AT_{RC}(u) = \partial_u RC_2^{-1}[RTC_{max} - RC_1(u)] \\ &= \tfrac{\partial_u (RTC_{max} - RC_1(u))}{RC_2'[RC_2^{-1}(RTC_{max} - RC_1(u))]} = \tfrac{-RC_1'(u)}{RC_2'[AT_{RC}(u)]}, \end{aligned}$$

as required. In the case of fair transfer costs, we have $RC_j'(u) = \partial_u RC_j(u) = -[1 - F_j(u)]$ for $j = 1, 2$. Thus,

$$AT_{1RC}(u) = -\frac{1 - F_1(u)}{1 - F_2[AT_{RC}(u)]}.$$

17.5.10 Exercise 5.10. Derivatives of the Risk Measure at the Equality Constraint

Instead of only the VaR, let us examine the derivative of the general retained risk function $RM[S(u, AT_{RC}(u))]$. First define the terms

$$\begin{aligned} &RM_1[S(u,u_2)] = \partial_u RM[S(u,u_2], \\ &RM_2[S(u_1,u] = \partial_u RM[S(u_1,u], \text{and} \\ &AT_{1RC}(u) = \partial_u AT_{RC}(u). \end{aligned}$$

With these terms, we have

$$\begin{aligned} \partial_u RM[S(u,u_2^*)] &= \partial_u RM[S(u,AT_{RC}(u))] \\ &= RM_1[S(u,AT_{RC}(u))] + RM_2[S(u,AT_{RC}(u))]AT_{1RC}(u) \\ &= RM_1[S(u,u_2^*)] + RM_2[S(u,u_2^*)]AT_{1RC}(u). \end{aligned}$$

To get insights, let us continue by examining the value at risk measure VAR. In Section 5.3.1, we developed

$$\begin{aligned} RM_1[S(u_1,u_2)] &= \partial_{u_1} VaR(u_1,u_2) = \tfrac{-1}{F_y[u_1,u_2,y]} F_{u_1}[u_1,u_2;y]\Big|_{y=VaR(u_1,u_2)} \\ RM_2[S(u_1,u_2)] &= \partial_{u_2} VaR(u_1,u_2) = \tfrac{-1}{F_y[u_1,u_2,y]} F_{u_2}[u_1,u_2;y]\Big|_{y=VaR(u_1,u_2)} \end{aligned}$$

where

$$\begin{aligned} F(u_1,u_2;y) &= F(u_1,u_2;y) = \Pr(X_1 \wedge u_1 + X_2 \wedge u_2 \leq y) \\ F_y(u_1,u_2;y) &= \partial_y\ F(u_1,u_2;y) \\ VaR(u_1,u_2) &= F_\alpha^{-1}(u_1,u_2) \\ F_{u_1}(u_1,u_2;y) &= -f_2(y-u_1)\left\{1 - C_2[F_1(u_1), F_2(y-u_1)]\right\} \\ F_{u_2}(u_1,u_2;y) &= -f_1(y-u_2)\left\{1 - C_2[F_2(u_2), F_1(y-u_2)]\right\}. \end{aligned}$$

This suffices for equation (5.16).

17.5.11 Exercise 5.11. Special Case. Identical Distributions.

Here, equation (5.16) becomes

$$\begin{aligned} &\partial_u VaR[u,u_2^*] \\ &= \tfrac{1}{F_y[u,u_2^*,y]} \left\{ f(y-u)\left\{1 - C_2[F(u), F(y-u)]\right\} \right. \\ &\qquad \left. -f(y-u_2^*)\left\{1 - C_2[F(u_2^*), F(y-u_2^*)]\right\} \tfrac{1-F(u)}{1-F(u_2^*)} \right\}, \end{aligned}$$

Of interest is the case where we choose u such that $u = u_2^*$ resulting in $u = RC^{-1}(RTC_{max}/2)$. In this case, $\partial_u VaR[u,u_2^*] = 0$. Note that this result does not rely upon the dependence.

This behavior is evident in Figure 5.13 for identical uniform distributions. Here, we see in the left-hand panel that the value of u with a zero derivative for the VaR is the same for all three dependence scenarios, about 0.55.

17.5.12 Exercise 5.12 Special Case. Independent Distributions.

For the case of independence, we have $C(u, u_2) = u_1 \cdot u_2$, $C_1(u_1, u_2) = u_2$, $C_2(u_1, u_2) = u_1$. Thus, equation (5.16) becomes

$$\begin{aligned}\partial_u VaR[u, u_2^*] &= \frac{1}{F_y[u,u_2^*,y]} \left\{ f_2(y-u) \left\{1 - F_1(u)\right\} \right. \\ &\quad \left. - f_1(y-u_2^*) \left\{1 - F_2(u_2^*)\right\} \frac{1-F_1(u)}{1-F_2(u_2^*)} \right\}, \\ &= \frac{1-F_1(u)}{F_y[u,u_2^*,y]} \left\{ f_2(y-u) - f_1(y-u_2^*) \right\},\end{aligned}$$

For the derivative to be zero, a solution occurs at values of u such that $f_2(y-u) = f_1(y-u_2^*)$. Very interesting.

17.6 Chapter Seven Exercise Solutions

17.6.1 Exercise 7.1. Quantile Regression Approach

a. The expected loss is

$$\begin{aligned}\mathrm{E}\ \phi_\alpha(Y - z_0) &= \tfrac{1}{2}\mathrm{E}|Y - z_0| + (\alpha - 0.5)\left[\mathrm{E}(Y) - z_0\right] \\ &= \tfrac{1}{2}\left[\int_{-\infty}^{z_0}(z_0 - z)dF(z) + \int_{z_0}^{\infty}(z - z_0)dF(z)\right] + (\alpha - 0.5)\left[\mathrm{E}(Y) - z_0\right].\end{aligned}$$

Using the Leibniz integral rule, on taking derivatives, we have

$$\begin{aligned}\partial_{z_0}\ \mathrm{E}\ \phi_\alpha(Y - z_0) &= \tfrac{1}{2}\left[\int_{-\infty}^{z_0}(1)dF(z) + \int_{z_0}^{\infty}(-1)dF(z)\right] + (\alpha - 0.5)\left[-1\right] \\ &= \tfrac{1}{2}\left[F(z_0) - \left\{1 - F(z_0)\right\}\right] - (\alpha - 0.5) = F(z_0) - \alpha.\end{aligned}$$

Equating this to zero yields the best value of z_0, $F^{-1}(\alpha)$.

b. Now, suppose that we have *i.i.d* replications of $Y_r = g(\mathbf{X}_r; \boldsymbol{\theta})$, $r = 1, \ldots, R$. Next, we take the minimum of all such solutions over different values of $\boldsymbol{\theta}$ while restricting consideration to a feasible set. Here is a summary of the quantile regression retention problem:

c. We can actually prove an even stronger result, that the relationship in equation (7.13) holds for each replication r. To ease notation, define the retained risk $Y_r = g(\mathbf{X}_r; \boldsymbol{\theta})$.

First note that $(y)_+ = \frac{|y|+y}{2}$ for all y. Thus, with $|y| = 2(y)_+ - y$, one can write $\phi_\alpha(y) = (y)_+ - (1-\alpha)y$. Using $Y_r - z_0$ for y, we have

$$\begin{aligned}\tfrac{1}{1-\alpha}\phi_\alpha(Y_r - z_0) &= \tfrac{1}{1-\alpha}\{(Y_r - z_0)_+ - (1-\alpha)(Y - z_0)\} \\ &\iff z_0 + \tfrac{1}{1-\alpha}(Y_r - z_0)_+ = Y_r + \tfrac{1}{1-\alpha}\phi_\alpha(Y_r - z_0).\end{aligned}$$

Taking an average over $r = 1, \ldots, R$ yields the expression in equation (7.13).

17.6.2 Exercise 7.2. No Kernel Smoothing: Simulated Expected Shortfall Derivatives.

Solution. To show part (a), we first establish

$$\begin{aligned}\partial_{z_0}\ [g(\mathbf{X}_r;\boldsymbol{\theta})-z_0]_+ &= \partial_{z_0}\ \begin{cases} 0 & \text{if } g(\mathbf{X}_r;\boldsymbol{\theta})\le z_0 \\ g(\mathbf{X}_r;\boldsymbol{\theta})-z_0 & \text{if } g(\mathbf{X}_r;\boldsymbol{\theta})\ge z_0 \end{cases} \\ &= \begin{cases} 0 & \text{if } g(\mathbf{X}_r;\boldsymbol{\theta})< z_0 \\ -1 & \text{if } g(\mathbf{X}_r;\boldsymbol{\theta})> z_0 \end{cases} \\ &= -I\left[g(\mathbf{X}_r;\boldsymbol{\theta})> z_0\right].\end{aligned}$$

This leads to the expression

$$\partial_{z_0}\ ES1_R(z_0,\boldsymbol{\theta}) = 1-\frac{1}{1-\alpha}\frac{1}{R}\sum_{r=1}^{R} I\left[g(\mathbf{X}_r;\boldsymbol{\theta})> z_0\right].$$

In the same way, we have that

$$\partial_{\boldsymbol{\theta}}\ [g(\mathbf{X}_r;\boldsymbol{\theta})-z_0]_+ = I\left[g(\mathbf{X}_r;\boldsymbol{\theta})> z_0\right]\partial_{\boldsymbol{\theta}} g(\mathbf{X}_r;\boldsymbol{\theta}).$$

Thus,

$$\partial_{\boldsymbol{\theta}}\ ES1_R(z_0,\boldsymbol{\theta}) = \frac{1}{1-\alpha}\frac{1}{R}\sum_{r=1}^{R} I\left[g(\mathbf{X}_r;\boldsymbol{\theta})> z_0\right]\partial_{\boldsymbol{\theta}} g(\mathbf{X}_r;\boldsymbol{\theta}),$$

which is sufficient for part (a).

For part (b) second derivatives, note that

$$\begin{aligned}\partial_{z_0}\ I\left[g(\mathbf{X}_r;\boldsymbol{\theta})> z_0\right] &= \partial_{z_0}\ \begin{cases} 0 & \text{if } g(\mathbf{X}_r;\boldsymbol{\theta})< z_0 \\ 1 & \text{if } g(\mathbf{X}_r;\boldsymbol{\theta})> z_0 \end{cases} \\ &= \begin{cases} 0 & \text{if } g(\mathbf{X}_r;\boldsymbol{\theta})< z_0 \\ 0 & \text{if } g(\mathbf{X}_r;\boldsymbol{\theta})> z_0 \end{cases} \\ &= 0.\end{aligned}$$

Thus, $\partial_{z_0}\partial_{z_0}\ ES1_R(z_0,\boldsymbol{\theta}) = 0$ and $\partial_{z_0}\partial_{\boldsymbol{\theta}}\ ES1_R(z_0,\boldsymbol{\theta}) = \mathbf{0}$.

From part (a), we have

$$\partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'}\ ES1_R(z_0,\boldsymbol{\theta}) = \frac{1}{1-\alpha}\frac{1}{R}\sum_{r=1}^{R} I\left[g(\mathbf{X}_r;\boldsymbol{\theta})> z_0\right]\partial_{\boldsymbol{\theta}} g(\mathbf{X}_r;\boldsymbol{\theta})\partial_{\boldsymbol{\theta}'} g(\mathbf{X}_r;\boldsymbol{\theta}).$$

This is sufficient for part (b).

17.6.3 Exercise 7.3. Quota Sharing of ANU Risks

```
# Exercise 7.3 Illustrative Code
library(CVXR)
load(file = "../Data/SimData/AggClaimsMay100Kw0.7.Rdata")

#save(ANURTCmax.vec, OutMatANU.ES,
#     file= "../ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")
load(file= "../ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")
                   # Bring in ANURTCmax.vec

Xsim    <- Claims[,-1]
ANU.mu  <- colMeans(Xsim, na.rm = FALSE)
numVars <- ncol(Xsim)
nsim    <- nrow(Xsim)
TotCostANU <- sum(ANU.mu)
Sigma   <- cov(Xsim)
alpha   <- 0.90

c.names <- paste("c", 1:numVars, sep="")
numPrint = 5+min(3, numVars)

pos_part <- function(z) { 0.5 * ( z + abs(z) ) }

SumCVXR <- function(coeff){
  theta    <- coeff[-1]
  RTC      <- TotCostANU - t(theta) %*% ANU.mu
  Variance <- t(theta) %*% Sigma %*% theta
  Claims.retained <- Xsim %*% theta
  VaR      <- quantile(Claims.retained, prob = alpha)
  CTE      <- VaR + mean(pmax(Claims.retained - VaR, 0)) / (1-alpha)
  output   <- c ( RTC, VaR, CTE, sqrt(Variance) )
  return(output)
  }

time1 <- Sys.time()
OutMat.Q.VaR <- matrix(0,nrow = length(ANURTCmax.vec), ncol = 5+numVars)
colnames(OutMat.Q.VaR) <- c("RTCmax","RTC", "VaR", "ES", "Std Dev", c.names)

for (jCost in 1:length(ANURTCmax.vec)){
  coeff   <- Variable(numVars+1)
  VaR.z0        <- coeff[1]
  theta.vec     <- coeff[-1]
  Claims.retained <- Xsim %*% theta.vec
  RTC           <- TotCostANU - t(theta.vec) %*% ANU.mu
  RTCmax        <- ANURTCmax.vec[jCost]
  constraints   <- list(RTC <= RTCmax,
                        coeff >= 0, theta.vec <= 1)
  objES         <- VaR.z0 + mean( pos_part(Claims.retained  - VaR.z0) ) / (1-alpha)
  probES        <- Problem(Minimize(objES), constraints)
  CVXR.ESresult <- solve(probES, solver = "ECOS")
```

```
  coeffES.opt   <- as.vector(CVXR.ESresult$getValue(coeff))
  OutMat.Q.VaR[jCost,]  <- c(RTCmax, SumCVXR(coeffES.opt), coeffES.opt[-1])

 cat("CVXR results", OutMat.Q.VaR[jCost,1:8], "\n")
}

save(OutMat.Q.VaR,
     file = "../OnlineSupp/Chap7/Solutions/Exercise73Soln.Rdata")
Sys.time() - time1
#  Based on nsim = 100000  ##Time difference of 3.959546 mins
```

17.6.4 Exercise 7.4. Quota Sharing of ANU Risks Minimizing the Variance

```
# Exercise 74 Illustrative Code
library(CVXR)
load(file = "../Data/SimData/AggClaimsMay100Kw0.7.Rdata")

#save(ANURTCmax.vec, OutMatANU.ES,
#     file= "../ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")
load(file= "../ChapTablesData/Chap7/OutMatANUES.Feb2024.Rdata")
                   # Bring in ANURTCmax.vec

Xsim    <- Claims[,-1]
ANU.mu  <- colMeans(Xsim, na.rm = FALSE)
numVars <- ncol(Xsim)
nsim    <- nrow(Xsim)
TotCostANU <- sum(ANU.mu)
Sigma   <- cov(Xsim)
alpha   <- 0.90

c.names <- paste("c", 1:numVars, sep="")
numPrint = 5+min(3, numVars)

pos_part <- function(z) { 0.5 * ( z + abs(z) ) }

SumCVXRvar <- function(coeff){
  theta    <- coeff
  RTC      <- TotCostANU - t(theta) %*% ANU.mu
  Variance <- t(theta) %*% Sigma %*% theta
  Claims.retained <- Xsim %*% theta
  VaR      <- quantile(Claims.retained, prob = alpha)
  CTE      <- VaR + mean(pmax(Claims.retained - VaR, 0)) / (1-alpha)
  output   <- c ( RTC, VaR, CTE, sqrt(Variance) )
  return(output)
  }

time1 <- Sys.time()
OutMat.var <- matrix(0,nrow = length(ANURTCmax.vec), ncol = 5+numVars)
colnames(OutMat.var) <- c("RTCmax","RTC", "VaR", "ES",
```

```
                          "Std Dev", c.names)

for (jCost in 1:length(ANURTCmax.vec)){
  coeff         <- Variable(numVars)
  theta.vec     <- coeff
  Claims.retained <- Xsim %*% theta.vec
  RTC           <- TotCostANU - t(theta.vec) %*% ANU.mu
  RTCmax        <- ANURTCmax.vec[jCost]
  constraints   <- list(RTC <= RTCmax,
                        coeff >= 0, theta.vec <= 1)
  objvar         <- quad_form(theta.vec, Sigma)
  probvar        <- Problem(Minimize(objvar), constraints)
  CVXR.varresult <- solve(probvar, solver = "ECOS")
  coeffvar.opt   <- as.vector( CVXR.varresult$getValue(coeff) )
  OutMat.var[jCost,]  <- c(RTCmax, SumCVXRvar(coeffvar.opt), coeffvar.opt )

 cat("CVXR results", OutMat.var[jCost,1:8], "\n")
}

save(OutMat.var, file = "../OnlineSupp/Chap7/Solutions/Exercise74Soln.Rdata")
Sys.time() - time1
#  Based on nsim = 100000  ##Time difference of 5.15878 secs

OutMat.varA <- cbind(round(OutMat.var[,1:5],digits=1), OutMat.var[,6:8])

colnames(OutMat.varA) <- c("$RTC_{max}$","$RTC$", "$VaR$", "$ES$",
                           "$Std$ $Dev$", "$c_1$", "$c_2$", "$c_3$" )

TableGen1(TableData=OutMat.varA,
         TextTitle='ANU Quota Share Variance Optimization
                    Results using CVXR',
         Align='r', Digits=3, ColumnSpec=1:5,
         BorderRight= 5, ColWidth=ColWidth7)
```

17.6.5 Exercise 7.5. Comparing Quota Sharing of ANU Risks Using Different Objective Functions

```
# Exercise 7.5 Illustrative Code
#save(OutMat.Q.VaR,
#     file = "../OnlineSupp/Chap7/Solutions/Exercise73Soln.Rdata")
load(file = "OnlineSupp/Chap7/Solutions/Exercise73Soln.Rdata")

#save(OutMat.var,
#     file = "../OnlineSupp/Chap7/Solutions/Exercise74Soln.Rdata")
load(file = "OnlineSupp/Chap7/Solutions/Exercise74Soln.Rdata")

colnames(OutMat.Q.VaR)[4] <- c("ES")
colnames(OutMat.var)[4] <- c("ES")
```

```
par(mfrow=c(2,2))
  plot(OutMat.Q.VaR[,"VaR"], OutMat.var[,"VaR"],
       xlab="VaR from Minimizing ES", ylab="VaR: Min StDev")
    abline(0,1)
  plot(OutMat.Q.VaR[,"ES"], OutMat.var[,"ES"],
       xlab="ES from Minimizing ES", ylab="ES: Min StDev")
    abline(0,1)
  plot(OutMat.Q.VaR[,"c1"], OutMat.var[,"c1"],
       xlab=expression(c[1] ~from~Minimizing ~ES),
       ylab=expression(c[1] ~from ~Minimizing ~StdDev) )
    abline(0,1)
  plot(OutMat.Q.VaR[,"c2"], OutMat.var[,"c2"],
       xlab=expression(c[2] ~from ~Minimizing ~ES),
       ylab=expression(c[2] ~from ~Minimizing ~StdDev))
    abline(0,1)
```

17.6.6 Exercise 7.6. Kernel Smoothing of Expectations, with Derivatives

Solution. To show part (a), we start with a function $h(\cdot)$ and use a kernel density estimator $k(\cdot)$ to define $\mathrm{E}_{Rk}[h(Y)]$. Then with a change of variables $z = (y - Y_r)/b$. Thus, we can approximate $\mathrm{E}[h(Y)]$ using

$$\begin{aligned}
\mathrm{E}_{Rk}[h(Y)] &= \int h(y) f_{Rk}(y) dy \\
&= \frac{1}{R\,b} \sum_{r=1}^{R} \int h(y) k\left(\tfrac{y - Y_r}{b}\right) dy \\
&= \frac{1}{R} \sum_{r=1}^{R} \int h[Y_r + bz]\; k(z) dz \\
&= \int \left\{ \frac{1}{R} \sum_{r=1}^{R} h[Y_r + bz] \right\}\; k(z) dz.
\end{aligned}$$

as in equation (7.5).

b. As in part (a), use a change of variable $z = [g(\mathbf{X}_r; \boldsymbol{\theta}] - Y_r)/b$, to get

$$\begin{aligned}
ES1_{Rk}(z_0, \boldsymbol{\theta}) &= z_0 + \tfrac{1}{1-\alpha} \left\{ \mathrm{E}_{Rk}[g(\mathbf{X}; \boldsymbol{\theta})] - \mathrm{E}_{Rk}[g(\mathbf{X}; \boldsymbol{\theta}) \wedge z_0] \right\} \\
&= z_0 + \tfrac{1}{1-\alpha} \left\{ \mathrm{E}_{Rk}[g(\mathbf{X}; \boldsymbol{\theta}) - z_0]_+ \right\} \\
&= z_0 + \tfrac{1}{1-\alpha} \tfrac{1}{R} \sum_{r=1}^{R} \int [g(\mathbf{X}_r; \boldsymbol{\theta}) + bz - z_0]_+ \; k(z) dz \\
&= z_0 + \tfrac{1}{1-\alpha} \int \left\{ \frac{1}{R} \sum_{r=1}^{R} [g(\mathbf{X}_r; \boldsymbol{\theta}) + bz - z_0]_+ \right\} \; k(z) dz,
\end{aligned}$$

as in equation (7.6).

c. As in parts (a) and (b) of Exercise 7.1.2, first note that

$$\begin{aligned}
\partial_{z_0} \; [g(\mathbf{X}_r; \boldsymbol{\theta}) + bz - z_0]_+ &= -I\left(z > [z_0 - g(\mathbf{X}_r; \boldsymbol{\theta})]/b\right) \\
\text{and} \\
\partial_{\boldsymbol{\theta}} \; [g(\mathbf{X}_r; \boldsymbol{\theta}) + bz - z_0]_+ &= I\left(z > [z_0 - g(\mathbf{X}_r; \boldsymbol{\theta})]/b\right) \partial_{\boldsymbol{\theta}} g(\mathbf{X}_r; \boldsymbol{\theta}).
\end{aligned}$$

Using the observation that $\int I(z > c) k(z) dz = 1 - K(c)$ for a constant c, we have the partial derivatives

$$\partial_{z_0} \; ES1_{Rk}(z_0, \boldsymbol{\theta}) = 1 - \frac{1}{1-\alpha}\frac{1}{R}\sum_{r=1}^{R}\left\{1 - K\left(\frac{z_0 - g(\mathbf{X}_r;\boldsymbol{\theta})}{b}\right)\right\}$$

and

$$\partial_{\boldsymbol{\theta}} \; ES1_{Rk}(z_0, \boldsymbol{\theta}) = \frac{1}{1-\alpha}\frac{1}{R}\sum_{r=1}^{R}\left\{1 - K\left(\frac{z_0 - g(\mathbf{X}_r;\boldsymbol{\theta})}{b}\right)\right\}\partial_{\boldsymbol{\theta}} g(\mathbf{X}_r;\boldsymbol{\theta}).$$

which is sufficient for the result.

