RMarkdown

This file discusses the R code and corresponding output that is used for creating (part of) the paper "Portfolio optimization under natural clustering" by Debrauwer and Gijbels. In particular, we first focus on the computation of Example 1, followed by one of the settings for the simulation study for this Example 1. Finally, we display code used for the real data application on the stock portfolio.

Useful libraries:

```
library(NMOF)
library(MASS)
library(corpcor)
library(parallel)
library(foreach)
library(doParallel)
library(copula)
library(quantmod)
library(stats)
library(tictoc)
library(latex2exp)
```

Example 1: multivariate normal distribution

We start by computing Example 1. This corresponds to Section 5.1 "Example 1: multivariate normal distribution". In this script we compute the optimal portfolios for the five settings of Table 1, and the misspecification models from the supplementary material.

Setting 1: high within-cluster relatedness, low across-cluster relatedness

We define the setup, i.e. correlation matrix and standard deviations, as detailed in Table 1 for setting 1.

```
between=0.1
group1=0.8
group2=0.7
group3=0.6
correlations=c(group1,group1,c(0.2,0.2,0.2,0.2,0.2),
               group1,rep(between,5),
               rep(between,5),
               group2,group2,rep(between,2),
               group2,rep(between,2),
               rep(between,2),
               group3)

# Number of variables (size of the matrix)
d <- 8
cor_matrix <- diag(1, d)

# Fill the lower triangular part
cor_matrix[lower.tri(cor_matrix)] <- correlations
```

```r
# Mirror the lower triangle to the upper triangle
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)

# Display the correlation matrix
print(cor_matrix)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]  1.0  0.8  0.8  0.2  0.2  0.2  0.2  0.2
## [2,]  0.8  1.0  0.8  0.1  0.1  0.1  0.1  0.1
## [3,]  0.8  0.8  1.0  0.1  0.1  0.1  0.1  0.1
## [4,]  0.2  0.1  0.1  1.0  0.7  0.7  0.1  0.1
## [5,]  0.2  0.1  0.1  0.7  1.0  0.7  0.1  0.1
## [6,]  0.2  0.1  0.1  0.7  0.7  1.0  0.1  0.1
## [7,]  0.2  0.1  0.1  0.1  0.1  0.1  1.0  0.6
## [8,]  0.2  0.1  0.1  0.1  0.1  0.1  0.6  1.0
```

```r
eigen(cor_matrix)$values
```

```
## [1] 3.0336198 2.0942368 1.4892529 0.4000000 0.3000000 0.3000000 0.2000000
## [8] 0.1828905
```

```r
# Compute standard deviations from variances
std_devs=c(2,2.5,2,2,2,2,3,3)
# Compute the diagonal matrix of standard deviations
diag_std_devs <- diag(std_devs)

# Compute the covariance matrix
cov_matrix <- diag_std_devs %*% cor_matrix %*% diag_std_devs
```

Next, for each of the Optimization methods (Single-step and Optimizations 1–5) we compute the true optimal portfolio and the corresponding risk using expected shortfall.