17.6.7 Exercise 7.7. Kernel Smoothed Expected Shortfall Hessian

Solution. The hessian can be written as

$$\partial_{\mathbf{z}}\partial_{\mathbf{z}'} \; ES1_{Rk}(\mathbf{z}) = \begin{pmatrix} \partial_{z_0}\partial_{z_0} \; ES1_{Rk}(z_0, \boldsymbol{\theta}) & \partial_{z_0}\partial_{\boldsymbol{\theta}'} \; ES1_{Rk}(z_0, \boldsymbol{\theta}) \\ \partial_{\boldsymbol{\theta}}\partial_{z_0} \; ES1_{Rk}(z_0, \boldsymbol{\theta}) & \partial_{\boldsymbol{\theta}}\partial_{\boldsymbol{\theta}'} \; ES1_{Rk}(z_0, \boldsymbol{\theta}) \end{pmatrix}.$$

For second derivatives, we have

$$\begin{aligned}
\partial_{z_0 z_0} \; ES1_{Rk}(z_0) &= \tfrac{1}{1-\alpha}\tfrac{1}{Rb}\textstyle\sum_{r=1}^{R}\left\{k\left(\frac{z_0 - g(\mathbf{X}_r;\boldsymbol{\theta})}{b}\right)\right\} \\
\partial_{z_0\boldsymbol{\theta}} \; ES1_{Rk}(z_0) &= -\tfrac{1}{1-\alpha}\tfrac{1}{Rb}\textstyle\sum_{r=1}^{R}\left\{k\left(\frac{z_0 - g(\mathbf{X}_r;\boldsymbol{\theta})}{b}\right)\partial_{\boldsymbol{\theta}} g(\mathbf{X}_r;\boldsymbol{\theta})\right\} \\
\partial_{\boldsymbol{\theta}\boldsymbol{\theta}'} \; ES1_{Rk}(z_0) &= \tfrac{1}{1-\alpha}\tfrac{1}{R}\textstyle\sum_{r=1}^{R}\left[\left\{1 - K\left(\frac{z_0 - g(\mathbf{X}_r;\boldsymbol{\theta})}{b}\right)\right\}\partial_{\boldsymbol{\theta}\boldsymbol{\theta}'} g(\mathbf{X}_r;\boldsymbol{\theta})\right. \\
&\qquad \left. + \tfrac{1}{b}\, k\left(\frac{z_0 - g(\mathbf{X}_r;\boldsymbol{\theta})}{b}\right)\left\{\partial_{\boldsymbol{\theta}} g(\mathbf{X}_r;\boldsymbol{\theta})\right\}\partial_{\boldsymbol{\theta}'} g(\mathbf{X}_r;\boldsymbol{\theta})\right].
\end{aligned}$$

17.6.8 Exercise 7.8. Value at Risk with Kernel Smoothing

Solution. As we saw in Section 5.3.1, we may write its sensitivities as

$$\partial_{\boldsymbol{\theta}} VaR_{Rk} = -\frac{\partial_{\boldsymbol{\theta}} F_{Rk}[y]}{f_{Rk}[y]}\bigg|_{y=VaR_{Rk}}.$$

In addition,

$$\begin{aligned}
\partial_{\boldsymbol{\theta}} F_{Rk}[y] &= \tfrac{1}{R}\textstyle\sum_{r=1}^{R}\partial_{\boldsymbol{\theta}} \; K\left(\frac{y - g(\mathbf{X}_r;\boldsymbol{\theta})}{b}\right) \\
&= -\tfrac{1}{Rb}\textstyle\sum_{r=1}^{R}\left\{\; k\left(\frac{y - g(\mathbf{X}_r;\boldsymbol{\theta})}{b}\right)\partial_{\boldsymbol{\theta}} g(\mathbf{X}_r;\boldsymbol{\theta})\right\}.
\end{aligned}$$

This yields

$$\partial_{\boldsymbol{\theta}} VaR_{Rk} = \frac{\sum_{r=1}^{R}\left\{\; k\left(\frac{y - g(\mathbf{X}_r;\boldsymbol{\theta})}{b}\right)\right\}\partial_{\boldsymbol{\theta}} g(\mathbf{X}_r;\boldsymbol{\theta})}{\sum_{r=1}^{R} k\left(\frac{y - g(\mathbf{X}_r;\boldsymbol{\theta})}{b}\right)}\bigg|_{y=VaR_{Rk}}.$$

With an excess of loss policy, we have

$$\partial_{\mathbf{u}} VaR_{Rk} = \left. \frac{\sum_{r=1}^{R} \left\{ k\left(\frac{y-S(\mathbf{u})}{b}\right) \begin{pmatrix} I(X_{1r} > u_1) \\ \vdots \\ I(X_{pr} > u_p) \end{pmatrix} \right\}}{\sum_{r=1}^{R} k\left(\frac{y-S(\mathbf{u})}{b}\right)} \right|_{y=VaR_{Rk}},$$

as required.

17.6.9 Exercise 7.9. Bivariate Excess of Loss Using ES Optimization

```
# BidataGenProcedure
#  Set Simulation Parameters
Bignsim <- 250000
set.seed(202020)

#  Normal, or Gaussian, Copula
corrparams <- c(0.5)
norm.cop <- copula::normalCopula(param=corrparams,
                                 dim = 2,'dispstr' ="un")
UCop     <- copula::rCopula(Bignsim, norm.cop)

BiClaims = matrix(0, nrow = Bignsim, ncol = 2)
BiClaims[,1] = q1(UCop[,1])
BiClaims[,2] = q2(UCop[,2])
```

```
# Exercise 7.9 Illustrative Code

BiCstLevNum <- 1 - c(0.05, seq(0.1, 0.9, length.out=5),0.95)
BiCstLevNum <- round(BiCstLevNum*100)/100
TotCost     <- sum(colMeans(BiClaims))
RTCmax.vec  <- TotCost *BiCstLevNum

nsim    <- 5000
alpha   <- 0.80
bw      <- 1
numBoot <- 10
Xsim    <- BiClaims[1:(nsim*numBoot),]  #  For resample

starter.coeff <- rep(10, 3)
Output.array <- array(0, dim = c(length(RTCmax.vec),
                                 1 + 6 + 2 + 1, numBoot))

time1 <- Sys.time()
```

```
for (rBoot in 1:numBoot){
    block <- seq(from = (rBoot-1)*nsim+1,
                 to = rBoot*nsim, by = 1) #  For resample
    XsimBlock <- Xsim[block,]
    Output.array[,,rBoot]  <-
       ThetaOptim.fct(Xsim = XsimBlock, ExLoss = TRUE,
                      p.z = 3, RTCmax.vec=RTCmax.vec, starter.coeff)
    } #  end rBoot Loop
save(Output.array, file="../ChapTablesData/Chap7/Exer761.Rdata")

Sys.time() - time1
# For Resample Boot=10, nsim=10000, Time difference of  5.380804 mins
```

```
# Exercise 7.9 Illustrative Code, Figure 7.7
#save(Output.array,
#     file="../ChapTablesData/Chap7/Exer761.Rdata")
load(file="ChapTablesData/Chap7/Exer761.Rdata")

numBoot <- dim(Output.array)[3]
Output.mat <- Output.array[,,1]
for (rBoot in 2:numBoot){
   Output.mat <- rbind(Output.mat , Output.array[,,rBoot])}
Output.matplot <- Output.mat[,c(1, 2, 6, 8, 9)]
colnames(Output.matplot) <- c("RTCmax", "RTC", "ES", "u1", "u2")
dfPlot <- data.frame(Output.matplot)

p1 <- ggplot(dfPlot, aes(x=RTC, y=ES)) + geom_point() + theme_bw() +
         scale_y_continuous(limits = c(-10,10000)) + labs(y = 'ES')
p2 <- ggplot(dfPlot, aes(x=RTC, y=u1)) + geom_point() + theme_bw() +
              scale_y_continuous(limits = c(-10,10000)) +
              labs( y = expression(u[1]) )
p3 <- ggplot(dfPlot, aes(x=RTC, y=u2)) + geom_point() + theme_bw() +
              scale_y_continuous(limits = c(-10,10000)) +
              labs( y = expression(u[2]) )
grid.arrange(p1,p2,p3,nrow=1)
```

17.6.10 Appendix 7.7.3. Starting Values for Multivariate Excess of Loss

```
# Appendix 7.7.3 Illustrative Code
#  Bring in Simulated Claims Data

load(file = "../Data/SimData/AggClaimsMay100Kw0.7.Rdata")

OutRescaleMethod <- matrix(0,nrow = length(RTCmax.vec), ncol = numVars+1)
u.names <- paste("u",2:(1+numVars), sep="")
colnames(OutRescaleMethod) <- c("LMEs", u.names)

#  Rescale by Standard Deviation, `ANU.stdDev`
```

```
skale <- ANU.stdDev/10
vars <- 2:15

HjinvsTweedie <- function(LME, sd, mu.fct, phi.fct){
         dfFs   <- function(x){tweedie::ptweedie(x*sd, power=power,
                                                 mu=mu.fct, phi=phi.fct)}
         Hs     <- function(u){integrate(dfFs, lower = 0, upper = u)$value }
         Hstars <- function(x){Hs(x)-LME}
         return(uniroot(Hstars, lower = 0, upper = 1e6)$root)}
HjinvsNotTweedie <- function(LME, sd,  Claims, LineNum){
         Hs     <- function(u){u - mean( pmin(u, Claims[,LineNum]/sd) ) }
         Hstars <- function(x){Hs(x)-LME}
         return(uniroot(Hstars, lower = 0, upper = 1e6)$root)}

sdavg <- mean(skale[vars])

for (iRTC in 1:length(RTCmax.vec)){
  RTCmaxs = RTCmax.vec[iRTC]/ sdavg
  IndExsLoss <- function(LME, RTCmaxs){
    sumHjinvs <- 0
    for (jRisk in TweedieClaims[-1]){sumHjinvs = sumHjinvs +
      HjinvsTweedie(LME/2, sd =skale[jRisk], mu.fct=ANU.mu[jRisk],
                                          phi.fct=ANU.phi[jRisk])
    }
    for (jRisk in NotTweedieClaims){sumHjinvs = sumHjinvs +
      HjinvsNotTweedie(LME/2, sd =skale[jRisk], Claims, LineNum=jRisk)
    }
    target <- sumHjinvs  - length(vars)*LME/2 -
              (sum(ANU.mu[vars]/skale[vars]) - RTCmaxs)
    return(target)
    }
  IndExsLoss1 <- function(LME){IndExsLoss(LME, RTCmaxs)}
  upperANU = 1e4
  LMEs.opt <- uniroot(IndExsLoss1, lower = 0, upper = upperANU)$root
 # OutRescaleMethod[iRTC,1] <- RTCmax.vec[iRTC]
  OutRescaleMethod[iRTC,1] <- LMEs.opt

  ustart1 <- rep(0, length(vars))
  for (jRisk in TweedieClaims[-1]){
    ustart1[jRisk-1] <- HjinvsTweedie(LMEs.opt/2, sd =skale[jRisk],
                                mu.fct=ANU.mu[jRisk],phi.fct=ANU.phi[jRisk])
  }
    for (jRisk in NotTweedieClaims){
    ustart1[jRisk-1] <- HjinvsNotTweedie(LMEs.opt/2, sd =skale[jRisk],
                                 Claims, LineNum=jRisk)
    }
  ustart = ustart1* skale[-1]
  OutRescaleMethod[iRTC,-1] <- ustart
}
```

```
save(OutRescaleMethod,
     file= "../ChapTablesData/Chap7/OutRescaleMethod.Rdata")
```

17.7 Chapter Eight Exercise Solutions

17.7.1 Exercise 8.1. Wisconsin Property Fund Seeks Reinsurance Protection

```
# Exercise81Soln
load(file="../Data/WiscPropFundData/dataout.Rdata")
# Coverage is in millions, not in logs
Coverage <- cbind(dataout$CoverageBC, dataout$CoverageIM,
                  dataout$CoverageCN, dataout$CoverageCO,
                  dataout$CoveragePN, dataout$CoveragePO  )
colnames(Coverage) <- c("BC", "IM", "CN", "CO", "PN", "PO")
rm(dataout)

bw    <- 1
alpha <- 0.80

#save(Prop.array,
#     file = "../Data/WiscPropFundData/SimPropArray.RData")
load(file = "../Data/WiscPropFundData/SimPropArray.RData")

nsim  <- length(Prop.array[1,,1])
npol  <- length(Prop.array[1,1,])
nrisk <- length(Prop.array[,1,1])

SummaryOut <-cbind(rowSums(Prop.array[1,,]), rowSums(Prop.array[2,,]),
                   rowSums(Prop.array[3,,]), rowSums(Prop.array[4,,]),
                   rowSums(Prop.array[5,,]), rowSums(Prop.array[6,,]) )
dfSummaryOut <- data.frame(SummaryOut)

TotCost   <- sum(colMeans(dfSummaryOut))
CstLev    <- 1 - seq(0.05, 0.95, by = 0.10)
CstLev    <- round(CstLev*100)/100
PropRTCmax.vec <- TotCost * CstLev
starter.coeff  <- c(10, rep(0.1, 6) )

Time1 <- Sys.time()
OutMatPropExer.ES <- ThetaOptim.fct(Xsim=SummaryOut, ExLoss = TRUE, p.z = 7,
                                    RTCmax.vec=PropRTCmax.vec, starter.coeff)
Sys.time() - Time1   #Time difference of 4.220549 mins
```

17.7.2 Exercise 8.2. Excess of Loss Frontier Interpretation

a. First consider risk transfer costs. For these upper limits, the limited value is $\mathrm{E}(X_j \wedge u_j) = u_j$. Further, the expected excess is $\mathrm{E}[X_j - u_j]_+ = \mathrm{E}(X_j) - \mathrm{E}(X_j \wedge u_j) = \mathrm{E}(X_j) - u_j$. Let $SUM_u = \sum_{j=1}^p u_j$ be the sum of upper limits. With this notation, the budget constraint is

$$\begin{array}{ll} & \sum_{j=1}^p \mathrm{E}[X_j - u_j]_+ \le RTC_{max} \\ \iff & \sum_{j=1}^p \mathrm{E}(X_j) - SUM_u \le RTC_{max} \\ \iff & SUM_u \ge \sum_{j=1}^p \mathrm{E}(X_j) - RTC_{max} = DELTA_{RTC}. \end{array}$$

Without the kernel approximations, the expected shortfall objective function is

$$\begin{aligned} f_0(\mathbf{z}) &= ES1_R(z_0, \boldsymbol{\theta}) \\ &= z_0 + \tfrac{1}{(1-\alpha)R} \sum_{r=1}^R \left\{ \sum_{j=1}^p X_{jr} \wedge u_j - z_0 \right\}_+ \\ &= z_0 + \tfrac{1}{1-\alpha} \left\{ \sum_{j=1}^p u_j - z_0 \right\}_+ \\ &= z_0 + \tfrac{1}{1-\alpha} \left\{ SUM_u - z_0 \right\}_+, \end{aligned}$$

Thus, the problem is to

$$\boxed{\begin{array}{ll} \text{minimize}_{z_0, \mathbf{u}} & z_0 + \tfrac{1}{1-\alpha} \left\{ SUM_u - z_0 \right\}_+ \\ \text{subject to} & SUM_u \ge DELTA_{RTC}. \end{array}}$$

For a fixed value of z_0, the objective function is nondecreasing in SUM_u and so one minimizes this by taking the smallest possible value subject to the constraint, so $SUM_u = DELTA_{RTC}$. For this value, it is simple to show that we can minimize $z_0 + \frac{1}{1-\alpha} \left\{ SUM_u - z_0 \right\}_+$ by choosing $z_0 = SUM_u$. For this value of z_0, the objective function is also SUM_u.

Thus, **any** set of upper limits that satisfy $SUM_u = DELTA_{RTC}$ yields the best value at risk and expected shortfall, $z_0^* = DELTA_{RTC}$. This is sufficient for the result.

b. When do the results of part (a) apply?

We know that by allowing RTC_{max} to approach $\sum_{j=1}^p \mathrm{E}(X_j)$ that all the upper limits become zero (in a fair world of risk transfer costs), so we can think about situations where large values of RTC_{max} mean that $DELTA_{RTC}$ is sufficiently small so that the feasible region $\{u_1 \le a_1, \ldots, u_p \le a_p\} : SUM_u \ge DELTA_{RTC}$ is not empty. So, we are likely to observe this behavior for large values of RTC_{max}.

Further, when the results do apply, this means that retained risks $\sum_{j=1}^p X_j \wedge u_j = \sum_{j=1}^p u_j$ are not random, so that the standard deviation is zero and the $VaR = ES$.

Examining Table 8.12, we see that the results likely apply for $RTC_{max} \ge 10642$ and do not hold for $RTC_{max} \le 3547$. For intermediate optimization problems, there are likely to be pockets of regions were upper limits can not be uniquely determined.

17.7.3 Exercise 8.3. An Attractive Naive Portfolio

```
# Exercise83Soln
PolNum = 2
#save(Prop.array,
#     file = "../Data/WiscPropFundData/SimPropArray.RData")
load(file = "../Data/WiscPropFundData/SimPropArray.RData")
Xsim <- t(Prop.array[,,PolNum])

nsim = 10000

LimitClaims.fct <- function(u.vec,indices){
  limClaims <- matrix(0, nrow = nsim, ncol = 6)
  for (jClaims in 1:6){
  limClaims[,jClaims] <- pmin(u.vec[jClaims], Xsim[indices,jClaims])}
  return(limClaims)
}

UnLimClaims <- LimitClaims.fct(rep(1e10, 6),1:nsim)

QuanRM.fct <- function(coeff){
  VaR.x   <- coeff[1]
  u.vec   <- coeff[-1]
  LimClaims <- LimitClaims.fct(u.vec,1:nsim)
  temp    <- colMeans(UnLimClaims) - colMeans(LimClaims)
  RTC.R   <- sum(temp)
  AggY    <- rowSums(LimClaims)
  VaR     <- as.numeric(quantile(AggY, prob = alpha))
  ESmin  <- as.numeric(VaR.x + ( mean(pmax(AggY - VaR.x, 0)) )/(1-alpha) )
  ES     <- as.numeric(VaR   + ( mean(pmax(AggY- VaR,   0)) )/(1-alpha) )
  stddev  <- sd(AggY)
  return(c(RTC.R, VaR, ESmin, ES, stddev))
}

SummaryOut <- t(Xsim)

alphaVec <- seq(0.5, 0.95, length.out=11)
OutMat.Exer <- matrix(0,nrow = length(alphaVec), ncol = 5)
for (jAlpha in 1:length(alphaVec)){
NaivePortAlpha <- c(1,quantile(p = alphaVec[jAlpha], SummaryOut[1,]),
                      quantile(p = alphaVec[jAlpha], SummaryOut[2,]),
                      quantile(p = alphaVec[jAlpha], SummaryOut[3,]),
                      quantile(p = alphaVec[jAlpha], SummaryOut[4,]),
                      quantile(p = alphaVec[jAlpha], SummaryOut[5,]),
                      quantile(p = alphaVec[jAlpha], SummaryOut[6,])
                      )
OutMat.Exer[jAlpha,] <- c(alphaVec[jAlpha] ,QuanRM.fct(NaivePortAlpha)[-3])
}

save(OutMat.Exer,
    file = "../OnlineSupp/Chap8/Solutions/Exercise83Soln.Rdata")
```

17.7.4 Exercise 8.7. Varying the Cyber Risk Premium

Interpretation. The result reported earlier in Table 8.3 corresponds to $CyberLoad = 1$. When $CyberLoad = 0.5$, the premium for this risk is only half of the fair price and the resulting optimal upper limit is 0, corresponding to full insurance. When the value of $CyberLoad = 2$, the premium is double the fair price and the resulting optimal upper limit exceeds the 99th percentile (see Table 8.14), close to full retention.

```
#  Exercise87Soln
load(file = "SimDataMay/AggClaimsMay100Kw0.7.Rdata")

nsim.Load <- 10000
Claims.Load <- Claims[1:nsim.Load,]
TotCost   <- sum(colMeans(Claims.Load[,-1]))
RTCmaxFix = round(TotCost * 0.5,digits = 0)

QuanRMPropLoad.fct <- function(CyberLoad, coeff){
  VaR.x  <- coeff[1]
  u.vec  <- c(uPropFixed, coeff[-1])  #property is fixed
  LimClaims <- Claims.Load
  for (jVar in 1:ncol(Claims.Load)){
    LimClaims[,jVar] = pmin(Claims.Load[,jVar],u.vec[jVar])
    }
  temp <- colMeans(Claims.Load) -colMeans(LimClaims)
  temp1 <- temp[-1]              # do not include property
  RTC.R <- sum(temp1[-2]) +  CyberLoad * temp1[2]
  AggY <- rowSums(LimClaims)
  VaR   <- as.numeric(quantile(AggY, prob = alpha))
  CTEmin<- as.numeric(VaR.x + ( mean(pmax(AggY - VaR.x, 0)) )/(1-alpha) )
  CTE   <- as.numeric(VaR   + ( mean(pmax(AggY - VaR,   0)) )/(1-alpha) )
  stddev  <- sd(AggY)
  return(c(RTC.R, VaR, CTEmin, CTE, stddev))
}

time1 <- Sys.time()

load(file = "Data/OutMatProp.CTE.Rdata")

starter.u <- as.numeric(OutMatProp.CTE[6, 6:19])
VaR.initial <- as.numeric(OutMatProp.CTE[6, 3])
starter.coeff <- c(VaR.initial,starter.u)

Loadvec <- c(seq(from=0.5, to= 2.5, by= 0.5),3)
OutMat.Load  <- matrix(0,nrow = length(Loadvec), ncol = numVars+5)
u.names <- paste("u",2:(1+numVars), sep="")
colnames(OutMat.Load) <- c("Cyber Load",  "RTC", "VaR", "ES", "StdDev", u.names)

starter.Load <- starter.coeff

for (jCost in 1:length(Loadvec)){
```

```
  CyberLoad <-Loadvec[jCost]
  if (jCost > 1 ){starter.Load <- Load.opt$par}
  RTC.Load <- function(coeff){QuanRMPropLoad.fct(CyberLoad,coeff)[1]}
  f0.Load <- function(coeff){QuanRMPropLoad.fct(CyberLoad,coeff)[3]}
  hin.Load <- function(coeff) {h <- NA
    h[1] <- RTCmaxFix - RTC.Load(coeff)
    for (j in 1:numVars){h[j+1] <- coeff[j+1] }
    return(h)}
  Load.opt <- alabama::auglag(par=starter.Load,
           fn=f0.Load,
           hin=hin.Load,
           control.outer=list(method="nlminb",trace=FALSE))
  uparams <- Load.opt$par[-1]
  output <- c(CyberLoad, QuanRMPropLoad.fct(CyberLoad,Load.opt$par)[-3], uparams)
  cat("alabama results",output, "\n")
  OutMat.Load[jCost,] <- output

}

save(OutMat.Load, file = "Data/OutMat.Load.Rdata")

Sys.time() - time1
```

17.8 Chapter Nine Exercise Solutions

17.8.1 Exercise 9.1. Minimizing Risk Transfer Costs with an Auxiliary Level of Confidence and a VaR Constraint.

With $f_0(\boldsymbol{\theta}) = RTC(\boldsymbol{\theta})$, we have $\partial_\alpha f_0(\boldsymbol{\theta}) = 0$. For this application, $m = 1$ and so, from equation (9.3), we have

$$\partial_\alpha f_0[\boldsymbol{\theta}^*(\alpha), \alpha] = 0 + LME_1^*(\alpha)\ f_{con,1,\alpha}[\boldsymbol{\theta}^*(\alpha), \alpha].$$

Because $f_{con,1}[\boldsymbol{\theta}(\alpha), \alpha] = VaR_\alpha[\boldsymbol{\theta}(\alpha)] - VaR_{max}$, we have $f_{con,1,\alpha}[\boldsymbol{\theta}(\alpha), \alpha] = \partial_\alpha f_{con,1}[\boldsymbol{\theta}(\alpha), \alpha]$ $= \partial_\alpha VaR_\alpha[\boldsymbol{\theta}(\alpha)]$. Thus,

$$\begin{aligned} f_{con,1,\alpha}[\boldsymbol{\theta}(\alpha), \alpha] &= \partial_\alpha f_{con,1}[\boldsymbol{\theta}(\alpha), \alpha] = \partial_\alpha VaR_\alpha[\boldsymbol{\theta}(\alpha)] \\ &= \frac{1}{f_{g(X;\theta)}(VaR_\alpha[\boldsymbol{\theta}(\alpha)])} \end{aligned}$$

the required result.

For this last line, as we did in Section 5.3 we can assume some continuity and start with the relationship $\alpha = F_{g(\mathbf{X};\theta)}\left(F^{-1}_{g(\mathbf{X};\theta)}(\alpha)\right)$. Differentiating both sides with respect to α yields $1 = f_{g(\mathbf{X};\theta)}\left(F^{-1}_{g(\mathbf{X};\theta)}(\alpha)\right)\partial_\alpha F^{-1}_{g(\mathbf{X};\theta)}(\alpha)$ where $f(\cdot)$ is the probability density function corresponding

to the distribution function $F(\cdot)$. Thus, with the notation $VaR_\alpha[g(\mathbf{X};\theta)] = F^{-1}_{g(\mathbf{X};\theta)}(\alpha)$ and equation (9.9), we have

$$\partial_\alpha \; VaR_\alpha[g[\mathbf{X};\boldsymbol{\theta}^*(\alpha)] = \frac{1}{f_{g(\mathbf{X};\theta)}(VaR_\alpha[g[\mathbf{X};\boldsymbol{\theta}^*(\alpha)])}.$$

17.8.2 Exercise 9.2. Minimizing Risk Transfer Costs with an Auxiliary Level of Confidence and an ES Constraint.

From equation (2.3), we have

$$\begin{aligned}
\partial_\alpha ES_\alpha(X) &= \partial_\alpha \left(\tfrac{1}{1-\alpha} \int_\alpha^1 VaR_z(X)dz \right) \\
&= \left(\partial_\alpha \tfrac{1}{1-\alpha}\right) \int_\alpha^1 VaR_z(X)dz + \tfrac{1}{1-\alpha}\partial_\alpha \left(\int_\alpha^1 VaR_z(X)dz \right) \\
&= \left(\tfrac{1}{(1-\alpha)^2} \right) \int_\alpha^1 VaR_z(X)dz + \tfrac{1}{1-\alpha} \left(-VaR_\alpha(X)\right) \\
&= \tfrac{1}{1-\alpha} \left\{ ES_\alpha(X) - VaR_\alpha(X) \right\}.
\end{aligned}$$

Then, from equation (9.3), we have

$$\partial_\alpha RTC[\boldsymbol{\theta}^*(\alpha)] = \frac{LME_1^*(\alpha)}{1-\alpha} \left\{ ES_\alpha(\boldsymbol{\theta}^*(\alpha)) - VaR_\alpha(\boldsymbol{\theta}^*(\alpha)) \right\},$$

the required result.

17.8.3 Exercise 9.3. Minimizing Risk Transfer Costs with an Auxiliary β and a $RVaR$ Constraint

From equation (2.4), if $\beta > 0$, we have

$$\begin{aligned}
\partial_\beta RVaR_{(\alpha,\beta)}(X) &= \partial_\beta \left(\tfrac{1}{\beta} \int_\alpha^{\alpha+\beta} VaR_z(X)dz \right) \\
&= \tfrac{-1}{\beta^2} \int_\alpha^{\alpha+\beta} VaR_z(X)dz + \tfrac{1}{\beta} VaR_{\alpha+\beta}(X) \\
&= \tfrac{1}{\beta} \left\{ VaR_{\alpha+\beta}(X) - RVaR_{(\alpha,\beta)}(X) \right\}.
\end{aligned}$$

Then, from equation (9.3), we have

$$\partial_\beta RTC[\boldsymbol{\theta}^*(\beta)] = \frac{LME_1^*(\beta)}{\beta} \left\{ VaR_{\alpha+\beta}[\boldsymbol{\theta}^*(\alpha)] - RVaR_{(\alpha,\beta)}[\boldsymbol{\theta}^*(\alpha)] \right\},$$

the required result.

17.8.4 Exercise 9.4. Solution

From Display (7.3) and the Section 9.2 notation, we have

$$\begin{aligned}
f_0(z_0, \boldsymbol{\theta}, a) &= z_0 \\
f_{con,1}(z_0, \boldsymbol{\theta}, a) &= -[F_{g(z_0;\boldsymbol{\theta})}(z_0) - \alpha] \\
f_{con,2}(z_0, \boldsymbol{\theta}, a) &= RTC(\boldsymbol{\theta}) - RTC_{max}.
\end{aligned}$$

Taking partial derivatives yields

- $f_{0a}(z_0, \boldsymbol{\theta}, a) = \partial_a f_0(z_0, \boldsymbol{\theta}, a) = 0$,
- $f_{con,1,a}(z_0, \boldsymbol{\theta}, a) = \partial_a f_{con,1}(\boldsymbol{\theta}, a) = -\partial_a[F_{g(X;\boldsymbol{\theta})}(z_0) - \alpha]$,
- $f_{con,2,a}(z_0, \boldsymbol{\theta}, a) = \partial_a f_{con,2}(\boldsymbol{\theta}, a) = \partial_a[RTC(\boldsymbol{\theta}) - RTC_{max}]$,
- $f_{con,1i}(z_0, \boldsymbol{\theta}, a) = \partial_{\theta_i} f_{con,1}(\boldsymbol{\theta}, a) = -\partial_{\theta_i} F_{g(X;\boldsymbol{\theta})}(z_0)$, and
- $f_{con,2i}(z_0, \boldsymbol{\theta}, a) = \partial_{\theta_i} f_{con,2}(\boldsymbol{\theta}, a) = \partial_{\theta_i} RTC(\boldsymbol{\theta})$.

Thus, from equation (9.3) of the Envelope theorem, we have the result in equation (9.15) where

$$\begin{array}{lll} f_{con,1,a}(z_0^*, \boldsymbol{\theta}^*, a) & = -\partial_a([_{g(X;\boldsymbol{\theta})}(z_0) - \alpha]\Big|_{z_0 = VaR^*(a)} & = \sum_{i=1}^p \partial_{\theta_i} F_{g(X;\boldsymbol{\theta})}(z_0)\Big|_{z_0 = VaR^*(a)} \times \partial_a \theta_i^*(a) \\ f_{con,2,a}(z_0^*, \boldsymbol{\theta}^*, a) & = \partial_a[RTC(\boldsymbol{\theta}) - RTC_{max}] & = -\sum_{i=1}^p \partial_{\theta_i} RTC(\boldsymbol{\theta})|_{\boldsymbol{\theta} = \boldsymbol{\theta}^*} \times \partial_a \theta_i^*(a). \end{array}$$

17.8.5 Exercise 9.5. Interpreting Multipliers

For a generic auxiliary variable, note that

$$\begin{array}{ll} f_{con,1,a}(z_0^*, \boldsymbol{\theta}^*, a) & = -\partial_a[F_{g(X;\boldsymbol{\theta})}(z_0) - \alpha]\big|_{z_0 = VaR^*(a)} \\ f_{con,2,a}(z_0^*, \boldsymbol{\theta}^*, a) & = \partial_a[RTC(\boldsymbol{\theta}) - RTC_{max}]. \end{array}$$

a. With $a = \alpha$, we see that $f_{con,1\alpha}[z_0^*, \boldsymbol{\theta}^*(\alpha), \alpha] = 1$ and $f_{con,2\alpha}[z_0^*, \boldsymbol{\theta}^*(\alpha), \alpha] = 0$. Thus, from equation (9.15), we have $\partial_\alpha VaR^*(\alpha) = LME_1^*$. This means that we can interpret the first Lagrange multiplier to be the marginal change in the optimal value at risk per unit change in the level of confidence, α.

b. Similarly, suppose that the auxiliary variable is the maximal risk transfer cost, that is, $a = RTC_{max}$. Then, $f_{con,1\alpha}[z_0^*, \boldsymbol{\theta}^*(RTC_{max}), RTC_{max}] = 0$ and $f_{con,2RTC_{max}}[z_0^*, \boldsymbol{\theta}^*(RTC_{max}), RTC_{max}] = -1$. Thus, we have $\partial_{RTC_{max}} VaR^*(RTC_{max}) = -LME_2^*$. This means that we can interpret the second Lagrange multiplier to be minus one times the marginal change in the optimal value at risk per unit change in the maximal risk transfer cost, RTC_{max}.

17.8.6 Exercise 9.6. Compare Changes in the ES and $GlueVaR$

Starting with equation (2.3), we have

$$\begin{array}{ll} \partial_\alpha ES_\alpha(X) & = \partial_\alpha \left(\frac{1}{1-\alpha} \int_\alpha^1 VaR_z(X) dz \right) \\ & = \left(\partial_\alpha \frac{1}{1-\alpha} \right) \int_\alpha^1 VaR_z(X) dz + \frac{1}{1-\alpha} \partial_\alpha \left(\int_\alpha^1 VaR_z(X) dz \right) \\ & = \left(\frac{1}{(1-\alpha)^2} \right) \int_\alpha^1 VaR_z(X) dz + \frac{1}{1-\alpha} \left(-VaR_\alpha(X) \right) \\ & = \frac{1}{1-\alpha} \left\{ ES_\alpha(X) - VaR_\alpha(X) \right\}. \end{array}$$

This is sufficient for the result.

17.8.7 Exercise 9.7. $GlueVaR$ Risk Retention

Let $a = \omega$ be the auxiliary parameter.

As in Section 9.3 examples, ω does not affect any of the equality or binding inequality constraints and so equation (9.9) holds. Thus,

$$\partial_\omega \; GlueVaR_\omega[g(\mathbf{X};\boldsymbol{\theta}^*)] \quad = ES_\alpha[g(\mathbf{X};\boldsymbol{\theta}^*)] - VaR_\alpha[g(\mathbf{X};\boldsymbol{\theta}^*)].$$

17.8.8 Exercise 9.8. $GlueVaR$ and the ANU Case

```
#R Code for Changing Weight (w) in GlueVaR

alpha = 0.90
RMuw.fct <- function(u,w){
  RTC.R   <- 0
  sumYj <- rep(0,nsim)
  for (j in 1:numVars){
    uj = as.numeric(u[j])
    Yj    <- pmin(uj, Claims[,j])
    sumYj <- sumYj + pmin(uj, Claims[,j])
    RTC.R   <- RTC.R + mean(Claims[,j])- mean(Yj)
    }
  VaR     <- quantile(sumYj, prob = alpha)
  CTE     <- VaR + ( mean(pmax(sumYj- VaR, 0)) )/(1-alpha)
  StdDev <- sd(sumYj)
  GlueVaR <- (1-w)*VaR + w*CTE
  c(RTC.R, VaR, CTE, StdDev, GlueVaR)
}

time1 <- Sys.time()
Weightvec <- seq(from=0, to=1, length.out=11)

OutMat.GVaR  <- matrix(0,nrow = length(Weightvec), ncol = 7+numVars)

u.names <- paste("$u_{",1:numVars, "}$", sep="")
colnames(OutMat.GVaR) <- c("Weight", "$RTC$", "$VaR$", "$ES$",
                           "Std Dev", "$GlueVaR$", "$LM$", u.names)

starter.GVaR <- starter.u

for (jCost in 1:length(Weightvec)){
  Weight = Weightvec[jCost]
  RTC.GVaR <-  function(u){RMuw.fct(u,w=Weight)[1]}
  f0.GVaR  <-  function(u){RMuw.fct(u,w=Weight)[5]}
  hin.GVaR <- function(u) {h <- NA
    h[1] <- RTCmaxFix - RTC.GVaR(u)
    for (j in 1:numVars){h[j+1] <- u[j] }
    return(h)}
  #if (jCost > 1){starter.GVaR <- GVaR.opt$par}
    GVaR.opt <- alabama::auglag(par=starter.GVaR,
           fn=f0.GVaR,
           hin=hin.GVaR,
           control.outer=list(method="nlminb",trace=FALSE))
  uparams <- GVaR.opt$par
  output <- c(Weight, RMuw.fct(uparams,Weight), GVaR.opt$lambda[1], uparams)
```

```
  #cat("alabama results",output, "\n")
  OutMat.GVaR[jCost,] <- output
  }
write.csv(OutMat.GVaR, file="Data/Chap9Tables/OutMatGVaR.csv")
Sys.time() - time1
```

```
OutMat.GVaR<- read.csv("Data/Chap9Tables/OutMatGVaR.csv",header=T)
numVars = 13
OutMat.GVaR <- OutMat.GVaR[,-1]

numVarsPrint <- min(3,numVars)
uPrint.names <- paste("$u_{",1:numVarsPrint, "}$", sep="")

OutMatPrint <- OutMat.GVaR[,1:(7+numVarsPrint)]
colnames(OutMatPrint) <- c("Weight", "$RTC$", "$VaR$", "$ES$",
                           "Std Dev", "$GlueVaR$", "$LME$", uPrint.names)
OutMatPrint[,1] <- round(OutMatPrint[,1], digits = 2)
OutMatPrint[,2:6] <- round(OutMatPrint[,2:6], digits = 0)
OutMatPrint[,7] <- round(OutMatPrint[,7], digits = 2)
OutMatPrint[,8:(7+numVarsPrint)] <-
      round(OutMatPrint[,8:(7+numVarsPrint)], digits = 0)

TableGen1(TableData=OutMatPrint,
  TextTitle=
   'ANU Excess of Loss Optimization - GlueVar Criterion - Varying Weight',
         Align='r', Digits=2, ColumnSpec=1:9,
         BorderRight= 1, ColWidth=ColWidth9 )
```

17.8.9 Exercise 9.9. $GlueVaR$ and the ANU Case - Upper

```
load(file = "ChapTablesData/Chap7/PremTable.Rdata")
riskTypesShortSelect  <- riskTypesShort[2:15]

#  Define a Function to Plot for Each Allocation
rndfac <- 20
bdd <- 2
pltfunc <- function(num,varx,vary,labelx){
     limy1 <- max(rndfac*(round(min(vary)/rndfac)-bdd),0)
     limy2 <- rndfac*(round(max(vary)/rndfac)+bdd)
     limx1 <- min(varx)
     limx2 <- max(varx)
     ggplot(dfPlot, aes(x=varx, y=vary)) + geom_point() + theme_bw() +geom_line() +
          scale_y_continuous(breaks = seq(from = limy1, to =   limy2, length.out = 3),
                             limits =  c(limy1, limy2)) +
          labs(y = riskTypesShortSelect[num]) +
          theme(axis.title.y = element_text(size = 6)) + labs(x = labelx) +
          scale_x_continuous(breaks = seq(from = limx1, to =   limx2, length.out = 3),
                             limits =  c(limx1, limx2))
```

```
      }

OutMat.GVaR<- read.csv("Data/Chap9Tables/OutMatGVaR.csv",header=T)

dfPlot <- data.frame(OutMat.GVaR)

q <- list(rep(0,numVars))
for (iPlot in 1:numVars){
  q[[iPlot]] <- pltfunc(iPlot,varx=dfPlot[["Weight"]],
                        vary=dfPlot[[paste("u",iPlot, sep="")]],labelx="Weight")
}
if (numVars == 2) {grid.arrange(q[[1]],q[[2]], nrow=1)}
if (numVars == 3) {grid.arrange(q[[1]],q[[2]],q[[3]], nrow=1)}
if (numVars == 13) {grid.arrange(q[[1]],  q[[2]],  q[[3]],  q[[4]], q[[5]],
                                 q[[6]],  q[[7]],  q[[8]],  q[[9]], q[[10]],
                                 q[[11]], q[[12]], q[[13]],         nrow=5)}
if (numVars == 14) {grid.arrange(q[[1]],  q[[2]],  q[[3]],  q[[4]], q[[5]],
                                 q[[6]],  q[[7]],  q[[8]],  q[[9]], q[[10]],
                                 q[[11]], q[[12]], q[[13]], q[[14]], nrow=5)}
```

17.8.10 Exercise 9.10. Quota Share

To determine the optimal choice variables $\mathbf{c}$, we take partial derivatives

$$\begin{aligned} \partial_{\mathbf{c}} LA &= \partial_{\mathbf{c}} \left\{ \mathbf{c}' \boldsymbol{\Sigma} \mathbf{c} + LME_1 [\boldsymbol{\mu}'(\mathbf{1} - \mathbf{c}) - RTC_{max}] \right\} \\ &= 2\boldsymbol{\Sigma}\mathbf{c} + LME_1 \ \boldsymbol{\mu} \\ \Longrightarrow \quad \mathbf{c}^* &= \tfrac{LME_1^*}{2} \ \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}. \end{aligned}$$

With the binding constraint $f_{con}(\mathbf{c}) = \boldsymbol{\mu}'(\mathbf{1} - \mathbf{c}) - RTC_{max}$, we have

$$\begin{aligned} RTC_{max} &= \boldsymbol{\mu}'(\mathbf{1} - \mathbf{c}^*) \\ &= \boldsymbol{\mu}'(\mathbf{1} - \tfrac{LME_1^*}{2} \ \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}) \\ \Longrightarrow \quad LME_1^* &= 2(\boldsymbol{\mu}'\mathbf{1} - RTC_{max}) / (\boldsymbol{\mu}' \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}). \end{aligned}$$

Thus,

$$\begin{aligned} \tfrac{1}{2} \partial_\mu \ LME_1^* &= \tfrac{1}{(\boldsymbol{\mu}' \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu})^2} \left\{ (\boldsymbol{\mu}' \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu})(1) - (\boldsymbol{\mu}'\mathbf{1} - RTC_{max}) 2 (\partial_\mu \boldsymbol{\mu})' \ \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} \right\} \\ &= \tfrac{1}{\boldsymbol{\mu}' \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}} \left\{ 1 - LME_1^* (\partial_\mu \boldsymbol{\mu})' \ \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} \right\}, \end{aligned}$$

which is sufficient for equation (9.17).

In the same way, taking a partial derivative with respect to μ of the optimal decision variables yields

$$\begin{aligned} \partial_\mu \mathbf{c}^* &= \partial_\mu \left\{ \tfrac{LME_1^*(\mu)}{2} \ \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} \right\} \\ &= \tfrac{1}{2} \left\{ [\partial_\mu LME_1^*(\mu)] \ \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + LME_1^*(\mu) \ \boldsymbol{\Sigma}^{-1} [\partial_\mu \boldsymbol{\mu}] \right\}, \end{aligned}$$

which is sufficient for equation (9.16).

17.8.11 Exercise 9.11. Quota Share

Starting with the optimal value of the Lagrange multiplier, we have

$$\begin{aligned}\partial_\sigma LME_1^* &= 2\partial_\sigma \tfrac{(\boldsymbol{\mu}'\mathbf{1}-RTC_{max})}{(\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu})} &&= -2\tfrac{(\boldsymbol{\mu}'\mathbf{1}-RTC_{max})}{(\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu})^2}\partial_\sigma(\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}) \\ &= -\tfrac{LME_1^*}{(\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu})}(\boldsymbol{\mu}'[\partial_\sigma\boldsymbol{\Sigma}^{-1}]\boldsymbol{\mu}) &&= \tfrac{LME_1^*}{(\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu})}(\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}[\partial_\sigma\boldsymbol{\Sigma}]\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}) \\ &= \tfrac{2}{(\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu})}(\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}[\partial_\sigma\boldsymbol{\Sigma}]\mathbf{c}^*),\end{aligned}$$

which is sufficient for equation (9.19).