```r
##################
##  Single step ##
##################

minvariance=minvar(cov_matrix)
```

```
## Loading required namespace: quadprog
```

```r
singlestep=c(minvariance[1:8])
varianceSingle=t(singlestep)%*%cov_matrix%*%singlestep
ESSingle=sqrt(varianceSingle)*dnorm(qnorm(0.95))/(1-0.95)

##################
##Optimization 1##
##################

### cluster 1
d <- 3
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group1,3)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs1 <- c(std_devs[1:3])
diag_std_devs <- diag(std_devs1)
cov_matrix1 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
```

```
minvariance=minvar(cov_matrix1)
cluster1=c(minvariance[1:3])


### cluster 2
d <- 3
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group2,3)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs2 <- c(std_devs[4:6])
diag_std_devs <- diag(std_devs2)
cov_matrix2 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix2)
cluster2=c(minvariance[1:3])



### cluster 3
d <- 2
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group3,1)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs2 <- c(std_devs[7:8])
diag_std_devs <- diag(std_devs2)
cov_matrix2 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix2)
cluster3=c(minvariance[1:2])


#between clusters
varS1=cluster1[1]^2*std_devs[1]^2+cluster1[2]^2*std_devs[2]^2+cluster1[3]^2*std_devs[3]^2+
  2*cluster1[1]*cluster1[2]*correlations[1]*std_devs[1]*std_devs[2]+
  2*cluster1[1]*cluster1[3]*correlations[2]*std_devs[1]*std_devs[3]+
  2*cluster1[2]*cluster1[3]*correlations[8]*std_devs[2]*std_devs[3]

varS2=cluster2[1]^2*std_devs[4]^2+cluster2[2]^2*std_devs[5]^2+cluster2[3]^2*std_devs[6]^2+
  2*cluster2[1]*cluster2[2]*correlations[19]*std_devs[4]*std_devs[5]+
  2*cluster2[1]*cluster2[3]*correlations[20]*std_devs[4]*std_devs[6]+
  2*cluster2[2]*cluster2[3]*correlations[23]*std_devs[5]*std_devs[6]


varS3=cluster3[1]^2*std_devs[7]^2+cluster3[2]^2*std_devs[8]^2+2*cluster3[1]*cluster3[2]*correlations[le


weights=c(cluster1,cluster2)
combinations=expand.grid(c(1:3),c(4:6))
covarianceS1S2=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S2=covarianceS1S2+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}
```

```
weights=c(cluster1,cluster3)
combinations=expand.grid(c(1:3),c(7:8))
weights=c(cluster1,rep(0,3),cluster3)

covarianceS1S3=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S3=covarianceS1S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster2,cluster3)
combinations=expand.grid(c(4:6),c(7:8))
covarianceS2S3=0
weights=c(rep(0,3),cluster2,cluster3)
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS2S3=covarianceS2S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}

cov_matrixBetweenClusters=matrix(c(varS1,covarianceS1S2,covarianceS1S3,
                                   covarianceS1S2,varS2,covarianceS2S3,
                                   covarianceS1S3,covarianceS2S3,varS3),nrow = 3, byrow = TRUE)
minvariance=minvar(cov_matrixBetweenClusters)
Opt1Between=c(minvariance[1:3])
varianceOpt1=t(Opt1Between)%*%cov_matrixBetweenClusters%*%Opt1Between
ESOpt1=sqrt(varianceOpt1)*dnorm(qnorm(0.95))/(1-0.95)




##################
##Optimization 2##
##################
#Equal weights in clusters
cluster1Eq=rep(1/3,3)
cluster2Eq=rep(1/3,3)
cluster3Eq=rep(1/2,2)

varS1=cluster1Eq[1]^2*std_devs[1]^2+cluster1Eq[2]^2*std_devs[2]^2+cluster1Eq[3]^2*std_devs[3]^2+
  2*cluster1Eq[1]*cluster1Eq[2]*correlations[1]*std_devs[1]*std_devs[2]+
  2*cluster1Eq[1]*cluster1Eq[3]*correlations[2]*std_devs[1]*std_devs[3]+
  2*cluster1Eq[2]*cluster1Eq[3]*correlations[8]*std_devs[2]*std_devs[3]

varS2=cluster2Eq[1]^2*std_devs[4]^2+cluster2Eq[2]^2*std_devs[5]^2+cluster2Eq[3]^2*std_devs[6]^2+
  2*cluster2Eq[1]*cluster2Eq[2]*correlations[19]*std_devs[4]*std_devs[5]+
  2*cluster2Eq[1]*cluster2Eq[3]*correlations[20]*std_devs[4]*std_devs[6]+