In the same way, taking a partial derivative with respect to σ of the optimal decision variables yields

$$\begin{aligned}2\ \partial_\sigma\mathbf{c}^* &= \partial_\sigma\left\{LME_1^*(\sigma)\ \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right\} \\ &= \{\partial_\sigma LME_1^*(\sigma)\}\ \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + LME_1^*(\sigma)\ \left\{\partial_\sigma\boldsymbol{\Sigma}^{-1}\right\}\boldsymbol{\mu} \\ &= \{\partial_\sigma LME_1^*(\sigma)\}\ \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - LME_1^*(\sigma)\ \boldsymbol{\Sigma}^{-1}\{\partial_\sigma\boldsymbol{\Sigma}\}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ &= \{\partial_\sigma LME_1^*(\sigma)\}\ \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - 2\ \boldsymbol{\Sigma}^{-1}\{\partial_\sigma\boldsymbol{\Sigma}\}\mathbf{c}^*,\end{aligned}$$

which is sufficient for equation (9.18).

17.8.12 Exercise 9.12. Asset Allocation

Recall from Section 9.4.1 that we have

$$\begin{aligned}LME_1^* &= [\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\mathbf{1} - 2\lambda_{Port}]/\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1} \\ \mathbf{c}^* &= \tfrac{1}{2\lambda_{Port}}\ \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - LME_1^*\ \mathbf{1}).\end{aligned}$$

Starting with the optimal value of the Lagrange multiplier, we have

$$\begin{aligned}\partial_\sigma LME_1^* &= \partial_\sigma \tfrac{(\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\mathbf{1}-2\lambda_{Port})}{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1}} \\ &= \tfrac{1}{(\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1})^2}\left[(\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1})\{\partial_\sigma(\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\mathbf{1} - 2\lambda_{Port})\} - (\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\mathbf{1} - 2\lambda_{Port})\{\partial_\sigma(\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1})\}\right] \\ &= \tfrac{1}{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1}}\left[\{\partial_\sigma(\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\mathbf{1})\} - LME_1^*\{\partial_\sigma(\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1})\}\right] \\ &= \tfrac{1}{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1}}\left[\{-\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\{\partial_\sigma\boldsymbol{\Sigma}\}\boldsymbol{\Sigma}^{-1}\mathbf{1}\} + LME_1^*\{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\{\partial_\sigma\boldsymbol{\Sigma}\}\boldsymbol{\Sigma}^{-1}\mathbf{1}\}\right] \\ &= \tfrac{1}{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1}}\left[LME_1^*\ \mathbf{1} - \boldsymbol{\mu}\right]'\boldsymbol{\Sigma}^{-1}\{\partial_\sigma\boldsymbol{\Sigma}\}\boldsymbol{\Sigma}^{-1}\mathbf{1} \\ &= \tfrac{1}{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1}}\left[\mathbf{c}^*(2\lambda_{Port})\right]'\{\partial_\sigma\boldsymbol{\Sigma}\}\boldsymbol{\Sigma}^{-1}\mathbf{1},\end{aligned}$$

which is sufficient for equation (9.21).

In the same way, taking a partial derivative with respect to σ of the optimal decision variables yields

$$\begin{aligned}2\lambda_{Port}\ \partial_\sigma\mathbf{c}^* &= \partial_\sigma\left[\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - LME_1^*(\sigma)\ \mathbf{1})\right] \\ &= [\partial_\sigma\boldsymbol{\Sigma}^{-1}](\boldsymbol{\mu} - LME_1^*(\sigma)\ \mathbf{1}) - \boldsymbol{\Sigma}^{-1}[\partial_\sigma LME_1^*(\sigma)]\ \mathbf{1} \\ &= -[\boldsymbol{\Sigma}^{-1}\{\partial_\sigma\boldsymbol{\Sigma}\}\boldsymbol{\Sigma}^{-1}](\boldsymbol{\mu} - LME_1^*(\sigma)\ \mathbf{1}) - \boldsymbol{\Sigma}^{-1}[\partial_\sigma LME_1^*(\sigma)]\ \mathbf{1} \\ &= -\boldsymbol{\Sigma}^{-1}\{\partial_\sigma\boldsymbol{\Sigma}\}\mathbf{c}^*(2\lambda_{Port}) - \boldsymbol{\Sigma}^{-1}[\partial_\sigma LME_1^*(\sigma)]\ \mathbf{1},\end{aligned}$$

which is sufficient for equation (9.20).

```
LPort <- 10

MeanPort.sigmamat <- 0* cor(Return_trn)
VariancePort.sigmamat <- 0* cor(Return_trn)
for (iauxrow in 1:8){
  for (iauxcol in 1:8){ if (iauxrow <= iauxcol) {
  partialSig <-  matrix(0, nrow=8, ncol=8)
  partialSig[iauxrow,iauxcol] <- 1 -> partialSig[iauxcol, iauxrow]
  LMEsig <- as.numeric(2*LPort * one.SigInv %*% partialSig %*% cstar / sum.SigInv)
  c.sigAdd <- 1/(2*LPort) * SigmaInv %*% oneone.vec * LMEsig -
                  SigmaInv %*% partialSig %*% cstar
  MeanPort.sigmamat[iauxrow, iauxcol] <- t(Means) %*% c.sigAdd -> MeanPort.sigmamat[iauxcol, iauxrow]
  Temp <- t(c.sigAdd) %*% Sigma %*% cstar +
        t(cstar) %*% partialSig %*% cstar +
        t(cstar) %*% Sigma %*% c.sigAdd
  VariancePort.sigmamat[iauxrow, iauxcol] <- Temp -> VariancePort.sigmamat[iauxcol, iauxrow]
  }}}
StdDevPort.sigmamat <- as.numeric(0.5 * PortfolioStdDev**(-1)) * VariancePort.sigmamat
```

17.9 Chapter 10 Exercise Solutions

17.9.1 Exercise 10.1. Quota Share

The decision variables are $\mathbf{z} = \mathbf{c}$, there is a single binding constraint $f_{con,1}(\mathbf{c}) = \boldsymbol{\mu}'(\mathbf{1} - \mathbf{c}) - RTC_{max}$, and the (shortened) Lagrangian is $LA = \mathbf{c}'\boldsymbol{\Sigma}\mathbf{c} + LMI_1[\boldsymbol{\mu}'(\mathbf{1} - \mathbf{c}) - RTC_{max}]$.

Thus, the hessian of the shortened Lagrangian

$$\begin{aligned} SLA_{\mathbf{cc}'} = \partial_{\mathbf{c}}\partial_{\mathbf{c}'} LA &= \partial_{\mathbf{c}}\partial_{\mathbf{c}'} \{\mathbf{c}'\boldsymbol{\Sigma}\mathbf{c} + LMI_1[\boldsymbol{\mu}'(\mathbf{1} - \mathbf{c}) - RTC_{max}]\} \\ &= \partial_{\mathbf{c}} \{2\boldsymbol{\Sigma}\mathbf{c} - LMI_1\ \boldsymbol{\mu}\} \\ &= 2\boldsymbol{\Sigma}. \end{aligned}$$

For auxiliary variable $a = \mu$, we have

$$\begin{aligned} SLA_{\mathbf{c}\mu} &= \partial_{\mathbf{c}}\partial_{\mu} \{\mathbf{c}'\boldsymbol{\Sigma}\mathbf{c} + LMI_1[\boldsymbol{\mu}'(\mathbf{1} - \mathbf{c}) - RTC_{max}]\} \\ &= \partial_{\mu} \{2\boldsymbol{\Sigma}\mathbf{c} - LMI_1\ \boldsymbol{\mu}\} \\ &= -LMI_1\ \partial_{\mu}\boldsymbol{\mu}. \end{aligned}$$

For the constraint function, we have

$$\begin{aligned} f_{con,1,\mathbf{c}}(\mathbf{c}) = \partial_{\mathbf{c}} f_{con,1}(\mathbf{c}) &= \partial_{\mathbf{c}} [\boldsymbol{\mu}'(\mathbf{1} - \mathbf{c}) - RTC_{max}] \\ &= -\boldsymbol{\mu} \end{aligned}$$

and

$$\begin{aligned} f_{con,1,\mu}(\mathbf{c}) = \partial_{\mu} f_{con,1}(\mathbf{c}) &= \partial_{\mu} [\boldsymbol{\mu}'(\mathbf{1} - \mathbf{c}) - RTC_{max}] \\ &= 1 - c^*, \end{aligned}$$

where c^* corresponds to the choice of μ.

From the Perturbation Sensitivity Proposition, we have

$$\begin{aligned}\partial_\mu LMI_1^* &= \left\{f^*_{con,1,\mathbf{c}'} SLA^{*-1}_{\mathbf{cc}'}\ f^*_{con,1,\mathbf{c}}\right\}^{-1}\left\{f^*_{con,1,\mu} - f^*_{con,1,\mathbf{c}'}\ SLA^{*-1}_{\mathbf{cc}'}\ SLA^*_{\mathbf{c}\mu}\right\}\\ &= \left\{\boldsymbol{\mu}'(2\boldsymbol{\Sigma})^{-1}\ \boldsymbol{\mu}\right\}^{-1}\left\{(1-c^*) - \boldsymbol{\mu}'\ (2\boldsymbol{\Sigma})^{-1}\ LMI_1^*\ \partial_\mu\boldsymbol{\mu}\right\}\\ &= \frac{2}{\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\ \boldsymbol{\mu}}\left\{1 - c^* - \frac{LMI_1^*}{2}\boldsymbol{\mu}'\ \boldsymbol{\Sigma}^{-1}\ \ \partial_\mu\boldsymbol{\mu}\right\}.\end{aligned}$$

For the optimal decision variables, we have

$$\begin{aligned}&\mathbf{c}^* = \frac{LMI_1^*}{2}\ \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\\ \Longrightarrow\quad &c^* = \frac{LMI_1^*}{2}\ \boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\partial_\mu\boldsymbol{\mu},\end{aligned}$$

which is sufficient for equation (9.17).

From the Perturbation Sensitivity Proposition, we have

$$\begin{aligned}\partial_\mu\ \mathbf{c}^* &= -SLA^{*-1}_{\mathbf{cc}'}\left\{SLA^*_{\mathbf{z}\mu} + f^*_{con,1,\mathbf{c}}\ \partial_\mu LMI_1^*\right\}\\ &= -(2\boldsymbol{\Sigma})^{-1}\left\{-LMI_1^*\ \partial_\mu\boldsymbol{\mu} - \boldsymbol{\mu}\ \partial_\mu LMI_1^*\right\}\\ &= \tfrac{1}{2}\boldsymbol{\Sigma}^{-1}\left\{\boldsymbol{\mu}\ \partial_\mu LMI_1^* + LMI_1^*\ \partial_\mu\boldsymbol{\mu}\right\},\end{aligned}$$

which is sufficient for equation (9.16).

17.9.2 Exercise 10.2. Quota Share

As in Exercise 10.4.1, the decision variables are $\mathbf{z} = \mathbf{c}$, there is a single binding constraint $f_{con,1}(\mathbf{c}) = \boldsymbol{\mu}'(\mathbf{1}-\mathbf{c}) - RTC_{max}$, and the (shortened) Lagrangian is $LA = \mathbf{c}'\boldsymbol{\Sigma}\mathbf{c} + LMI_1[\boldsymbol{\mu}'(\mathbf{1}-\mathbf{c}) - RTC_{max}]$. Further recall that the hessian of the shortened Lagrangian is $SLA_{\mathbf{cc}'} = 2\boldsymbol{\Sigma}$.

For auxiliary variable $a = \sigma$, we have

$$\begin{aligned}SLA_{\mathbf{c}\sigma} &= \partial_{\mathbf{c}}\partial_\sigma\left\{\mathbf{c}'\boldsymbol{\Sigma}\mathbf{c} + LMI_1[\boldsymbol{\mu}'(\mathbf{1}-\mathbf{c}) - RTC_{max}]\right\}\\ &= \partial_\sigma\left\{2\boldsymbol{\Sigma}\mathbf{c} - LMI_1\ \boldsymbol{\mu}\right\}\\ &= 2\left\{\partial_\sigma\boldsymbol{\Sigma}\right\}\mathbf{c}.\end{aligned}$$

For the constraint function, recall

$$f_{con,1,\mathbf{c}}(\mathbf{c}) = \partial_{\mathbf{c}} f_{con,1}(\mathbf{c}) = -\boldsymbol{\mu}$$

Further,

$$\begin{aligned}f_{con,1,\sigma}(\mathbf{c}) = \partial_\sigma f_{con,1}(\mathbf{c}) &= \partial_\sigma\left[\boldsymbol{\mu}'(\mathbf{1}-\mathbf{c}) - RTC_{max}\right]\\ &= 0.\end{aligned}$$

From the Perturbation Sensitivity Proposition, we have

$$\begin{aligned}\partial_\sigma LMI_1^* &= \left\{f^*_{con,1,\mathbf{c}'} SLA^{*-1}_{\mathbf{cc}'}\ f^*_{con,1,\mathbf{c}}\right\}^{-1}\left\{f^*_{con,1,\sigma} - f^*_{con,1,\mathbf{c}'}\ SLA^{*-1}_{\mathbf{cc}'}\ SLA^*_{\mathbf{c}\sigma}\right\}\\ &= \frac{2}{\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\ \boldsymbol{\mu}}\left\{0 + \boldsymbol{\mu}'\ (2\boldsymbol{\Sigma})^{-1}\ 2\left\{\partial_\sigma \boldsymbol{\Sigma}\right\}\mathbf{c}^*\right\}\\ &= \frac{2}{\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\ \boldsymbol{\mu}}\left\{\boldsymbol{\mu}'\ \boldsymbol{\Sigma}^{-1}\ \left\{\partial_\sigma \boldsymbol{\Sigma}\right\}\mathbf{c}^*\right\},\end{aligned}$$

which is sufficient for equation (9.19).

From the Perturbation Sensitivity Proposition, we have

$$\begin{aligned}\partial_\sigma\ \mathbf{c}^* &= -SLA^{*-1}_{\mathbf{cc}'}\left\{SLA^*_{\mathbf{z}\sigma} + f^*_{con,1,\mathbf{c}}\ \partial_\sigma LMI_1^*\right\}\\ &= -(2\boldsymbol{\Sigma})^{-1}\left\{2\left\{\partial_\sigma \boldsymbol{\Sigma}\right\}\mathbf{c}^* - \boldsymbol{\mu}\ \partial_\mu LMI_1^*\right\}\\ &= \boldsymbol{\Sigma}^{-1}\left\{\tfrac{1}{2}\boldsymbol{\mu}\ \partial_\sigma LMI_1^* - \partial_\sigma \boldsymbol{\Sigma}\mathbf{c}^*\right\},\end{aligned}$$

which is sufficient for equation (9.18).

17.9.3 Exercise 10.3. Asset Allocation

The decision variables are $\mathbf{z} = \mathbf{c}$, there is a single binding constraint $f_{con,1}(\mathbf{c}) = \mathbf{c}'\mathbf{1} - 1$, and the (shortened) Lagrangian is $LA = -\left(\mathbf{c}'\boldsymbol{\mu} - \lambda_{Port}\ \mathbf{c}'\boldsymbol{\Sigma}\mathbf{c}\right) + LMI_1[\mathbf{c}'\mathbf{1} - 1]\$$.

Thus, the hessian of the shortened Lagrangian

$$\begin{aligned}SLA_{\mathbf{cc}'} = \partial_{\mathbf{c}}\partial_{\mathbf{c}'} LA &= \partial_{\mathbf{c}}\partial_{\mathbf{c}'}\left\{-\left(\mathbf{c}'\boldsymbol{\mu} - \lambda_{Port}\ \mathbf{c}'\boldsymbol{\Sigma}\mathbf{c}\right) + LMI_1[\mathbf{c}'\mathbf{1} - 1\right\}\\ &= \partial_{\mathbf{c}}\left\{-\boldsymbol{\mu} + 2\lambda_{Port}\boldsymbol{\Sigma}\mathbf{c} + LMI_1\ \mathbf{1}\right\}\\ &= 2\lambda_{Port}\boldsymbol{\Sigma}.\end{aligned}$$

For auxiliary variable $a = \mu$, we have

$$\begin{aligned}SLA_{\mathbf{c}\mu} &= \partial_{\mathbf{c}}\partial_{\mu}\left\{-\left(\mathbf{c}'\boldsymbol{\mu} - \lambda_{Port}\ \mathbf{c}'\boldsymbol{\Sigma}\mathbf{c}\right) + LMI_1[\mathbf{c}'\mathbf{1} - 1\right\}\\ &= \partial_{\mu}\left\{-\boldsymbol{\mu} + 2\lambda_{Port}\boldsymbol{\Sigma}\mathbf{c} + LMI_1\ \mathbf{1}\right\}\\ &= -\ \partial_\mu\boldsymbol{\mu}.\end{aligned}$$

For the constraint function, we have

$$\begin{aligned}f_{con,1,\mathbf{c}}(\mathbf{c}) = \partial_{\mathbf{c}} f_{con,1}(\mathbf{c}) &= \partial_{\mathbf{c}}\left[\mathbf{c}'\mathbf{1} - 1\right]\\ &= \mathbf{1}\end{aligned}$$

and

$$\begin{aligned}f_{con,1,\mu}(\mathbf{c}) = \partial_{\mu} f_{con,1}(\mathbf{c}) &= \partial_{\mu}\left[\mathbf{c}'\mathbf{1} - 1\right]\\ &= 0.\end{aligned}$$

From the Perturbation Sensitivity Proposition, we have

$$\begin{aligned}\partial_\mu LMI_1^* &= \left\{f^*_{con,1,\mathbf{c}'} SLA^{*-1}_{\mathbf{cc}'}\ f^*_{con,1,\mathbf{c}}\right\}^{-1}\left\{f^*_{con,1,\mu} - f^*_{con,1,\mathbf{c}'}\ SLA^{*-1}_{\mathbf{cc}'}\ SLA^*_{\mathbf{c}\mu}\right\}\\ &= \left\{\mathbf{1}'(2\lambda_{Port}\boldsymbol{\Sigma})^{-1}\ \mathbf{1}\right\}^{-1}\left\{0 + \mathbf{1}'\ (2\lambda_{Port}\boldsymbol{\Sigma})^{-1}\ \partial_\mu\boldsymbol{\mu}\right\}\\ &= \frac{1}{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\ \mathbf{1}}\left\{\mathbf{1}'\ \boldsymbol{\Sigma}^{-1}\ \partial_\mu\boldsymbol{\mu}\right\},\end{aligned}$$

as in equation (9.12).

From the Perturbation Sensitivity Proposition, we have

$$\begin{aligned}\partial_\mu\ \mathbf{c}^* &= -SLA^{*-1}_{\mathbf{cc'}}\left\{SLA^*_{\mathbf{c}\mu} + f^*_{con,1,\mathbf{c}}\ \partial_\mu LMI^*_1\right\}\\ &= -(2\lambda_{Port}\mathbf{\Sigma})^{-1}\left\{-\ \partial_\mu\boldsymbol{\mu} + \mathbf{1}\ \partial_\mu LMI^*_1\right\}\\ &= \tfrac{1}{2\lambda_{Port}}\mathbf{\Sigma}^{-1}\left\{\partial_\mu\boldsymbol{\mu} - \mathbf{1}\ \partial_\mu LMI^*_1\right\},\end{aligned}$$

which is sufficient for equation (9.11).

17.9.4 Exercise 10.4. Asset Allocation

The decision variables are $\mathbf{z} = \mathbf{c}$, there is a single binding constraint $f_{con,1}(\mathbf{c}) = \mathbf{c'1} - 1$, and the (shortened) Lagrangian is $LA = -\left(\mathbf{c'}\boldsymbol{\mu} - \lambda_{Port}\ \mathbf{c'\Sigma c}\right) + LMI_1[\mathbf{c'1} - 1]\$$.

As in Exercise 10.1.3, the hessian of the shortened Lagrangian is $SLA_{\mathbf{cc'}} = 2\lambda_{Port}\mathbf{\Sigma}$.

For auxiliary variable $a = \sigma$, we have

$$\begin{aligned}SLA_{\mathbf{c}\sigma} &= \partial_{\mathbf{c}}\partial_\sigma\left\{-\left(\mathbf{c'}\boldsymbol{\mu} - \lambda_{Port}\ \mathbf{c'\Sigma c}\right) + LMI_1[\mathbf{c'1} - 1\right\}\\ &= \partial_\sigma\left\{-\boldsymbol{\mu} + 2\lambda_{Port}\mathbf{\Sigma c} + LMI_1\ \mathbf{1}\right\}\\ &= 2\lambda_{Port}[\partial_\sigma\mathbf{\Sigma}]\mathbf{c}.\end{aligned}$$

For the constraint function, we have $f_{con,1,\mathbf{c}}(\mathbf{c}) = \mathbf{1}$, as in Exercise 10.1.3. Further

$$\begin{aligned}f_{con,1,\sigma}(\mathbf{c}) = \partial_\sigma f_{con,1}(\mathbf{c}) &= \partial_\sigma\left[\mathbf{c'1} - 1\right]\\ &= 0.\end{aligned}$$

From the Perturbation Sensitivity Proposition, we have

$$\begin{aligned}\partial_\sigma LMI^*_1 &= \left\{f^*_{con,1,\mathbf{c'}}SLA^{*-1}_{\mathbf{cc'}}\ f^*_{con,1,\mathbf{c}}\right\}^{-1}\left\{f^*_{con,1,\sigma} - f^*_{con,1,\mathbf{c'}}\ SLA^{*-1}_{\mathbf{cc'}}\ SLA^*_{\mathbf{c}\sigma}\right\}\\ &= \left\{\mathbf{1'}(2\lambda_{Port}\mathbf{\Sigma})^{-1}\ \mathbf{1}\right\}^{-1}\left\{0 + \mathbf{1'}\ (2\lambda_{Port}\mathbf{\Sigma})^{-1}\ 2\lambda_{Port}[\partial_\sigma\mathbf{\Sigma}]\mathbf{c}^*\right\}\\ &= \tfrac{2\lambda_{Port}}{\mathbf{1'\Sigma^{-1}1}}\left\{\mathbf{1'}\ \mathbf{\Sigma}^{-1}\ [\partial_\sigma\mathbf{\Sigma}]\mathbf{c}^*\right\},\end{aligned}$$

as in equation (9.21).

From the Perturbation Sensitivity Proposition, we have

$$\begin{aligned}\partial_\sigma\ \mathbf{c}^* &= -SLA^{*-1}_{\mathbf{cc'}}\left\{SLA^*_{\mathbf{c}\sigma} + f^*_{con,1,\mathbf{c}}\ \partial_\sigma LMI^*_1\right\}\\ &= -(2\lambda_{Port}\mathbf{\Sigma})^{-1}\left\{2\lambda_{Port}[\partial_\sigma\mathbf{\Sigma}]\mathbf{c}^* + \mathbf{1}\ \partial_\sigma LMI^*_1\right\}\\ &= -\mathbf{\Sigma}^{-1}[\partial_\sigma\mathbf{\Sigma}]\mathbf{c}^* - \tfrac{1}{2\lambda_{Port}}\mathbf{\Sigma}^{-1}\mathbf{1}\ \partial_\sigma LMI^*_1,\end{aligned}$$

as in equation (9.20).

17.9.5 Exercise 10.5. Excess of Loss VaR Sensitivity, part (a)

```
# First, determine optimal parameters without auxiliary variables
#
#  Set Simulation Parameters
nsim <- 2000000
```

```
set.seed(202020)
risk1shape <- 2
risk1scale <- 2000
risk2shape <- 3
risk2scale <- 2000
#  Independent normal random variables
rGaus    <- matrix(rnorm(2*nsim),nrow=nsim,ncol=2)
sigparam <- 0.5
Sigma.11 <- matrix(c(1,sigparam,sigparam,1),nrow=2,ncol=2)
rGaus.new <- rGaus %*% chol(Sigma.11)
UCop1    <- pnorm(rGaus.new[,1])
UCop2    <- pnorm(rGaus.new[,2])
#  Marginal Distribution Random variables
X1 <-          qgamma(p=UCop1, shape = risk1shape, scale = risk1scale)
X2 <- actuar::qpareto(p=UCop2, shape = risk2shape, scale = risk2scale)
alpha0 <- 0.95
alpha1 <- 0.85
alpha2 <- 0.75

# Objective Function
f0.a <- function(u1,u2,a){
  S <- pmin(X1,u1) +  pmin(X2*(1+a/100),u2)
  stddev  <- sd(S)
  Quan0   <- quantile(S, probs = alpha0, na.rm = TRUE)
  Quan1   <- quantile(S, probs = alpha1, na.rm = TRUE)
  Quan2   <- quantile(S, probs = alpha2, na.rm = TRUE)
  excess0 <- pmax(S-Quan0,0)
  CTE0    <- Quan0 + mean(excess0)/(1-alpha0)
  excess1 <- pmax(S-Quan1,0)
  CTE1    <- Quan1 + mean(excess1)/(1-alpha1)
  excess2 <- pmax(S-Quan2,0)
  CTE2    <- Quan2 + mean(excess2)/(1-alpha2)
  Output  <- c(stddev, Quan0,Quan1, Quan2, CTE0,CTE1, CTE2)
  return(Output)
}

#  Constraint Function
RTC.a <- function(u1,u2,a){
  TotalCost <-
    actuar::mgamma(order=1, shape = risk1shape, scale = risk1scale) +
    actuar::mpareto(order=1, shape = risk2shape, scale = risk2scale*(1+a/100))
  TotalRetained <-
    actuar::levgamma(limit = u1,shape = risk1shape, scale = risk1scale) +
    actuar::levpareto(limit = u2,shape = risk2shape, scale = risk2scale*(1+a/100))
  RTCost<- TotalCost - TotalRetained
  return(RTCost)
}

RTC.Upper <-  actuar::mgamma( order=1, shape = risk1shape, scale = risk1scale) +
              actuar::mpareto(order=1, shape = risk2shape, scale = risk2scale)
RTCmax <- 0.20*RTC.Upper
```

```
#  Start retention parameters at mean values
starter <- c(actuar::mgamma(order=1, shape = risk1shape, scale = risk1scale),
             actuar::mpareto(order=1, shape = risk2shape, scale = risk2scale))

fineq <- function(par) {
  h <- NA
  h[1] <- {-(RTC.a(u1=par[1],u2=par[2],a=0)-RTCmax)}
  h[2] <- par[1]
  h[3] <- par[2]
  return(h)}

RMAlternatives <- 1:7
OptOutMat <- matrix(0,10,length(RMAlternatives))

colnames(OptOutMat) <-
  c("Std Dev", paste("$VaR:$",alpha0), paste("$VaR:$",alpha1), paste("$VaR:$",alpha2),
    paste("$ES:$",alpha0), paste("$ES:$",alpha1),  paste("$ES:$",alpha2) )
rownames(OptOutMat) <-
  c(paste("Optimal","$u_1$"), paste("Optimal","$u_2$"), paste("Optimal","$LME$"),
    "Std Dev",  paste("$VaR:$",alpha0),  paste("$VaR:$",alpha1), paste("$VaR:$",alpha2),
    paste("$ES:$",alpha0), paste("$ES:$",alpha1),  paste("$ES:$",alpha2))

time4<-Sys.time()
for (kindex in 1:length(RMAlternatives)) {

  f0.Vec <- function(upar){f0.a(u1=upar[1],u2=upar[2],a=0)[kindex]}
  f0.opt <- auglag(par=starter,                    # initial value
                   fn=f0.Vec,                     # objective function
                   hin=fineq,                     # inequality constraint
                   control.outer=list(method="nlminb",trace=FALSE))
  OptOutMat[1:2,kindex] <- f0.opt$par
  OptOutMat[3,kindex]   <- f0.opt$lambda[1]
  OptOutMat[4:10,kindex] <- f0.a(f0.opt$par[1],f0.opt$par[2],a=0)

}

save(OptOutMat, file="../ChapTablesData/Chap10/Exer1021Base.Rdata")
Sys.time() - time4     #Time difference of 23 mins
```

17.9.6 Exercise 10.5. Excess of Loss VaR Sensitivity, part (b)

```
# Use the baseline model to develop sensitivities
#  Start with scale of the second risk

load(file = "../ChapTablesData/Chap10/Exer1021Base.Rdata")

#  Set Simulation Parameters
nsim <- 20000000
set.seed(202020)
```

```
risk1shape <- 2
risk1scale <- 2000
risk2shape <- 3
risk2scale <- 2000
#  Independent normal random variables
rGaus    <- matrix(rnorm(2*nsim),nrow=nsim,ncol=2)
sigparam <- 0.5
Sigma.11 <- matrix(c(1,sigparam,sigparam,1),nrow=2,ncol=2)
rGaus.new <- rGaus %*% chol(Sigma.11)
UCop1    <- pnorm(rGaus.new[,1])
UCop2    <- pnorm(rGaus.new[,2])
#  Marginal Distribution Random variables
X1 <-          qgamma(p=UCop1, shape = risk1shape, scale = risk1scale)
X2 <- actuar::qpareto(p=UCop2, shape = risk2shape, scale = risk2scale)
alpha0 <- 0.95
alpha1 <- 0.85
alpha2 <- 0.75
rm(rGaus, rGaus.new, UCop1, UCop2)

# Objective Function
f0.a <- function(u1,u2,a){
  S <- pmin(X1,u1) +  pmin(X2*(1+a/100),u2)
  stddev  <- sd(S)
  Quan0   <- quantile(S, probs = alpha0, na.rm = TRUE)
  Quan1   <- quantile(S, probs = alpha1, na.rm = TRUE)
  Quan2   <- quantile(S, probs = alpha2, na.rm = TRUE)
  excess0 <- pmax(S-Quan0,0)
  CTE0    <- Quan0 + mean(excess0)/(1-alpha0)
  excess1 <- pmax(S-Quan1,0)
  CTE1    <- Quan1 + mean(excess1)/(1-alpha1)
  excess2 <- pmax(S-Quan2,0)
  CTE2    <- Quan2 + mean(excess2)/(1-alpha2)
  Output  <- c(stddev, Quan0,Quan1, Quan2, CTE0,CTE1, CTE2)
  return(Output)
}

#  Constraint Function
RTC.a <- function(u1,u2,a){
  TotalCost <-
    actuar::mgamma(order=1, shape = risk1shape, scale = risk1scale) +
    actuar::mpareto(order=1, shape = risk2shape, scale = risk2scale*(1+a/100))
  TotalRetained <-
    actuar::levgamma(limit = u1,shape = risk1shape, scale = risk1scale) +
    actuar::levpareto(limit = u2,shape = risk2shape, scale = risk2scale*(1+a/100))
  RTCost<- TotalCost - TotalRetained
  return(RTCost)
}

RMAlternatives <- 1:7

Scale2OutMat <- matrix(0,3,length(RMAlternatives))
```

```
colnames(Scale2OutMat) <-
  c("Std Dev", paste("VaR:",alpha0), paste("$VaR:$",alpha1),  paste("$VaR:$",alpha2),
    paste("$ES:$",alpha0), paste("$ES:$",alpha1),  paste("$ES:$",alpha2) )
rownames(Scale2OutMat) <-
  c(paste("Sensitivity","$u_1$"),
    paste("Sensitivity","$u_2$"),  paste("Sensitivity","$LME$"))

time4 <- Sys.time()
for (kindex in 1:length(RMAlternatives)) {
  uparam <- OptOutMat[1:2,kindex]
  LME <- OptOutMat[3,kindex]

  f0.Theta.a.Vec<- function(par){f0.a(u1=par[1],u2=par[2],a=par[3])[kindex]}
  f0.Theta.a  <- numDeriv::hessian(f=f0.Theta.a.Vec, x = c(uparam,0))

  RTC.a.vec   <- function(par){u1=par[1];u2=par[2];a=par[3]
                          RTC.a(u1,u2,a) }
  Deriv.eq    <- numDeriv::grad(f=RTC.a.vec, x = c(uparam,0))
  feq.Theta   <- Deriv.eq[1:2]
  feq.a       <- Deriv.eq[3]
  feq.Theta.a <- numDeriv::hessian(f=RTC.a.vec, x = c(uparam,0))

  LHS.924     <- c(-feq.a,-(f0.Theta.a[1:2,3] + feq.Theta.a[1:2,3]* LME) )
  SLA         <- f0.Theta.a[1:2,1:2] + LME*feq.Theta.a[1:2,1:2]

  EnvMat      <- matrix(0,ncol = 3, nrow = 3)
  EnvMat[2:3,1:2] <- SLA
  EnvMat[2:3,3]   <- feq.Theta -> EnvMat[1,1:2]

  if (sum(abs(LHS.924)) > 1e-8) {
    Scale2OutMat[,kindex] <- solve(EnvMat, b=LHS.924 ) }
}  # end kindex loop

save(Scale2OutMat, file="../ChapTablesData/Chap10/Exer1021BaseScale2.Rdata")
```

17.9.7 Exercise 10.5. Excess of Loss VaR Sensitivity, part (c)

```
# Use the baseline model to develop sensitivities
#  Start with scale of the second risk

load(file = "../ChapTablesData/Chap10/Exer1021Base.Rdata")

#
#  Set Simulation Parameters
nsim <- 20000000
set.seed(202020)
risk1shape <- 2
risk1scale <- 2000
```

```
risk2shape <- 3
risk2scale <- 2000
#  Independent normal random variables
rGaus    <- matrix(rnorm(2*nsim),nrow=nsim,ncol=2)
sigparam <- 0.5
Sigma.11 <- matrix(c(1,sigparam,sigparam,1),nrow=2,ncol=2)
rGaus.new <- rGaus %*% chol(Sigma.11)
UCop1    <- pnorm(rGaus.new[,1])
UCop2    <- pnorm(rGaus.new[,2])
#  Marginal Distribution Random variables
X1 <-          qgamma(p=UCop1, shape = risk1shape, scale = risk1scale)
X2 <- actuar::qpareto(p=UCop2, shape = risk2shape, scale = risk2scale)
alpha0 <- 0.95
alpha1 <- 0.85
alpha2 <- 0.75

# Objective Function
f0.a.data <- function(u1,u2,a,beginblock,endblock){
  S <- pmin(X1[beginblock:endblock],u1) +  pmin(X2[beginblock:endblock]*(1+a/100),u2)
  stddev  <- sd(S)
  Quan0   <- quantile(S, probs = alpha0, na.rm = TRUE)
  Quan1   <- quantile(S, probs = alpha1, na.rm = TRUE)
  Quan2   <- quantile(S, probs = alpha2, na.rm = TRUE)
  excess0 <- pmax(S-Quan0,0)
  CTE0    <- Quan0 + mean(excess0)/(1-alpha0)
  excess1 <- pmax(S-Quan1,0)
  CTE1    <- Quan1 + mean(excess1)/(1-alpha1)
  excess2 <- pmax(S-Quan2,0)
  CTE2    <- Quan2 + mean(excess2)/(1-alpha2)
  Output  <- c(stddev, Quan0, Quan1,Quan2, CTE0,CTE1,  CTE2)
  return(Output)
}

#  Constraint Function
RTC.a <- function(u1,u2,a){
  TotalCost <-
    actuar::mgamma(order=1, shape = risk1shape, scale = risk1scale) +
    actuar::mpareto(order=1, shape = risk2shape, scale = risk2scale*(1+a/100))
  TotalRetained <-
    actuar::levgamma(limit = u1,shape = risk1shape, scale = risk1scale) +
    actuar::levpareto(limit = u2,shape = risk2shape, scale = risk2scale*(1+a/100))
  RTCost<- TotalCost - TotalRetained
  return(RTCost)
}

nsim <- 1000000
RMAlternatives <- 1:7
numBoot <- 10
BootLength <- nsim/numBoot
Output.array <- array(0, dim = c(3,length(RMAlternatives), numBoot))
```

```
for (rBoot in 1:numBoot){
  beginblock = (rBoot-1)*BootLength+1
  endblock = rBoot*BootLength
  f0.a <- function(u1,u2,a){ f0.a.data(u1,u2,a,beginblock, endblock) }

Scale2OutMat <- matrix( 0,3,length(RMAlternatives) )

time4<-Sys.time()
for (kindex in 1:length(RMAlternatives)) {
  uparam <- OptOutMat[1:2,kindex]
  LME <- OptOutMat[3,kindex]

  f0.Theta.a.Vec<- function(par){f0.a(u1=par[1],u2=par[2],a=par[3])[kindex]}
  f0.Theta.a  <- numDeriv::hessian(f=f0.Theta.a.Vec, x = c(uparam,0))

  RTC.a.vec   <- function(par){ u1=par[1];u2=par[2];a=par[3]
                          RTC.a(u1,u2,a) }
  Deriv.eq    <- numDeriv::grad(f=RTC.a.vec, x = c(uparam,0))
  feq.Theta   <- Deriv.eq[1:2]
  feq.a       <- Deriv.eq[3]
  feq.Theta.a <- numDeriv::hessian(f=RTC.a.vec, x = c(uparam,0))

  LHS.924     <- c(-feq.a, -(f0.Theta.a[1:2,3] + feq.Theta.a[1:2,3]* LME) )
  SLA         <- f0.Theta.a[1:2,1:2] + LME*feq.Theta.a[1:2,1:2]

  EnvMat      <- matrix(0,ncol = 3, nrow = 3)
  EnvMat[2:3,1:2] <- SLA
  EnvMat[2:3,3]   <- feq.Theta -> EnvMat[1,1:2]

  if (sum(abs(LHS.924)) > 1e-8) {Scale2OutMat[,kindex] <- solve(EnvMat, b=LHS.924 ) }
  }  # end kindex loop
Output.array[,,rBoot] <- Scale2OutMat
}  # end rBoot loop

Output.mean <- Scale2OutMat*0 -> Output.sq
for (rBoot in 1:numBoot){
  Output.mean <- Output.mean + Output.array[,,rBoot]
  Output.sq   <- Output.sq   + Output.array[,,rBoot]* Output.array[,,rBoot]
  }
Output.mean <- Output.mean/numBoot
Output.var  <- Output.sq /numBoot - Output.mean*Output.mean
Output.var  <- Output.var*(Output.var>0)
Output.se   <- sqrt(Output.var)/sqrt(numBoot)

colnames(Output.mean) <-
  c("Std Dev", paste("$VaR:$",alpha0), paste("$VaR:$",alpha1),  paste("$VaR:$",alpha2),
    paste("$ES:$",alpha0),paste("$ES:$",alpha1),  paste("$ES:$",alpha2) ) ->
       colnames(Output.se)
rownames(Output.mean) <-
  c(paste("Sensitivity","$u_1$"),
    paste("Sensitivity","$u_2$"),  paste("Sensitivity","$LME$"))->
```

```
       rownames(Output.se)

save(Output.mean, Output.se ,
     file="../ChapTablesData/Chap10/Exer1021BaseScale2Repeat.Rdata")
```

17.9.8 Exercise 10.5. Excess of Loss VaR Sensitivity, part (d)

```
# Use the baseline model to develop sensitivities

load(file = "../ChapTablesData/Chap10/Example1021Base.Rdata")

#
nsim <- 20000000
set.seed(202020)
risk1shape <- 2
risk1scale <- 2000
risk2shape <- 3
risk2scale <- 2000
#  Independent normal random variables
rGaus    <- matrix(rnorm(2*nsim),nrow=nsim,ncol=2)
sigparam <- 0.5
Sigma.11 <- matrix(c(1,sigparam,sigparam,1),nrow=2,ncol=2)
rGaus.new <- rGaus %*% chol(Sigma.11)
UCop1    <- pnorm(rGaus.new[,1])
UCop2    <- pnorm(rGaus.new[,2])
#  Marginal Distribution Random variables
X1 <-          qgamma(p=UCop1, shape = risk1shape, scale = risk1scale)
X2 <- actuar::qpareto(p=UCop2, shape = risk2shape, scale = risk2scale)
alpha0 <- 0.95
alpha1 <- 0.85
alpha2 <- 0.75

# Objective Function
f0.aParameters <- function(u1,u2,Shape1, Shape2, Scale1, Scale2){
  X1 <-          qgamma(p=UCop1, shape = Shape1, scale = Scale1)
  X2 <- actuar::qpareto(p=UCop2, shape = Shape2, scale = Scale2)
  S <- pmin(X1,u1) +  pmin(X2,u2)
  stddev  <- sd(S)
  Quan0   <- quantile(S, probs = alpha0, na.rm = TRUE)
  Quan1   <- quantile(S, probs = alpha1, na.rm = TRUE)
  Quan2   <- quantile(S, probs = alpha2, na.rm = TRUE)
  excess0 <- pmax(S-Quan0,0)
  CTE0    <- Quan0 + mean(excess0)/(1-alpha0)
  excess1 <- pmax(S-Quan1,0)
  CTE1    <- Quan1 + mean(excess1)/(1-alpha1)
  excess2 <- pmax(S-Quan2,0)
  CTE2    <- Quan2 + mean(excess2)/(1-alpha2)
  Output  <- c(stddev, Quan1, Quan2, CTE1, CTE2)
  return(Output)
```

```
}

f0.a <- function(u1,u2,a){
  risk1shape.a <- risk1shape*(1+a/100)
  f0.aParameters(u1,u2,Shape1=risk1shape.a, Shape2=risk2shape,
                 Scale1 = risk2scale, Scale2=risk2scale)
  }

RMAlternatives <- 1:4
Shape1OutMat <- matrix(0,3,length(RMAlternatives))
colnames(Shape1OutMat) <- c("Std Dev", paste("$ES:$",alpha0),
                            paste("$ES:$",alpha1),  paste("$ES:$",alpha2) )
rownames(Shape1OutMat) <- c(paste("Sensitivity","$u_1$"),
                            paste("Sensitivity","$u_2$"),
                            paste("Sensitivity","$LME$"))

time4<-Sys.time()
for (kindex in 1:length(RMAlternatives)) {

  uparam <- OptOutMat[1:2,kindex]
  LME <- OptOutMat[3,kindex]

  f0.Theta.a.Vec<- function(par){f0.a(u1=par[1],u2=par[2],a=par[3])[kindex]}
  f0.Theta.a  <- numDeriv::hessian(f=f0.Theta.a.Vec, x = c(uparam,0))

  RTC.a.vec   <- function(par){u1=par[1];u2=par[2];a=par[3]
                               RTC.a(u1,u2,a) }
  Deriv.eq    <- numDeriv::grad(f=RTC.a.vec, x = c(uparam,0))
  feq.Theta   <- Deriv.eq[1:2]
  feq.a       <- Deriv.eq[3]
  feq.Theta.a <- numDeriv::hessian(f=RTC.a.vec, x = c(uparam,0))

  LHS.924     <- c(-feq.a,-(f0.Theta.a[1:2,3] + feq.Theta.a[1:2,3]* LME) )
  #LHS.924     <- c(-feq.a,-f0.Theta.a[1:2,3] )  NEW
  SLA         <- f0.Theta.a[1:2,1:2] + LME*feq.Theta.a[1:2,1:2]

  EnvMat      <- matrix(0,ncol = 3, nrow = 3)
  EnvMat[2:3,1:2] <- SLA
  EnvMat[2:3,3]   <- feq.Theta -> EnvMat[1,1:2]

  if (sum(abs(LHS.924)) > 1e-8) {
    Shape1OutMat[,kindex] <- solve(EnvMat, b=LHS.924 ) }

}

save(Shape1OutMat, file="../ChapTablesData/Chap10/Example1021BaseShape1.Rdata")

Sys.time() - time4  #Time difference of 12  mins
```

17.10 Chapter Eleven Exercise Solutions

17.10.1 Exercise 11.1. de Finetti Optimal Retention Proportions

a. To put this in standard form, we use the inequalities

$$\begin{array}{lll} f_{1,in}(\mathbf{c}) & = (\mathrm{E}\ \mathbf{X})'(\mathbf{1}-\mathbf{c}) - RTC_{max} \leq 0 & \\ f_{2,in,i}(\mathbf{c}) & = c_i - 1 \leq 0, & i = 1,\ldots,n \\ f_{3,in,i}(\mathbf{c}) & = -c_i \leq 0, & i = 1,\ldots,n \end{array}$$

With this, the Lagrangian is

$$LA \quad = \tfrac{1}{2}\mathbf{c}'\mathbf{\Sigma c} + LMI_1[(\mathrm{E}\ \mathbf{X})'(\mathbf{1}-\mathbf{c}) - RTC_{max}] + \textstyle\sum_{i=1}^n LMI_{2,i}\ (c_i - 1) - \sum_{i=1}^n LMI_{3,i}\ c_i.$$

Taking derivatives, we have

$$\begin{array}{ll} \frac{\partial}{\partial \mathbf{c}} LA & = \mathbf{\Sigma c} - LMI_1\ [\mathrm{E}\ \mathbf{X}] + \mathbf{LMI}_2 - \mathbf{LMI}_3. \\ \Rightarrow & \frac{\partial}{\partial c_i} LA = \mathbf{1}_i'\mathbf{\Sigma c} - LMI_1\ [\mathrm{E}\ X_i] + LMI_{2i} - LMI_{3i}. \end{array}$$

Note that, with zero correlations, we have $\mathbf{1}_i'\mathbf{\Sigma c} = \sigma_i^2 c_i$.