  2*cluster2Eq[2]*cluster2Eq[3]*correlations[23]*std_devs[5]*std_devs[6]


varS3=cluster3Eq[1]^2*std_devs[7]^2+cluster3Eq[2]^2*std_devs[8]^2+2*cluster3Eq[1]*cluster3Eq[2]*correla
```

```r
weights=c(cluster1Eq,cluster2Eq)
combinations=expand.grid(c(1:3),c(4:6))
covarianceS1S2=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S2=covarianceS1S2+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster1Eq,cluster3Eq)
combinations=expand.grid(c(1:3),c(7:8))
weights=c(cluster1Eq,rep(0,3),cluster3Eq)

covarianceS1S3=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S3=covarianceS1S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster2Eq,cluster3Eq)
combinations=expand.grid(c(4:6),c(7:8))
covarianceS2S3=0
weights=c(rep(0,3),cluster2Eq,cluster3Eq)
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS2S3=covarianceS2S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}

cov_matrixBetweenClusters=matrix(c(varS1,covarianceS1S2,covarianceS1S3,
                                   covarianceS1S2,varS2,covarianceS2S3,
                                   covarianceS1S3,covarianceS2S3,varS3),nrow = 3, byrow = TRUE)
minvariance=minvar(cov_matrixBetweenClusters)
Opt2Between=c(minvariance[1:3])
weightsOpt2=c(cluster1*Opt2Between[1],cluster2*Opt2Between[2],cluster3*Opt2Between[3])
varianceOpt2=t(weightsOpt2)%*%cov_matrix%*%weightsOpt2
ESOpt2=sqrt(varianceOpt2)*dnorm(qnorm(0.95))/(1-0.95)





##################
##Optimization 3##
##################
between=0.1
group1=0.8
group2=0.7
group3=0.6
correlations=c(group1,group1,c(0.2,0.2,0.2,0.2,0.2),
```

```r
              group1,rep(between,5),
              rep(between,5),
              group2,group2,rep(between,2),
              group2,rep(between,2),
              rep(between,2),
              group3)

# Number of variables (size of the matrix)
d <- 8
cor_matrix <- diag(1, d)

# Fill the lower triangular part
cor_matrix[lower.tri(cor_matrix)] <- correlations

# Mirror the lower triangle to the upper triangle
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)


#representative cluster 1
toMinimize1=std_devs[1:3]
for (i in 1:3){
  sum=0
  for (j in 4:8){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize1[i]=toMinimize1[i]*sum
}
clusterRepresentative1=which.min(toMinimize1)


#representative cluster 2
toMinimize2=std_devs[4:6]
for (i in 4:6){
  sum=0
  for (j in c(1,2,3,7,8)){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize2[i-3]=toMinimize2[i-3]*sum
}
clusterRepresentative2=which.min(toMinimize2)+3


#representative cluster 3
toMinimize3=std_devs[7:8]
for (i in 7:8){
  sum=0
  for (j in c(1,2,3,4,5,6)){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize3[i-6]=toMinimize3[i-6]*sum
}
clusterRepresentative3=which.min(toMinimize3)+6
```

```r
correlationsRepresentatives=c(0.1,0.1,0.1)
cor_matrixRepresentatives <- diag(1, 3)
cor_matrixRepresentatives[lower.tri(cor_matrixRepresentatives)] <- correlationsRepresentatives
cor_matrixRepresentatives <- cor_matrixRepresentatives + t(cor_matrixRepresentatives) - diag(1, 3)
std_devsRepresentatives=c(2,2,3)
diag_std_devsRepresentatives <- diag(std_devsRepresentatives)
cov_matrixRepresentatives <- diag_std_devsRepresentatives %*% cor_matrixRepresentatives %*% diag_std_dev

minvarianceRepresentatives=minvar(cov_matrixRepresentatives)


WeigthsRepresentatives=c(minvarianceRepresentatives[1:3])
varianceRepresentatives=t(WeigthsRepresentatives)%*%cov_matrixRepresentatives%*%WeigthsRepresentatives
ESRepresentatives=sqrt(varianceRepresentatives)*dnorm(qnorm(0.95))/(1-0.95)






##################
##Optimization 4##
##################
#Between: 0.3473537 0.4789003 0.1737460 #via optimization 2

weightsOpt4=c(0.3473537*c(1/3,1/3,1/3),0.4789003*c(1/3,1/3,1/3),0.1737460*c(1/2,1/2))
varianceOpt4=t(weightsOpt4)%*%cov_matrix%*%weightsOpt4