Assuming zero correlations, the KKT conditions are as in Display (11.13).

b. Now suppose $0 < c_i^* < 1$. Then, from KKT, we have $LMI_{2i}^* = LMI_{3i}^* = 0$ and

$$\begin{array}{c} \sigma_i^2 c_i^* - LMI_1^*\ [\mathrm{E}\ X_i] = 0 \\ \Rightarrow LMI_1^* = \frac{\sigma_i^2 c_i^*}{\mathrm{E}\ X_i}. \end{array}$$

Suppose $ c_iˆ* =0$. Then, from KKT, we have $LMI_{2i}^* = 0$ and

$$\begin{array}{c} -LMI_1^*\ [\mathrm{E}\ X_i] - LMI_{3i}^* = 0 \\ \Rightarrow LMI_{3i}^* = -LMI_1^*\ [\mathrm{E}\ X_i] \leq 0 \\ \Rightarrow LMI_{3i}^* = 0. \end{array}$$

Suppose $c_i^* = 1$. Then, from KKT, we have $LMI_{3i}^* = 0$ and

$$\begin{array}{c} \sigma_i^2 - LMI_1^*\ [\mathrm{E}\ X_i] + LMI_{2i}^* = 0 \\ \Rightarrow LMI_{2i}^* = LMI_1^*\ [\mathrm{E}\ X_i] - \sigma_i^2. \end{array}$$

We summarize these results in Display (11.14).

c. Again, start with the KKT conditions in Display (11.13).

Suppose $LMI_{3i}^* > 0$. Then, from *KKT*, we have $c_i^* = 0$. In the same way, if $LMI_{2i}^* > 0$. Then, we have $c_i^* = 1$. So, apparently, we cannot have both $LMI_{3i}^* > 0$ and $LMI_{2i}^* > 0$.

How about the case $LMI_{2i}^* = LMI_{3i}^* = 0$? For this case, we have

$$\begin{gathered}\sigma_i^2 c_i^* - LMI_1^* \ [\mathrm{E}\ X_i] = 0 \\ \Rightarrow LMI_1^* = \tfrac{\sigma_i^2 c_i^*}{\mathrm{E}\ X_i}.\end{gathered}$$

Now suppose $LMI_1^* = 0$. Then,

$$\sigma_i^2 c_i^* + LMI_{2i}^* - LMI_{3i}^* = 0 \quad i = 1, \ldots, n$$

- If in addition $LMI_{3i}^* > 0$, then $c_i^* = 0$ meaning that $LMI_{2i}^* = LMI_{3i}^* > 0$ which cannot happen.
- If in addition $LMI_{2i}^* > 0$, then $c_i^* = 1$ and $LMI_{3i}^* = 0$. Thus, $LMI_{2i}^* = -\sigma_i^2$ which cannot happen.
- Thus, if $LMI_1^* = 0$, then $LMI_{2i}^* = LMI_{3i}^* = 0$ meaning that $c_i^* = 0$.

Now suppose $LMI_1^* > 0$. Then,

- If in addition $LMI_{3i}^* > 0$, then $c_i^* = 0$, $LMI_{2i}^* = 0$, and $-LMI_1^* \ [\mathrm{E}\ X_i] - LMI_{3i}^* = 0$. This violates the *KKT* conditions.
- If in addition $LMI_{2i}^* > 0$, then $c_i^* = 1$, $LMI_{3i}^* = 0$, and $\sigma_i^2 - LMI_1^* \ [\mathrm{E}\ X_i] + LMI_{2i}^* = 0$.
- If in addition $LMI_{2i}^* = LMI_{3i}^* = 0$, then $\sigma_i^2 c_i^* - LMI_1^* \ [\mathrm{E}\ X_i] = 0$.

These results are summarized in Display (11.15).

d. From Display (11.15), no feasible solution is likely to exist when $LMI_1^* = 0$. When $LMI_1^* > 0$, the risk transfer condition is binding so that $\sum_i (1 - c_i^*)\mathrm{E}(X_i) = RTC_{max}$. This suggests finding LMI_1^* as the solution of the equation (11.16).

$$\begin{gathered}\sigma_i^2 c_i^* - LMI_1^* \ [\mathrm{E}\ X_i] = 0 \\ \Rightarrow c_i^* = \tfrac{LMI_1^* \ [\mathrm{E}\ X_i]}{\sigma_i^2} \\ \textstyle\sum_i \mathrm{E}(X_i) - \sum_i c_i^* \mathrm{E}(X_i) = RTC_{max} \\ \textstyle\sum_i \mathrm{E}(X_i) - \sum_i \min\left\{1, \tfrac{LMI_1^* \ [\mathrm{E}\ X_i]}{\sigma_i^2}\right\} \mathrm{E}(X_i) = RTC_{max}\end{gathered}$$

To check this, let us examine what happens when the ratio is large.

$$\begin{gathered}\text{if } \tfrac{LMI_1^* \ [\mathrm{E}\ X_i]}{\sigma_i^2} \geq 1 \qquad \text{then } LMI_1^* \ [\mathrm{E}\ X_i] \geq \sigma_i^2 > \sigma_i^2 c_i^* \\ LMI_1^* \ [\mathrm{E}\ X_i] - \sigma_i^2 c_i^* > 0 \qquad \Rightarrow LMI_{2,i}^* > 0 \text{ and } c_i^* = 1\end{gathered}$$

In the same way, let us examine the case when the ratio is small.

$$\begin{gathered}\text{if } \tfrac{LMI_1^* \ [\mathrm{E}\ X_i]}{\sigma_i^2} < 1 \qquad \text{then } LMI_1^* \ [\mathrm{E}\ X_i] < \sigma_i^2 \\ \text{we can find } c_i^* \text{ such that } LMI_1^* \ [\mathrm{E}\ X_i] = \sigma_i^2 c_i^* \qquad \Rightarrow LMI_{2,i}^* = 0\end{gathered}$$

Suppose		*KKT* Results	
$LMI_1^* > 0$	$LMI_1^*\ [\mathrm{E}\ X_i] - \sigma_i^2 > 0$	$LMI_{2i}^* > 0$	$c_i^* = 1$
$LMI_1^* > 0$	$LMI_1^*\ [\mathrm{E}\ X_i] - \sigma_i^2 \leq 0$	$LMI_{2i}^* = 0$	$0 < c_i^* < 1$

17.10.2 Exercise 11.2. Lasso Regression Conditions

a. Using equation (3.10), the Lagrangian can be expressed as

$$\begin{aligned} LA(\boldsymbol{\beta}^+, \boldsymbol{\beta}^-) = \quad & f_{0,ls}(\boldsymbol{\beta}^+, \boldsymbol{\beta}^-) + LMI_1[\textstyle\sum_{j=1}^p (\beta_j^+ + \beta_j^-) - c_{lasso}] \\ & - \textstyle\sum_{j=1}^p LMI_{1+j}^+\ \beta_j^+ - \sum_{j=1}^p LMI_{1+j}^-\ \beta_j^-. \end{aligned}$$

b. For the *KKT* conditions, we can use

$$\begin{aligned} \partial_{\boldsymbol{\beta}^+}\ f_{0,ls}(\boldsymbol{\beta}^+, \boldsymbol{\beta}^-) \quad & = \tfrac{1}{2} \textstyle\sum_{i=1}^n \partial_{\boldsymbol{\beta}^+}\ [y_i^* - \mathbf{x}_i' \boldsymbol{\beta}^+]^2 \\ & = -\textstyle\sum_{i=1}^n\ [y_i^* - \mathbf{x}_i' \boldsymbol{\beta}^+] \mathbf{x}_i \\ & = -\textstyle\sum_{i=1}^n\ [y_i - \mathbf{x}_i' (\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-)] \mathbf{x}_i \\ & = -\mathbf{X}'\mathbf{y} + \mathbf{X}'\mathbf{X}(\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-) \\ & = -\mathbf{X}' \left[\mathbf{y} - \mathbf{X}(\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-)\right] = -\mathbf{X}' \left[\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right] \\ & = -\mathbf{X}'\mathbf{e}, \end{aligned}$$

where $y_i^* = y_i + \mathbf{x}_i' \boldsymbol{\beta}^-$. To ease notation, let the jth row of this be denoted as $\nabla_j^+ f_{0,ls}(\boldsymbol{\beta}^+, \boldsymbol{\beta}^-) = \nabla_j^+ f_{0,ls}$.

In the same way, with $y_i^{**} = y_i - \mathbf{x}_i' \boldsymbol{\beta}^+$

$$\begin{aligned} \partial_{\boldsymbol{\beta}^-}\ f_{0,ls}(\boldsymbol{\beta}^+, \boldsymbol{\beta}^-) \quad & = \tfrac{1}{2} \textstyle\sum_{i=1}^n \partial_{\boldsymbol{\beta}^-}\ [y_i^{**} + \mathbf{x}_i' \boldsymbol{\beta}^-]^2 \\ & = \textstyle\sum_{i=1}^n\ [y_i^{**} + \mathbf{x}_i' \boldsymbol{\beta}^-] \mathbf{x}_i \\ & = \textstyle\sum_{i=1}^n\ [y_i - \mathbf{x}_i' (\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-)] \mathbf{x}_i \\ & = -\partial_{\boldsymbol{\beta}^+}\ f_{0,ls}(\boldsymbol{\beta}^+, \boldsymbol{\beta}^-). \end{aligned}$$

c. Taking derivatives of the Lagrangian yields

$$\begin{aligned} \partial_{\beta_j^+}\ LA(\boldsymbol{\beta}^+, \boldsymbol{\beta}^-) \quad & = \nabla_j^+ f_{0,ls} + LMI_1 - LMI_{1+j}^+ \\ \partial_{\beta_j^-}\ LA(\boldsymbol{\beta}^+, \boldsymbol{\beta}^-) \quad & = \nabla_j^- f_{0,ls} + LMI_1 - LMI_{1+j}^-. \end{aligned}$$

By making c_{lasso} sufficiently small, the constraint on the sum of absolute values is active, meaning that $LMI_1^* > 0$ at the optimum.

Further, if $\beta_j^+ > 0$ at the optimum, $LMI_{1+j}^+ = 0$. In this case, $LMI_1 = -\nabla_j^+ f_{0,ls}$. This means that $-\nabla_j^+ f_{0,ls} = \nabla_j^- f_{0,ls} > 0$ so that when we look at

$$\partial_{\beta_j^-}\ LA(\boldsymbol{\beta}^+, \boldsymbol{\beta}^-) = \nabla_j^- f_{0,ls} + LMI_1 - LMI_{1+j}^-,$$

we have $LMI_{1+j}^{-} > 0$, meaning that $\beta_j^{-} = 0$. Thus, there is no contradiction; a value of $\beta_j^{+} > 0$ implies $\beta_j^{-} = 0$, as one would hope.

d. If $\mathbf{x}^{(j)'}\mathbf{e} = \nabla_j^{+} f_{0,ls} = 0$, then $\partial_{\beta_j^{+}} LA(\boldsymbol{\beta}^{+}, \boldsymbol{\beta}^{-}) = LMI_1 - LMI_{1+j}^{+} = 0$ by the *KKT* conditions. Because $LMI_1 > 0$, we have $LMI_{1+j}^{+} > 0$ and so, by the *KKT* conditions, $\beta_j^{+} = 0$. In the same way, $\beta_j^{-} = 0$ which is sufficient for the result.

17.10.3 Exercise 11.3. Linearity of Lasso Regression Conditions

a. Taking a derivative of the Lagrangian, we have

$$\begin{aligned} \partial_{\beta_j^{+}} LA(\boldsymbol{\beta}^{+}, \boldsymbol{\beta}^{-}) &= \nabla_j^{+} f_{0,ls} + \lambda - LMI_{1+j}^{+} \\ &= \mathbf{x}^{(j)'}\mathbf{e} + \lambda - LMI_{1+j}^{+}. \end{aligned}$$

If $\beta_j^{+} > 0$, then $LMI_{1+j}^{+} = 0$ by the *KKT* conditions and so $\lambda = -\mathbf{x}^{(j)'}\mathbf{e}$ which is sufficient for the result. The case of $\beta_j^{-} > 0$ can be established in the same way.

$$\partial_{\boldsymbol{\beta}^{+}} f_{0,ls}(\boldsymbol{\beta}^{+}, \boldsymbol{\beta}^{-}) = -\mathbf{X}'[\mathbf{y} - \mathbf{X}\boldsymbol{\beta}]$$

b. Starting with the relationship

$$\partial_{\boldsymbol{\beta}^{+}} f_{0,ls}(\boldsymbol{\beta}^{+}, \boldsymbol{\beta}^{-}) = -\mathbf{X}'[\mathbf{y} - \mathbf{X}\boldsymbol{\beta}] = -\partial_{\boldsymbol{\beta}^{-}} f_{0,ls}(\boldsymbol{\beta}^{+}, \boldsymbol{\beta}^{-}),$$

the Hessian is

$$\begin{aligned} \nabla^2 LA &= \begin{pmatrix} \partial_{\boldsymbol{\beta}^{+}}\partial'_{\boldsymbol{\beta}^{+}} LA(\boldsymbol{\beta}^{+}, \boldsymbol{\beta}^{-}) & \partial_{\boldsymbol{\beta}^{+}}\partial'_{\boldsymbol{\beta}^{-}} LA(\boldsymbol{\beta}^{+}, \boldsymbol{\beta}^{-}) \\ \partial_{\boldsymbol{\beta}^{-}}\partial'_{\boldsymbol{\beta}^{+}} LA(\boldsymbol{\beta}^{+}, \boldsymbol{\beta}^{-}) & \partial_{\boldsymbol{\beta}^{-}}\partial'_{\boldsymbol{\beta}^{-}} LA(\boldsymbol{\beta}^{+}, \boldsymbol{\beta}^{-}) \end{pmatrix} \\ &= \begin{pmatrix} \partial_{\boldsymbol{\beta}^{+}}\partial'_{\boldsymbol{\beta}^{+}} f_{0,ls}(\boldsymbol{\beta}^{+}, \boldsymbol{\beta}^{-}) & \partial_{\boldsymbol{\beta}^{+}}\partial'_{\boldsymbol{\beta}^{-}} f_{0,ls}(\boldsymbol{\beta}^{+}, \boldsymbol{\beta}^{-}) \\ \partial_{\boldsymbol{\beta}^{-}}\partial'_{\boldsymbol{\beta}^{+}} f_{0,ls}(\boldsymbol{\beta}^{+}, \boldsymbol{\beta}^{-}) & \partial_{\boldsymbol{\beta}^{-}}\partial'_{\boldsymbol{\beta}^{-}} f_{0,ls}(\boldsymbol{\beta}^{+}, \boldsymbol{\beta}^{-}) \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{X}'\mathbf{X} & -\mathbf{X}'\mathbf{X} \\ -\mathbf{X}'\mathbf{X} & \mathbf{X}'\mathbf{X} \end{pmatrix}. \end{aligned}$$

c.

$$\begin{aligned} \delta &= \lambda_\delta - \lambda \\ &= -\mathbf{x}^{(j)'}[\mathbf{y} - \mathbf{X}\boldsymbol{\beta}_\delta] + \mathbf{x}^{(j)'}[\mathbf{y} - \mathbf{X}\boldsymbol{\beta}] \\ &= \mathbf{x}^{(j)'}\mathbf{X}[\boldsymbol{\beta}_\delta - \boldsymbol{\beta}] \end{aligned}$$

d. Perturbation Sensitivity Proposition from Section 10.1, we have

$$-f_{0,\theta a}(\boldsymbol{\theta}, a) = SLA_{\theta\theta}[\boldsymbol{\theta}, a]\,\partial_a \boldsymbol{\theta}(a)$$

where $a = \lambda$, $\boldsymbol{\theta} = (\boldsymbol{\beta}^{+}, \boldsymbol{\beta}^{-})$,

$$\begin{aligned} f_{0,\theta a}(\boldsymbol{\theta}, a) &= \partial_\lambda \partial_{\boldsymbol{\theta}} \left\{ f_{0,ls}(\boldsymbol{\beta}^+, \boldsymbol{\beta}^-) + \lambda \textstyle\sum_{j=1}^p (\beta_j^+ + \beta_j^-) \right\} \\ &= \partial_{(\boldsymbol{\beta}^+, \boldsymbol{\beta}^-)} \textstyle\sum_{j=1}^p (\beta_j^+ + \beta_j^-), \end{aligned}$$

and

$$\begin{aligned} SLA_{\theta\theta}\left[\boldsymbol{\theta}, a\right] &= \partial_{\boldsymbol{\theta}} \partial_{\boldsymbol{\theta}'} \left\{ f_{0,ls}(\boldsymbol{\beta}^+, \boldsymbol{\beta}^-) + \lambda \textstyle\sum_{j=1}^p (\beta_j^+ + \beta_j^-) \right\} \\ &= \nabla^2 LA. \end{aligned}$$

Restricting the rows and columns to be those in the active set (where the corresponding parameters are not zero) is sufficient for part *d*.

17.10.4 Exercise 11.4. Standard Errors of Ratios

a. From equation (11.4.2), the risk measure relative marginal change for the expected shortfall can be written as

$$\begin{aligned} RM_{ES,j}^2(\boldsymbol{\theta}) &= \tfrac{1}{(1-\alpha)} \tfrac{\mathrm{E}_R\left\{\tilde{k}_{R\theta} \;\; g_{Rj}\right\}}{\bar{g}_j} \\ &= \bar{x}_1 / \bar{x}_2 \quad , \end{aligned}$$

where $x_{1r} = \tilde{k}_{rj} \;\; g_{rj}$ and $x_{2r} = (1-\alpha) g_{rj}$. Further, recall that $\tilde{k}_{r\theta} = 1 - K\left([VaR_{Rk} - g(\mathbf{X}_r; \boldsymbol{\theta})]/b\right)$.

b.

$$\begin{aligned} r_{12} &\approx \tfrac{1}{\sqrt{R^2 s_1^2 \;\; s_2^2}} \left[\textstyle\sum_{r=1}^R (x_{1r} x_{2r}) - R \; \bar{x}_1 \bar{x}_2 \right] \\ &= \tfrac{1}{\sqrt{R^2 \bar{x}_1(1-\bar{x}_1) \;\; \bar{x}_2(1-\bar{x}_2)}} \left[R\bar{x}_{12} - R\bar{x}_1\bar{x}_2 \right] \\ &= \tfrac{\bar{x}_{12} - \bar{x}_1\bar{x}_2}{\sqrt{\bar{x}_1(1-\bar{x}_1) \;\; \bar{x}_2(1-\bar{x}_2)}} \end{aligned}$$

c.

$$\begin{aligned} r_{12} &= \tfrac{\bar{x}_{12} - \bar{x}_1\bar{x}_2}{\sqrt{\bar{x}_1(1-\bar{x}_1) \;\; \bar{x}_2(1-\bar{x}_2)}} \\ &= \tfrac{\bar{x}_2(1-\bar{x}_1)}{\sqrt{\bar{x}_1(1-\bar{x}_1) \;\; \bar{x}_2(1-\bar{x}_2)}} \\ &= \sqrt{\tfrac{\bar{x}_2(1-\bar{x}_1)}{\bar{x}_1 \;\; (1-\bar{x}_2)}} \\ &= \sqrt{\tfrac{(1-\bar{x}_1)/\bar{x}_1}{(1-\bar{x}_2)/\bar{x}_2}} \end{aligned}$$

17.10.5 Exercise 11.5. Quantile Regression Approach

a. For the quantile regression problem, we can express the Lagrangian as

$$LA_{QR}(z_0, \boldsymbol{\theta}) = f_{QR,0}(z_0, \boldsymbol{\theta}) + LMI_{1,QR}[RTC_R(\boldsymbol{\theta}) - RTC_{max}].$$

Recall from Section 11.4.2 that we established $\partial_{\theta_j} RTC_R(\boldsymbol{\theta}) = -\bar{g}_j$. With this, we have

$$\partial_{\theta_j} LA_{QR} \quad = \partial_{\theta_j} f_{QR,0}(z_0, \boldsymbol{\theta}) - \bar{g}_j LMI_{1,QR},$$

confirming the first expression. With equation (7.12), we can express the Lagrangian of the expected shortfall as

$$\begin{aligned} LA_{ES} &= f_{ES,0}(z_0, \boldsymbol{\theta}) + LMI_{1,ES}[RTC_R(\boldsymbol{\theta}) \leq RTC_{max}] \\ &= \mathrm{E}_R\ g(\mathbf{X}; \boldsymbol{\theta}) + \tfrac{1}{1-\alpha} f_{QR,0}(z_0, \boldsymbol{\theta}) + LMI_{1,ES}[RTC_R(\boldsymbol{\theta}) - RTC_{max}]. \end{aligned}$$

Taking a partial derivative with respect to a retention parameter yields

$$\begin{aligned} \partial_{\theta_j} LA_{ES} &= \partial_{\theta_j} \mathrm{E}_R\ g(\mathbf{X}; \boldsymbol{\theta}) + \tfrac{1}{1-\alpha} \partial_{\theta_j} f_{QR,0}(z_0, \boldsymbol{\theta}) + LMI_{1,ES} \partial_{\theta_j} RTC_R(\boldsymbol{\theta}) \\ &= \tfrac{1}{1-\alpha} \partial_{\theta_j} f_{QR,0}(z_0, \boldsymbol{\theta}) - \bar{g}_j [LMI_{1,ES} - 1], \end{aligned}$$

confirming the second expression.