ESOpt4=sqrt(varianceOpt4)*dnorm(qnorm(0.95))/(1-0.95)

##################
##Optimization 5##
##################
weightsOpt5=c(1/3*c(0.5,0.0,0.5), 1/3*c(1/3,1/3,1/3), 1/3*c(0.5,0.5)) #Via optimization 1
varianceOpt5=t(weightsOpt5)%*%cov_matrix%*%weightsOpt5
ESOpt5=sqrt(varianceOpt5)*dnorm(qnorm(0.95))/(1-0.95)

######
# Summary
######

# Helper function for pretty printing
show_results <- function(title, weights, risk, w_digits = 2, r_digits = 3) {
  cat("\n", title, "\n")
  print(round(weights, digits = w_digits))
  print(round(risk, digits = r_digits))
}

# Single step
show_results("Single-step: optimal weight and risk", singlestep, ESSingle)
```

```
##
```

```
##  Single-step: optimal weight and risk
## [1] 0.00 0.00 0.36 0.16 0.16 0.16 0.09 0.09
##       [,1]
## [1,] 2.719
```

```
# Double step
show_results("Optimization 1: optimal weight and risk",
             c(cluster1*Opt1Between[1], cluster2*Opt1Between[2], cluster3*Opt1Between[3]),
             ESOpt1)
```

```
##
##  Optimization 1: optimal weight and risk
## [1] 0.19 0.00 0.19 0.15 0.15 0.15 0.08 0.08
##       [,1]
## [1,] 2.759
```

```
show_results("Optimization 2: optimal weight and risk", weightsOpt2, ESOpt2)
```

```
##
##  Optimization 2: optimal weight and risk
## [1] 0.17 0.00 0.17 0.16 0.16 0.16 0.09 0.09
##       [,1]
## [1,] 2.763
```

```
show_results("Optimization 3: optimal weight and risk", WeigthsRepresentatives, ESRepresentatives)
```

```
##
##  Optimization 3: optimal weight and risk
## [1] 0.42 0.42 0.16
##       [,1]
## [1,] 2.876
```

```
show_results("Optimization 4: optimal weight and risk", weightsOpt4, ESOpt4, r_digits = 2)
```

```
##
##  Optimization 4: optimal weight and risk
## [1] 0.12 0.12 0.12 0.16 0.16 0.16 0.09 0.09
##      [,1]
## [1,] 2.79
```

```
show_results("Optimization 5: optimal weight and risk", weightsOpt5, ESOpt5, r_digits = 2)
```

```
##
##  Optimization 5: optimal weight and risk
## [1] 0.17 0.00 0.17 0.11 0.11 0.11 0.17 0.17
##      [,1]
## [1,] 2.93
```

Setting 2: low within-cluster relatedness, low across-cluster relatedness

We define the setup, i.e. correlation matrix and standard deviations, as detailed in Table 1 for setting 2.

```
between=0.1
group1=0.3
group2=0.3
group3=0.2
correlations=c(group1,group1,c(0.2,0.2,0.2,0.2,0.2),
               group1,rep(between,5),
```

```r
              rep(between,5),
              group2,group2,rep(between,2),
              group2,rep(between,2),
              rep(between,2),
              group3)

# Number of variables (size of the matrix)
d <- 8
cor_matrix <- diag(1, d)

# Fill the lower triangular part
cor_matrix[lower.tri(cor_matrix)] <- correlations

# Mirror the lower triangle to the upper triangle
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)

# Display the correlation matrix
print(cor_matrix)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]  1.0  0.3  0.3  0.2  0.2  0.2  0.2  0.2
## [2,]  0.3  1.0  0.3  0.1  0.1  0.1  0.1  0.1
## [3,]  0.3  0.3  1.0  0.1  0.1  0.1  0.1  0.1
## [4,]  0.2  0.1  0.1  1.0  0.3  0.3  0.1  0.1
## [5,]  0.2  0.1  0.1  0.3  1.0  0.3  0.1  0.1
## [6,]  0.2  0.1  0.1  0.3  0.3  1.0  0.1  0.1
## [7,]  0.2  0.1  0.1  0.1  0.1  0.1  1.0  0.2
## [8,]  0.2  0.1  0.1  0.1  0.1  0.1  0.2  1.0
```

```r
eigen(cor_matrix)$values
```

```
## [1] 2.1814629 1.2276503 1.0385046 0.8000000 0.7000000 0.7000000 0.7000000
## [8] 0.6523822
```

```r
# Compute standard deviations from variances
std_devs=c(2,2.5,2,2,2,2,3,3)
# Compute the diagonal matrix of standard deviations
diag_std_devs <- diag(std_devs)

# Compute the covariance matrix
cov_matrix <- diag_std_devs %*% cor_matrix %*% diag_std_devs
```

For each of the Optimization methods (Single-step and Optimizations 1–5) we compute the true optimal portfolio and the corresponding risk using expected shortfall.