17.11 Chapter Twelve Exercise Solutions

17.11.1 Exercise 12.1 Reinsurance Expected Payments

```
#  Exercise 12.1
load(file= "..//ChapTablesData/Chap12/Example1211.Rdata")
set.seed(2023)
nsim    <- 5000000
params  <- as.numeric(Uni.EstResults[,1])
rho     <- 0
cop     <- copula::normalCopula(param = rho,dim=2)
U       <- copula::rCopula(n=nsim, copula = cop)
SimLoss <- actuar::qpareto(U[,1], shape=params[1], scale=params[2]*1000)
SimAlae <- actuar::qpareto(U[,2], shape=params[3], scale=params[4]*1000)
zLoss   <- qnorm(U[,1])
zAlae   <- qnorm(U[,2])
zVec    <- cbind(zLoss,zAlae)

Sigma      <- matrix(c(1,rho,rho,1), nrow=2, ncol=2)
SigmaChol  <- chol(Sigma)
SigmaInv   <- solve(Sigma)
DerivSigma <- matrix(c(0,1,1,0),nrow=2,ncol=2)
Correction <- tr(SigmaInv %*% DerivSigma)
ZVecStar   <- SigmaInv %*% t(zVec)
zTzP       <- ZVecStar * (DerivSigma %*% ZVecStar)
Weight     <- colSums(zTzP)
Limit      <- c(10,100,500,1000)*1000
R.L        <- c(0,0.25,0.5,0.75,0.95)
IndRePrems <- matrix(0,nrow = length(Limit),ncol = length(R.L))
for (i in 1:length(Limit)){
  for (j in 1:length(R.L)){Lim = Limit[i]
```

```
  Ret   <- R.L[j]*Lim
  g.R.L <- (Ret<SimLoss)*(SimLoss<Lim)*
     (SimLoss-Ret + (SimLoss-Ret)/SimLoss*SimAlae) +
     (SimLoss>Lim)* (Lim-Ret + (Lim-Ret)/Lim*SimAlae)
  FirstTerm     <- g.R.L*Weight
  IndRePrems[i,j]   <- mean(g.R.L)
  } }
colnames(IndRePrems) <- c("0.0", "0.25", "0.50", "0.75", "0.95")
rownames(IndRePrems) <- c("10,000", "100,000", "500,000", "1,000,000")
save(IndRePrems,
     file="../ChapTablesData/Chap12/Exercise1231.Rdata")
```

17.11.2 Exercise 12.2. Two Risk Portfolio

a. When $p = 2$, the portfolio ($S = X_1 + X_2$) distribution function is

$$\begin{aligned} F(x;\rho) &= \Pr(X_1 + X_2 \leq x) = \Pr(F_1^{-1}(V_1) + F_2^{-1}(V_2) \leq x) \\ &= \int_0^{F_2(x)} \left(\int I[v_1 \leq F_1(x - F_2^{-1}(v_2))] \; c(v_1, v_2) dv_1 \right) dv_2 \\ &= \int_0^{F_2(x)} C_2(F_1(x - F_2^{-1}(v_2)), v_2) dv_2 \\ &= \int_0^{x} C_2(F_1(x - z), F_2(z)) \; f_2(z) \; dz. \end{aligned}$$

The last equality assumes X_2 is a continuous random variable. This expression relies on derivatives of copula functions. To be explicit, define

$$C_1(u, v) = \frac{\partial}{\partial u} C(u, v) = \frac{\partial}{\partial u} \int_0^u \int_0^v c(z_1, z_2) \; dz_1 dz_2 = \int_0^v c(u, z_2) \; dz_2$$

and similarly for C_2. Assuming the copula is symmetric in its arguments (the usual assumption), we have

$$C_2(u, v) = \frac{\partial}{\partial v} C(u, v) = \int_0^u c(z_1, v) \; dz_1 = \int_0^u c(v, z_1) \; dz_1 = C_1(v, u).$$

With this, we have

$$\begin{aligned} F_x(x;\rho) &= \tfrac{\partial}{\partial x} F(x;\rho) \\ &= \partial_x \textstyle\int_0^{F_2(x)} C_2(F_1(x - F_2^{-1}(u)), u) du \\ &= \textstyle\int_0^{F_2(x)} \partial_x C_2(F_1(x - F_2^{-1}(u)), u) du \\ &= \textstyle\int_0^{F_2(x)} c(F_1(x - F_2^{-1}(u)), u) f_1(x - F_2^{-1}(u)) du \\ &= \textstyle\int_0^{x} c(F_1(x - z), F_2(z)) f_1(x - z) f_2(z) dz \end{aligned}$$

The fourth equality assumes continuity of X_1 and the fifth assumes continuity of X_2. The fifth equality is based on the transform $u = F_2(z)$.

Further,

$$\begin{aligned} F_\rho(x;\rho) &= \partial_\rho F(x;\rho) \\ &= \partial_\rho \int_0^{F_2(x)} C_2(F_1(x - F_2^{-1}(u)), u)du \\ &= \int_0^{F_2(x)} \partial_\rho \; C_2(F_1(x - F_2^{-1}(u)), u)du \\ &= \int_0^{x} \partial_\rho \; C_2(F_1(x - z), F_2(z)) f_2(z)dz \end{aligned}$$

b. Consider uniform marginals and the Gaussian (normal) copula with partial/conditional distribution function

$$C_2(v_1, v_2) = C(v_1|v_2) = \Phi\left(\frac{\Phi^{-1}(v_1) - \rho\Phi^{-1}(v_2)}{\sqrt{1-\rho^2}}\right),$$

using ρ for a dependence parameter. For other details in the bivariate case, see Appendix Section 14.1.1. For example, using the normal scores $z_j = \Phi^{-1}(v_j)$, $j = 1, 2$, we have

$$\begin{aligned} F_\rho(x;\rho) &= \int_0^x \partial_\rho \; C_2(x - z, z) \; dz \\ &= \frac{1}{(1-\rho^2)^{3/2}} \int_0^x \phi\left(\frac{\Phi^{-1}(x-z) - \rho\Phi^{-1}(z)}{\sqrt{1-\rho^2}}\right) \left[\rho\Phi^{-1}(x - z) - \Phi^{-1}(z)\right] \; dz. \end{aligned}$$

At $\rho = 0$, we have $F_\rho(x;\rho) < 0$ and so $\partial_\rho q(\rho) > 0$, that is, the quantile increases as ρ increases (at zero).

17.11.3 Exercise 12.3. Portfolio Distribution with Gamma and Pareto Risks

```
#  Exercise 12.3

rhoparam<- c(-0.9,-.3,0.0,0.3,0.9)

F1xz  <- function(x,z){F1(x-q2(z))}
f1xz  <- function(x,z){f1(x-q2(z))}
QuanPortVec <- matrix(0,length(rhoparam),1) -> densityPortVec -> derivCorrPortVec

for (j in 1:length(rhoparam)) {
  copulaParam <- copula::iRho(copula::normalCopula(param=1, dim = 2), rho=rhoparam[j])
  cop  <- VineCopula::BiCop(family = 1, par = rhoparam[j])
  cop2 <- VineCopula::BiCop(family = 1, par = copulaParam)
  # family = "5" - Frank, family = "1" - Normal, family = "2" - t
  #  cop <- BiCop(family = 0)#, par = copulaParam)
   CopCDF     <- function(u1, u2){VineCopula::BiCopCDF(u1, u2, obj=cop)}
   C1d        <- function(u1, u2){VineCopula::BiCopHfunc(u1, u2, obj=cop)$hfunc1}
   C2d        <- function(u1, u2){VineCopula::BiCopHfunc(u1, u2, obj=cop)$hfunc2}
   C2d_rhoVC  <- function(u1, u2){u2 <- u2 + 1e-60*(u2==0)
   VineCopula::BiCopHfuncDeriv(u1, u2, obj=cop)
   }
 PortSumProb <- function(x){
  SumArg <- function(z){
```

```
    F2z <- F2(z)
    F1z <- F1(x-z)
    C2d(F1z,F2z)*f2(z)
    }
  Sum1 <- integrate(SumArg, lower = 0, upper = x)$value - alpha
  return(Sum1)
 }

# Quantile
LowRangeRisk2 <- 0
UpRangeRisk2  <- 10000*ERisk2fun()
x.quan<- uniroot(PortSumProb, lower = LowRangeRisk2, upper = UpRangeRisk2)$root
QuanPortVec[j] <- x.quan
# Portfolio Density at the quantile
    densitySumArg <- function(z){
    F2z <- F2(z)
    F1z <- F1(x.quan-z)
    VineCopula::BiCopPDF(F1z,F2z, obj=cop)*f1(x.quan-z)*f2(z)
    }
densityPortVec[j] <- integrate(densitySumArg, lower = 0, upper = x.quan)$value

#derivative wrt correlation parameter of Prob of Sum
 derCorrSumArg <- function(z){
   F2z <- F2(z)
   F1z <- F1(x.quan-z)
   C2d_rhoVC(F1z,F2z)*f2(z)
   }
  derCorrSum1 <- integrate(derCorrSumArg, lower =-1, upper = x.quan-.0001)$value
  xplot <- 0:x.quan
  derivCorrPortVec[j] <- derCorrSum1
  depsensVec <- -derivCorrPortVec/densityPortVec/100
  }
outMatA <- cbind(QuanPortVec, densityPortVec, derivCorrPortVec, depsensVec)
outMat  <- cbind(round(outMatA[,1],digits=0),round(outMatA[,2],digits=8),
                 round(outMatA[,3],digits=4),round(outMatA[,4],digits=2))
colnames(outMat) <- c("Quantile","Density", "Deriv:Rho Sum",
                      "Dependence Sensitivity (in percentages)")
rownames(outMat) <- c(paste("Rho=",rhoparam[1]), paste("Rho=",rhoparam[2]),
                      paste("Rho= ",rhoparam[3]),paste("Rho=",rhoparam[4]),
                      paste("Rho=",rhoparam[5]))
```

17.11.4 Exercise 12.4. Portfolio Distribution - Simulation Based Computations

```
#  Exercise 12.4
set.seed(202020)

# Generating Dependent Claims
```

```
rhoparam<- c(-0.9,-.3,0.000,0.3,0.9)
QuanPortVec <- matrix(0,length(rhoparam),1) ->
               densityPortVec -> derivCorrPortVec

for (j in 1:length(rhoparam)) {
  norm.cop <- copula::normalCopula(param=rhoparam[j], dim = 2)
  UCop <- copula::rCopula(nsim, norm.cop)
  X1 <- q1(UCop[,1])
  X2 <- q2(UCop[,2])
  Z1 <- qnorm(UCop[,1])
  Z2 <- qnorm(UCop[,2])
  Zmat <- cbind(Z1,Z2)
  S.port <- X1 + X2
  x.quan.sim <- quantile(S.port, probs = alpha)
  QuanPortVec[j] <- x.quan.sim
  f.xquan.sim <- density(S.port, from =x.quan.sim,
                         to=x.quan.sim )$y[1]
  densityPortVec[j] <- f.xquan.sim
  F.Q <- 1*(X1 + X2  <= x.quan.sim)
  Exp.G <- mean(F.Q)
  seExp.G <- sd(F.Q)/sqrt(nsim)

  sig <- rhoparam[j]
  Sigma <- matrix(c(1, sig, sig,  1),nrow = 2, ncol = 2)
  Sigma.inv <- solve(Sigma)

  dep.sens.partial <- function(i,j){
    partial.Sig <- matrix(0,nrow = 2, ncol = 2)
    partial.Sig[i,j] <- 1 -> partial.Sig[j,i]
    temp1 <- tr(Sigma.inv %*% partial.Sig)
    PartA <- (-0.5)*Exp.G*temp1
    ZPrime.Sig.inv <- Zmat %*% Sigma.inv
    ZPrime.Sig.inv.Z  <- rowSums(ZPrime.Sig.inv  * Zmat)
    Temp <-rowSums( (ZPrime.Sig.inv %*% partial.Sig) *
                    ZPrime.Sig.inv)
    PartB <- 0.5*mean(F.Q*Temp)
    Dep.Sen <- PartA + PartB
    return(Dep.Sen)
    }
  derivCorrPortVec[j] <- dep.sens.partial(1,2)
}
rm(UCop)
depsensVec <- -derivCorrPortVec/densityPortVec/100
outMatA <- cbind(QuanPortVec, densityPortVec,
                 derivCorrPortVec, depsensVec)
save(outMatA,
     file = "../ChapTablesData/Chap12/Example1224.RData")
```

Bibliography

Abdikerimova, Samal and Runhuan Feng (2022). "Peer-to-peer multi-risk insurance and mutual aid," *European Journal of Operational Research*, Vol. 299, pp. 735–749, URL: https://doi.org/10.1016/j.ejor.2021.09.017.

Actuarial Community (2020). *Loss Data Analytics*, URL: https://openacttexts.github.io/Loss-Data-Analytics/index.html.

Albrecher, Hansjörg, Jan Beirlant, and Jozef L Teugels (2017). *Reinsurance: Actuarial and Statistical Aspects*. John Wiley & Sons.

Albrecht, Peter and Markus Huggenberger (2017). "The fundamental theorem of mutual insurance," *Insurance: Mathematics and Economics*, Vol. 75, pp. 180–188.

Alexander, Gordon J (2013). "From Markowitz to modern risk management," in *Asset Management and International Capital Markets*. Routledge, pp. 5–15.

ANU (2018). "ANU Self-Insurance Reserve Guidelines," URL: https://services.anu.edu.au/financial-management/insurance.

——— (2020). "ANU Annual Report 2020," URL: https://www.anu.edu.au/about/strategic-planning/annual-report-2020.

——— (2022a). "ANU Insurance Policy Statement," URL: https://policies.anu.edu.au/ppl/document/ANUP_000422.

——— (2022b). "ANU Risk Management Policy Statement," URL: https://policies.anu.edu.au/ppl/document/ANUP_000462.

——— (2023). "Workers' Compensation at ANU," URL: https://services.anu.edu.au/human-resources/health-safety/workers-compensation-at-anu.

Aon Risk Solutions (2019). "Global Risk Management Survey," URL: https://www.aon.com/2019-top-global-risks-management-economics-geopolitics-brand-damage-insights/index.html.

Arrow, Kenneth J (1963). "Uncertainty and the welfare economics of medical care," *American Economic Review*, Vol. 53, pp. 941–973.

Asimit, Alexandru V, Tao Gao, Junlei Hu, and Eun-Seok Kim (2018). "Optimal risk transfer: a numerical optimization approach," *North American Actuarial Journal*, Vol. 22, pp. 341–364, URL: https://doi.org/10.1080/10920277.2017.1421472.

Assa, Hirbod (2015). "On optimal reinsurance policy with distortion risk measures and premiums," *Insurance: Mathematics and Economics*, Vol. 61, pp. 70–75, URL: https://doi.org/10.1016/j.insmatheco.2014.11.007.

Ban, Gah-Yi, Noureddine El Karoui, and Andrew EB Lim (2018). "Machine learning and portfolio optimization," *Management Science*, Vol. 64, pp. 1136–1154.

Beard, Robert, Teivo Pentikäinen, and Erkki Pesonen (1984). *Risk Theory: The Stochastic Basis of Insurance, Third Edition*. Chapman and Hall.

Beketov, Mikhail, Kevin Lehmann, and Manuel Wittke (2018). "Robo Advisors: quantitative methods inside the robots," *Journal of Asset Management*, Vol. 19, pp. 363–370.

Belles-Sampera, Jaume, Montserrat Guillén, and Miguel Santolino (2014). "Beyond value-at-risk: GlueVaR distortion risk measures," *Risk Analysis*, Vol. 34, pp. 121–134, URL: https://doi.org/10.1111/risa.12080.

Benamraoui, Abdelhafid and Aljandali Abdulkader (2020). "FinTech Innovations: A review of the recent developments and prospects," *Bancaria*, Vol. 12, pp. 85–95.

Bernard, Carole (2013). "Risk sharing and pricing in the reinsurance market," in Georges Dionne ed. *Handbook of Insurance*. Springer, pp. 603–626, URL: https://doi.org/10.1007/978-1-4614-0155-1_21.

Bernard, Carole, Fangda Liu, and Steven Vanduffel (2020). "Optimal insurance in the presence of multiple policyholders," *Journal of Economic Behavior & Organization*, Vol. 180, pp. 638–656.

Best, Michael J and Robert R Grauer (1991). "On the sensitivity of mean-variance-efficient portfolios to changes in asset means: some analytical and computational results," *The Review of Financial Studies*, Vol. 4, pp. 315–342.

Bierlaire, Michel (2018). *Optimization: Principles and Algorithms*, Lausanne. EPFL Press, 2nd edition, URL: http://optimizationprinciplesalgorithms.com/.

Billingsley, Patrick (2013). *Convergence of Probability Measures*. John Wiley & Sons.

Blanchet, Jose, Henry Lam, Qihe Tang, and Zhongyi Yuan (2019). "Robust actuarial risk analysis," *North American Actuarial Journal*, Vol. 23, pp. 33–63.

Boonen, Tim J and Mario Ghossoub (2023). "Bowley vs. Pareto optima in reinsurance contracting," *European Journal of Operational Research*, Vol. 307, pp. 382–391.

Borch, Karl H (1962). "Equilibrium in a reinsurance market," *Econometrica: Journal of the Econometric Society*, Vol. 30, pp. 424–444.

——— (1974). "The Mathematical Theory of Insurance: an annotated selections of papers on insurance published, 1960-1972."

Boyd, Stephen P and Lieven Vandenberghe (2004). *Convex Optimization*. Cambridge University Press.

Brockett, Patrick L, Samuel H Jr Cox, and Robert C Witt (1986). "Insurance versus self-insurance: A risk management perspective," *Journal of Risk and Insurance*, Vol. 53, pp. 242–257.

Bühlmann, Hans (1970). *Mathematical Methods in Risk Theory*, Vol. 172. Springer.

Cai, Jun and Yichun Chi (2020). "Optimal reinsurance designs based on risk measures: a review," *Statistical Theory and Related Fields*, pp. 1–13, URL: https://doi.org/10.1080/24754269.2020.1758500.

Cai, Jun, Haiyan Liu, and Ruodu Wang (2017). "Pareto-optimal reinsurance arrangements under general model settings," *Insurance: Mathematics and Economics*, Vol. 77, pp. 24–37.

Cai, Jun and Ken Seng Tan (2007). "Optimal retention for a stop-loss reinsurance under the VaR and CTE risk measures," *Astin Bulletin*, Vol. 37, pp. 93–112, URL: https://doi.org/10.1017/S0515036100014756.

Cai, Jun, Ken Seng Tan, Chengguo Weng, and Yi Zhang (2008). "Optimal reinsurance under VaR and CTE risk measures," *Insurance: Mathematics and Economics*, Vol. 43, pp. 185–196, URL: https://doi.org/10.1016/j.insmatheco.2008.05.011.

Candelon, Bertrand, Christophe Hurlin, and Sessi Tokpavi (2012). "Sampling error and double shrinkage estimation of minimum variance portfolios," *Journal of Empirical Finance*, Vol. 19, pp. 511–527.

Chalabi, Yohan and Diethelm Würtz (2015). "Portfolio allocation," in Arthur Charpentier ed. *Computational Actuarial Science with R*, pp. 447–470. CRC Press Taylor & Francis Group.

Cheung, Ka Chun, Wing Fung Chong, and Ambrose Lo (2019). "Budget-constrained optimal reinsurance design under coherent risk measures," *Scandinavian Actuarial Journal*, Vol. 2019, pp. 729–751, URL: https://doi.org/10.1080/03461238.2019.1598891.

Chopra, Vijay K and William T Ziemba (1993). "The effect of errors in means, variances, and covariances on optimal portfolio choice," *Journal of Portfolio Management;*, Vol. 19, pp. 6–11.

Cortis, Dominic, Jeremy Debattista, Johann Debono, and Mark Farrell (2019). "InsurTech," in *Disrupting Finance*. Palgrave Pivot, Cham, pp. 71–84.

Cummins, J. David and Olivier Mahul (2004). "The demand for insurance with an upper limit on coverage," *Journal of Risk and Insurance*, Vol. 71, pp. 253–264, URL: https://doi.org/10.1111/j.0022-4367.2004.00088.x.

Cummins, J David and Mary A Weiss (2014). "Systemic risk and the US insurance sector," *Journal of Risk and insurance*, Vol. 81, pp. 489–528.

D'Acunto, Francesco, Nagpurnanand Prabhala, and Alberto G Rossi (2019). "The promises and pitfalls of robo-advising," *The Review of Financial Studies*, Vol. 32, pp. 1983–2020.

Daw, RH (1979). "Smallpox and the double decrement table: a piece of actuarial prehistory," *Journal of the Institute of Actuaries*, Vol. 106, pp. 299–318.

Daykin, Chris D, Teivo Pentikäinen, and Martti Pesonen (1994). *Practical Risk Theory for Actuaries*. CRC Press.

De Finetti, Bruno (1940). "Il problema dei pieni," *Giorn. Ist. Ital. Attuari*, Vol. 11, pp. 1–88.

De Prado, Marcos Lopez (2016). "Building diversified portfolios that outperform out of sample," *The Journal of Portfolio Management*, Vol. 42, pp. 59–69.

DeMiguel, Victor, Lorenzo Garlappi, Francisco J Nogales, and Raman Uppal (2009). "A generalized approach to portfolio optimization: Improving performance by constraining portfolio norms," *Management science*, Vol. 55, pp. 798–812.

Denuit, Michel (2020). "Investing in your own and peers' risks: The simple analytics of P2P insurance," *European Actuarial Journal*, Vol. 10, pp. 335–359.

Denuit, Michel and Jan Dhaene (2012). "Convex order and comonotonic conditional mean risk sharing," *Insurance: Mathematics and Economics*, Vol. 51, pp. 265–270.

Denuit, Michel, Jan Dhaene, Marc Goovaerts, and Rob Kaas (2006). *Actuarial Theory for Dependent Risks: Measures, Orders and Models*. John Wiley & Sons.

Denuit, Michel, Jan Dhaene, and Christian Y Robert (2022). "Risk-sharing rules and their properties, with applications to peer-to-peer insurance," *Journal of Risk and Insurance*, Vol. 89, pp. 615–667.

Denuit, Michel and Christian Y Robert (2021a). "From risk sharing to pure premium for a large number of heterogeneous losses," *Insurance: Mathematics and Economics*, Vol. 96, pp. 116–126.

——— (2021b). "Risk sharing under the dominant peer-to-peer property and casualty insurance business models," *Risk Management and Insurance Review*, Vol. 24, pp. 181–205, URL: https://doi.org/10.1111/rmir.12180.

Dhaene, Jan, Andreas Tsanakas, Emiliano A Valdez, and Steven Vanduffel (2012). "Optimal capital allocation principles," *Journal of Risk and Insurance*, Vol. 79, pp. 1–28.

Dong, Yumo, Edward W Frees, and Fei Huang (2022). "Deductible costs for bundled insurance contracts," *Available at https://ssrn.com/abstract=4020299*.

Eling, Martin and Werner Schnell (2016). "What do we know about cyber risk and cyber risk insurance?" *The Journal of Risk Finance*, Vol. 17, pp. 474–491.

Embrechts, Paul, Rüdiger Frey, and Alexander McNeil (2005). *Quantitative Risk Management*. Princeton Series in Finance, Princeton.

Embrechts, Paul, Haiyan Liu, and Ruodu Wang (2018). "Quantile-based risk sharing," *Operations Research*, Vol. 66, pp. 936–949, URL: https://doi.org/10.1287/opre.2017.1716.

Feng, Runhuan (2023). *Decentralized Insurance: Technical Foundation of Business Models*. Springer Nature.

Feng, Runhuan, Chongda Liu, and Stephen Taylor (2023). "Peer-to-peer risk sharing with an application to flood risk pooling," *Annals of Operations Research*, Vol. 321, p. 813–842, URL: https://doi.org/10.1007/s10479-022-04841-x.

Fiacco, Anthony V (1976). "Sensitivity analysis for nonlinear programming using penalty methods," *Mathematical Programming*, Vol. 10, pp. 287–311.

Fiacco, Anthony V ed. (1983). *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*, Mathematics in Science and Engineering. Elsevier.

Fiacco, Anthony V and Garth P McCormick (1990). *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. SIAM.

Frahm, Gabriel, Markus Junker, and Alexander Szimayer (2003). "Elliptical copulas: applicability and limitations," *Statistics & Probability Letters*, Vol. 63, pp. 275–286.

Frank, Maurice J (1979). "On the simultaneous associativity of F(x, y) and x+y-F(x, y)," *Aequationes Mathematicae*, Vol. 19, pp. 194–226, URL: http://eudml.org/doc/136825.

Frees, Edward W (1990). "Stochastic life contingencies with solvency considerations," *Transactions of the Society of Actuaries*, Vol. 42, pp. 91–148, URL: https://www.soa.org/globalassets/assets/library/research/transactions-of-society-of-actuaries/1990-95/1990/january/tsa90v427.pdf.

——— (2009). *Regression Modeling with Actuarial and Financial Applications*. Cambridge University Press.

——— (2014). "Frequency and severity models," in Edward W Frees, Glenn Meyers, and

Richard Derrig eds. *Predictive Modeling Applications in Actuarial Science*, Vol. 1, pp. 138–164. Cambridge University Press Cambridge.

——— (2017). "Insurance portfolio risk retention," *North American Actuarial Journal*, Vol. 21, pp. 526–551, URL: https://doi.org/10.1080/10920277.2017.1317272.

Frees, Edward W and Adam Butt (2022). "ANU Insurable Risks," URL: https://doi.org/10.25911/0SE7-N746.

Frees, Edward W and Lisa Gao (2020). "Predictive analytics and medical malpractice," *North American Actuarial Journal*, Vol. 24, pp. 211–227, DOI: http://dx.doi.org/10.1080/10920277.2019.1634597.

Frees, Edward W and Gee Lee (2015). "Rating endorsements using generalized linear models," *Variance*, Vol. 10, pp. 51–74, URL: https://www.casact.org/sites/default/files/2021-07/Rating-Endorsements-Frees-Lee.pdf.

Frees, Edward W, Gee Lee, and Lu Yang (2016). "Multivariate frequency severity regression models in insurance," *Risks*, Vol. 4, p. 4, URL: https://doi.org/10.3390/risks4010004.

——— (2024). "Data and Code to Support 'Multivariate frequency severity regression models in insurance'," URL: https://sites.google.com/a/wisc.edu/jed-frees/home.

Frees, Edward W, Glenn Meyers, and A David Cummings (2010). "Dependent multi-peril ratemaking models," *ASTIN Bulletin: The Journal of the IAA*, Vol. 40, pp. 699–726.

Frees, Edward W and Peng Shi (2024). "Insurable risk portfolios and data uncertainty."

Frees, Edward W and Emiliano A. Valdez (1998). "Understanding relationships using copulas," *North American Actuarial Journal*, Vol. 2, pp. 1–25, URL: https://doi.org/10.1080/10920277.1998.10595667.

Fu, Michael C (2008). "What you should know about simulation and derivatives," *Naval Research Logistics*, Vol. 55, pp. 723–736, URL: https://doi.org/10.1002/nav.20313.

Fu, Michael C ed. (2015a). *Handbook of Simulation Optimization*. Springer.

Fu, Michael C (2015b). "Stochastic gradient estimation," in Michael C Fu ed. *Handbook of Simulation Optimization*, pp. 105–148. Springer.

Gassmann, H. I. (2003). "Multivariate normal probabilities: implementing an old idea of Plackett's," *Journal of Computational and Graphical Statistics*, Vol. 12, pp. 731–752.

Genest, Christian and Josh Mackay (1986). "The joy of copulas: Bivariate distributions with uniform marginals," *The American Statistician*, Vol. 40, pp. 280–283, URL: https://doi.org/10.1080/00031305.1986.10475414.

Gerber, Hans and Gérard Pafumi (1998). "Utility functions: from risk theory to finance," *North American Actuarial Journal*, Vol. 2, pp. 74–91, URL: https://doi.org/10.1080/10920277.1998.10595728.

Gerber, Hans U (1979). *An Introduction to Mathematical Risk Theory*. S.S. Huebner Foundation for Insurance Education.

Givens, Geof H and Jennifer Hoeting (2013). *Computational Statistics, Second Edition*, Vol. 596. Wiley Online Library.

Glineur, François and Jean-François Walhin (2006). "de Finetti's retention problem for proportional reinsurance revisited," *Blätter der DGVFM*, Vol. 27, p. 451 462.

Gollier, Christian (2013). "The economics of optimal insurance design," in Georges Dionne ed. *Handbook of Insurance*. Springer, pp. 107–122, URL: https://doi.org/10.1007/978-1-4614-0155-1_4.

Gollier, Christian and Harris Schlesinger (1995). "Second-best insurance contract design in an incomplete market," *The Scandinavian Journal of Economics*, pp. 123–135, URL: https://doi.org/10.2307/3440833.

Gray, Roger J. and Susan M. Pitts (2012). *Risk Modelling in General Insurance: From Principles to Practice*. Cambridge University Press.

Grealish, Adam and Petter N Kolm (2021). "Robo-Advisory: From Investing Principles and Algorithms to Future Developments."

Guo, Qiheng, Daniel Bauer, and George H Zanjani (2021). "Capital allocation techniques: Review and comparison," *Variance*, Vol. 14, URL: https://variancejournal.org/article/29684-capital-allocation-techniques-review-and-comparison.

Gutterman, Sam (2017). "IAA Risk Book Chapter 17—Risk and Uncertainty," URL: https://www.actuaries.org/IAA/Documents/Publications/RiskBook/Ch17_Risk_and_Uncertainty_2017-06-06.pdf.

Haberman, Steven and Ermanno Pitacco (2018). *Actuarial Models for Disability Insurance*. Routledge.

Han, Xia, Liyuan Lin, and Ruodu Wang (2023). "Diversification quotients based on VaR and ES," *Insurance: Mathematics and Economics*, Vol. 113, pp. 185–197.

Härdle, Wolfgang (1990). *Applied Nonparametric Regression*, No. 19. Cambridge University Press.

Harrington, Scott E and Greg Niehaus (1999). *Risk Management and Insurance*. McGraw-Hill/Irwin.

Hastie, Trevor, Robert Tibshirani, and Jerome H Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Vol. 2. Springer.

He, Guangliang and Robert Litterman (2002). "The intuition behind Black-Litterman model portfolios," *SSRN*, URL: https://ssrn.com/abstract=334304.

Henrion, René (2024). "Chance-constrained programming," URL: https://www.stoprog.org/what-stochastic-programming.

Hofert, Marius, Ivan Kojadinovic, Martin Mächler, and Jun Yan (2018). *Elements of Copula Modeling with R*. Springer.

Hong, L. Jeff (2009). "Estimating quantile sensitivities," *Operations Research*, Vol. 57, pp. 118–130, URL: https://doi.org/10.1287/opre.1080.0531.

Huebner, Solomon S, Kenneth Black, and Bernard Web (1996). *Property and Liability Insurance*. Prentice Hall.

Ingersoll, Jonathan E (1987). *Theory of Financial Decision Making*, Vol. 3. Rowman & Littlefield.

Insights, Deloitte (2019). "Global Risk Management Survey, 11th edition," URL: https://www2.deloitte.com/content/dam/insights/us/articles/4222_Global-risk-management-survey/DI_global-risk-management-survey.pdf.

Insurance Information Institute (2023a). "Finite Risk Reinsurance," URL: https://www.iii.org/article/finite-risk-reinsurance.

——— (2023b). "Small Business Insurance Basics," URL: https://www.iii.org/publications/insurance-handbook/insurance-basics/small-business-insurance-basics.

Jiang, Guangxin and Michael C Fu (2015). "Technical note. On estimating quantile sensitivities via infinitesimal perturbation analysis," *Operations Research*, Vol. 63, pp. 435–441, URL: https://doi.org/10.1287/opre.2015.1356.

Jiang, Wenjun, Hanping Hong, and Jiandong Ren (2021). "Pareto-optimal reinsurance policies with maximal synergy," *Insurance: Mathematics and Economics*, Vol. 96, pp. 185–198.

Joe, Harry (2014). *Dependence Modeling with Copulas*. CRC Press.

Kim, Sujin, Raghu Pasupathy, and Shane G Henderson (2015). "A guide to sample average approximation," *Handbook of Simulation Optimization*, pp. 207–243.

Klugman, Stuart A., Harry H. Panjer, and Gordon E. Willmot (2012). *Loss Models: From Data to Decisions*. John Wiley & Sons.

Knight, Frank H (1921). *Risk, Uncertainty, and Profit*, Boston/New York. Houghton Mifflin.

Koenker, Roger (2005). *Quantile Regression*, Vol. 38. Cambridge University Press.

Landsman, Zinoviy M and Emiliano A Valdez (2003). "Tail conditional expectations for elliptical distributions," *North American Actuarial Journal*, Vol. 7, pp. 55–71.

Ledoit, Olivier and Michael Wolf (2003). "Improved estimation of the covariance matrix of stock returns with an application to portfolio selection," *Journal of Empirical Finance*, Vol. 10, pp. 603–621.

Lee, Gee Yul (2017). *Multivariate insurance loss models with applications in risk retention*. University of Wisconsin-Madison, URL: https://na02.alma.exlibrisgroup.com/view/uresolver/01UWI_MAD/openurl?u.ignore_date_coverage=true&portfolio_pid=5384265458000212 2&Force_direct=true&rfr_id=info:sid/primo.exlibrisgroup.com, [Retrieved on January 1, 2023].

——— (2023). "Multivariate insurance portfolio risk retention using the method of multipliers," *North American Actuarial Journal*, URL: https://doi.org/10.1080/10920277.2022.2161578, [Retrieved on January 1, 2023].

Lemaire, Jean (1991). "Borch's theorem: A historical survey of applications," in *Risk, Information and Insurance*. Springer, pp. 15–37, URL: https://doi.org/10.1007/978-94-009-2183-2_2.

Levy, Paul S and Stanley Lemeshow (2013). *Sampling of Populations: Methods and Applications*. John Wiley & Sons.

Liu, Guangwu and Liu Jeff Hong (2009). "Kernel estimation of quantile sensitivities," *Naval Research Logistics*, Vol. 56, pp. 511–525.

Lo, Ambrose (2017). "A unifying approach to risk-measure-based optimal reinsurance problems with practical constraints," *Scandinavian Actuarial Journal*, Vol. 2017, pp. 584–605, URL: https://doi.org/10.1080/03461238.2016.1193558.

de Lourdes Centeno, Maria and Onofre Simões (2009). "Optimal reinsurance," *RACSAM-Revista de la Real Academia de Ciencias Exactas, Fisicas y Naturales. Serie A. Matematicas*, Vol. 103, pp. 387–404, URL: https://rac.es/ficheros/doc/00787.pdf.

Luenberger, David G, Yinyu Ye et al. (2016). *Linear and Nonlinear Programming*. Springer.

Machina, Mark J (2013). "Non-Expected Utility and the Robustness of the Classical Insurance Paradigm," in *Handbook of Insurance, Second Edition*. Springer, pp. 59–106.

Mainik, Georg and Paul Embrechts (2013). "Diversification in heavy-tailed portfolios: properties and pitfalls," *Annals of Actuarial Science*, Vol. 7, pp. 26–45.

Markowitz, Harry (1952). "The utility of wealth," *Journal of Political Economy*, Vol. 60, pp. 151–158.

Mayers, David and Clifford W Smith (2000). "Organizational forms within the insurance industry: Theory and evidence," *Handbook of Insurance*, pp. 689–707.

Metropolis, Nicholas and Stanislaw Ulam (1949). "The monte carlo method," *Journal of the American Statistical Association*, Vol. 44, pp. 335–341.

Metz, Jason and Ashlee Valentine (2023). "Business Owners Policy (BOP): Coverage and Costs," URL: https://www.forbes.com/advisor/business-insurance/business-owners-policy/.

Mildenhall, Stephen A and John Major (2022). *Pricing Insurance Risk: Theory and Practice*. John Wiley & Sons.

Morningstar Report (2022). "2022 Robo-Advisor Landscape," URL: https://www.morningstar.com/lp/robo-advisor-landscape.

Mossin, Jan (1968). "Optimal multiperiod portfolio policies," *The Journal of Business*, Vol. 41, pp. 215–229.

NAIC (2020). "Captive Insurance companies," URL: https://content.naic.org/cipr_topics/topic_captive_insurance_companies.htm.

——— (2023). "Business Interruption/Businessowner's Policies (BOP)," URL: https://content.naic.org/cipr-topics/business-interruptionbusinessowners-policies-bop.

National Association of Insurance Commissioners (2023a). "Peer-to-Peer (P2P) Insurance," URL: https://content.naic.org/cipr-topics/peer-peer-p2p-insurance.

——— (2023b). "Risk Retention Groups," URL: https://content.naic.org/cipr_topics/topic_risk_retention_groups.htm.

National Association of Risk Retention (2023). "NARR," URL: https://www.riskretention.org/.

Net, ABC (2020). "Hail storm sweeps through Canberra, damaging countless cars and smashing windows," *ABC Net*, URL: https://www.abc.net.au/news/2020-01-20/11882472.

Nocedal, Jorge and Stephen Wright (2006). *Numerical Optimization*. Springer Science & Business Media.

OECD (2020). "Insurance Statistics Yearbook," URL: https://stats.oecd.org/Index.aspx?DataSetCode=INSIND.

——— (2022). *OECD Insurance Statistics 2021*, pp.182, URL: https://www.oecd-ilibrary.org/content/publication/841fa619-en, DOI: http://dx.doi.org/https://doi.org/10.1787/841fa619-en.

Osborne, Martin J (2021). "Mathematical methods for economic theory: a tutorial," *University of Toronto, Toronto*, URL: https://mjo.osborne.economics.utoronto.ca/index.php/tutorial/index/1/toc.

Panjer, Harry H, Phelim P Boyle, Samuel H Cox, D Dufresne, HU Gerber, HH Mueller, HW Pedersen, SR Pliska, M Sherris, E Shiu et al. (1998). *Financial Economics: With Applications to Investments, Insurance, and Pensions*. Actuarial Foundation Schaumburg, Ill.

Pericoli, Marcello and Massimo Sbracia (2003). "A primer on financial contagion," *Journal of Economic Surveys*, Vol. 17, pp. 571–608.

Pesonen, Martti I (1984). "Optimal reinsurances," *Scandinavian actuarial journal*, Vol. 1984, pp. 65–90, URL: https://doi.org/10.1080/03461238.1984.10413754.

Plackett, Robin L. (1954). "A reduction formula for normal multivariate integrals," *Biometrika*, Vol. 41, pp. 351–360.

Pressacco, Flavio, Paolo Serafini, and Laura Ziani (2011). "Mean variance efficient strategies in proportional reinsurance under group correlation in a gaussian framework," *European Actuarial Journal*, Vol. 1, p. 433 454.

Puccetti, Giovanni and Ruodu Wang (2015). "Extremal dependence concepts," *Statistical Science*, Vol. 30, pp. 485–517.

Rego, Margarida Lima and Joana Campos Carvalho (2020). *Insurance in Today's Sharing Economy: New Challenges Ahead or a Return to the Origins of Insurance?*, pp. 27–47. Springer International Publishing, URL: https://doi.org/10.1007/978-3-030-27386-6_2, DOI: http://dx.doi.org/10.1007/978-3-030-27386-6_2.

Robinson, Stephen M (1974). "Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear-programming algorithms," *Mathematical programming*, Vol. 7, pp. 1–16.

Rockafellar, R Tyrrell and Stanislav Uryasev (2002). "Conditional value-at-risk for general loss distributions," *Journal of Banking & Finance*, Vol. 26, pp. 1443–1471, URL: https://doi.org/10.1016/S0378-4266(02)00271-6.

Rosset, Saharon and Ji Zhu (2007). "Piecewise linear regularized solution paths," *The Annals of Statistics*, pp. 1012–1030.

Schäfer, Juliane and Korbinian Strimmer (2005). "A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics," *Statistical Applications in Genetics and Molecular Biology*, Vol. 4.

Schepsmeier, Ulf and Jakob Stöber (2012). "Web supplement: Derivatives and Fisher information of bivariate copulas,"Technical report, Tech. rep. TU München.

——— (2014). "Derivatives and Fisher information of bivariate copulas," *Statistical Papers*, Vol. 55, pp. 525–542.

Schlesinger, Harris (2013). "The theory of insurance demand," in *Handbook of Insurance*. Springer New York, pp. 167–184, URL: https://doi.org/10.1007/978-1-4614-0155-1_7.

Seal, Hilary L (1969). *Stochastic Theory of a Risk Business*. Wiley.

Serfling, Robert J. (1980). *Approximation Theorems of Mathematical Statistics*, Vol. 162. John Wiley & Sons.

Serpa, Juan Camilo and Harish Krishnan (2017). "The strategic role of business insurance," *Management Science*, Vol. 63, pp. 384–404.

Shapiro, Alexander (2013). "Sample average approximation," *Encyclopedia of Operations Research and Management Science*, Vol. 3, pp. 1350–1355.

Simon, Carl P and Lawrence Blume (1994). *Mathematics for Economists*. W.W. Norton and Company, New York.

Skipper, Harold D. and W. Jean Kwon (2007). *Risk Management and Insurance: Perspectives in a Global Economy*. Blackwell.

Sklar, M (1959). "Fonctions de repartition a N dimensions et leurs marges," *Publ. inst. statist. univ. Paris*, Vol. 8, pp. 229–231.

State of Vermont (2020). "Captive basics," URL: https://www.vermontcaptive.com/captive/.

Stout, William F (1974). "Almost Sure Convergence."

Sun, Haoze, Chengguo Weng, and Yi Zhang (2017). "Optimal multivariate quota-share reinsurance: A nonparametric mean-CVaR framework," *Insurance: Mathematics and Economics*, Vol. 72, pp. 197–214.

Swiss.Re (2013). "The essential guide to reinsurance," URL: https://www.swissre.com/Library/the-essential-guide-to-reinsurance.html, [Retrieved on June 10, 2020].

Tan, Ken Seng and Chengguo Weng (2014). "Empirical approach for optimal reinsurance design," *North American Actuarial Journal*, Vol. 18, pp. 315–342, URL: https://doi.org/10.1080/10920277.2014.888008.

Toumi, Ralf and Lauren Restell (2014). "Catastrophe Modelling and Climate Change," URL: https://assets.lloyds.com/assets/pdf-modelling-and-climate-change-cc-and-modelling-template-v6/2/pdf-modelling-and-climate-change-CC-and-modelling-template-V6.pdf.

Tse, Yiu-Kuen (2009). *Nonlife Actuarial Models: Theory, Methods and Evaluation*. Cambridge University Press.

Vajda, Stefan (1962). "Minimum variance reinsurance," *ASTIN Bulletin: The Journal of the IAA*, Vol. 2, pp. 257–260, URL: https://doi.org/10.1017/S0515036100009995.

Verlaak, Robert and Jan Beirlant (2003). "Optimal reinsurance programs: an optimal combination of several reinsurance protections on a heterogeneous insurance portfolio," *Insurance: Mathematics and Economics*, Vol. 33, pp. 381–403, URL: https://doi.org/10.1016/j.insmatheco.2003.08.002.

Wang, Ruodu and Ričardas Zitikis (2021). "An axiomatic foundation for the Expected Shortfall," *Management Science*, Vol. 67, pp. 1413–1429, URL: https://pubsonline.informs.org/doi/pdf/10.1287/mnsc.2020.3617.

Willams, C Arthur, Michael L. Smith, and Peter C. Young (1995). *Risk Management and Insurance, Seventh Edition*. McGraw-Hill.

Xiao, Yugu and Emiliano A Valdez (2015). "A Black–Litterman asset allocation model under elliptical distributions," *Quantitative Finance*, Vol. 15, pp. 509–519.

Yates, Natalie (2023). "Reciprocal Insurance Exchange," URL: https://www.usnews.com/insurance/glossary/reciprocal-insurance-exchange.

Yin, Chenyang, Romain Perchet, and François Soupé (2021). "A practical guide to robust portfolio optimization," *Quantitative Finance*, Vol. 21, pp. 911–928.