```r
##################
##single step   ##
##################

minvariance=minvar(cov_matrix)
singlestep=c(minvariance[1:8])
varianceSingle=t(singlestep)%*%cov_matrix%*%singlestep
ESSingle=sqrt(varianceSingle)*dnorm(qnorm(0.95))/(1-0.95)


##################
```

```r
##Optimization 1##
##################
### cluster 1
d <- 3
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group1,3)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs1 <- c(std_devs[1:3])
diag_std_devs <- diag(std_devs1)
cov_matrix1 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix1)
cluster1=c(minvariance[1:3])


### cluster 2
d <- 3
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group2,3)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs2 <- c(std_devs[4:6])
diag_std_devs <- diag(std_devs2)
cov_matrix2 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix2)
cluster2=c(minvariance[1:3])



### cluster 3
d <- 2
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group3,1)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs2 <- c(std_devs[7:8])
diag_std_devs <- diag(std_devs2)
cov_matrix2 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix2)
cluster3=c(minvariance[1:2])


#between clusters
varS1=cluster1[1]^2*std_devs[1]^2+cluster1[2]^2*std_devs[2]^2+cluster1[3]^2*std_devs[3]^2+
  2*cluster1[1]*cluster1[2]*correlations[1]*std_devs[1]*std_devs[2]+
  2*cluster1[1]*cluster1[3]*correlations[2]*std_devs[1]*std_devs[3]+
  2*cluster1[2]*cluster1[3]*correlations[8]*std_devs[2]*std_devs[3]

varS2=cluster2[1]^2*std_devs[4]^2+cluster2[2]^2*std_devs[5]^2+cluster2[3]^2*std_devs[6]^2+
  2*cluster2[1]*cluster2[2]*correlations[19]*std_devs[4]*std_devs[5]+
  2*cluster2[1]*cluster2[3]*correlations[20]*std_devs[4]*std_devs[6]+
  2*cluster2[2]*cluster2[3]*correlations[23]*std_devs[5]*std_devs[6]


varS3=cluster3[1]^2*std_devs[7]^2+cluster3[2]^2*std_devs[8]^2+2*cluster3[1]*cluster3[2]*correlations[le
```

```r
weights=c(cluster1,cluster2)
combinations=expand.grid(c(1:3),c(4:6))
covarianceS1S2=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S2=covarianceS1S2+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster1,cluster3)
combinations=expand.grid(c(1:3),c(7:8))
weights=c(cluster1,rep(0,3),cluster3)

covarianceS1S3=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S3=covarianceS1S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster2,cluster3)
combinations=expand.grid(c(4:6),c(7:8))
covarianceS2S3=0
weights=c(rep(0,3),cluster2,cluster3)
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS2S3=covarianceS2S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}

cov_matrixBetweenClusters=matrix(c(varS1,covarianceS1S2,covarianceS1S3,
                                   covarianceS1S2,varS2,covarianceS2S3,
                                   covarianceS1S3,covarianceS2S3,varS3),nrow = 3, byrow = TRUE)
minvariance=minvar(cov_matrixBetweenClusters)
Opt1Between=c(minvariance[1:3])
varianceOpt1=t(Opt1Between)%*%cov_matrixBetweenClusters%*%Opt1Between
ESOpt1=sqrt(varianceOpt1)*dnorm(qnorm(0.95))/(1-0.95)





##################
##Optimization 2##
##################
#Equal weights in clusters
cluster1Eq=rep(1/3,3)
cluster2Eq=rep(1/3,3)
cluster3Eq=rep(1/2,2)

varS1=cluster1Eq[1]^2*std_devs[1]^2+cluster1Eq[2]^2*std_devs[2]^2+cluster1Eq[3]^2*std_devs[3]^2+
  2*cluster1Eq[1]*cluster1Eq[2]*correlations[1]*std_devs[1]*std_devs[2]+
```

```
  2*cluster1Eq[1]*cluster1Eq[3]*correlations[2]*std_devs[1]*std_devs[3]+
  2*cluster1Eq[2]*cluster1Eq[3]*correlations[8]*std_devs[2]*std_devs[3]

varS2=cluster2Eq[1]^2*std_devs[4]^2+cluster2Eq[2]^2*std_devs[5]^2+cluster2Eq[3]^2*std_devs[6]^2+
  2*cluster2Eq[1]*cluster2Eq[2]*correlations[19]*std_devs[4]*std_devs[5]+
  2*cluster2Eq[1]*cluster2Eq[3]*correlations[20]*std_devs[4]*std_devs[6]+
  2*cluster2Eq[2]*cluster2Eq[3]*correlations[23]*std_devs[5]*std_devs[6]


varS3=cluster3Eq[1]^2*std_devs[7]^2+cluster3Eq[2]^2*std_devs[8]^2+2*cluster3Eq[1]*cluster3Eq[2]*correla


weights=c(cluster1Eq,cluster2Eq)
combinations=expand.grid(c(1:3),c(4:6))
covarianceS1S2=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S2=covarianceS1S2+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster1Eq,cluster3Eq)
combinations=expand.grid(c(1:3),c(7:8))
weights=c(cluster1Eq,rep(0,3),cluster3Eq)

covarianceS1S3=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S3=covarianceS1S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster2Eq,cluster3Eq)
combinations=expand.grid(c(4:6),c(7:8))
covarianceS2S3=0
weights=c(rep(0,3),cluster2Eq,cluster3Eq)
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS2S3=covarianceS2S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}

cov_matrixBetweenClusters=matrix(c(varS1,covarianceS1S2,covarianceS1S3,
                                   covarianceS1S2,varS2,covarianceS2S3,
                                   covarianceS1S3,covarianceS2S3,varS3),nrow = 3, byrow = TRUE)
minvariance=minvar(cov_matrixBetweenClusters)
Opt2Between=c(minvariance[1:3])
weightsOpt2=c(cluster1*Opt2Between[1],cluster2*Opt2Between[2],cluster3*Opt2Between[3])
varianceOpt2=t(weightsOpt2)%*%cov_matrix%*%weightsOpt2
ESOpt2=sqrt(varianceOpt2)*dnorm(qnorm(0.95))/(1-0.95)
```

```r
##################
##Optimization 3##
##################
between=0.1
group1=0.3
group2=0.3
group3=0.2
correlations=c(group1,group1,c(0.2,0.2,0.2,0.2,0.2),
               group1,rep(between,5),
               rep(between,5),
               group2,group2,rep(between,2),
               group2,rep(between,2),
               rep(between,2),
               group3)

# Number of variables (size of the matrix)
d <- 8
cor_matrix <- diag(1, d)

# Fill the lower triangular part
cor_matrix[lower.tri(cor_matrix)] <- correlations

# Mirror the lower triangle to the upper triangle
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)


#representative cluster 1
toMinimize1=std_devs[1:3]
for (i in 1:3){
  sum=0
  for (j in 4:8){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize1[i]=toMinimize1[i]*sum
}
clusterRepresentative1=which.min(toMinimize1)


#representative cluster 2
toMinimize2=std_devs[4:6]
for (i in 4:6){
  sum=0
  for (j in c(1,2,3,7,8)){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize2[i-3]=toMinimize2[i-3]*sum
}
clusterRepresentative2=which.min(toMinimize2)+3
```

```r
#representative cluster 3
toMinimize3=std_devs[7:8]
for (i in 7:8){
  sum=0
  for (j in c(1,2,3,4,5,6)){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize3[i-6]=toMinimize3[i-6]*sum
}
clusterRepresentative3=which.min(toMinimize3)+6





correlationsRepresentatives=c(0.1,0.1,0.1)
cor_matrixRepresentatives <- diag(1, 3)
cor_matrixRepresentatives[lower.tri(cor_matrixRepresentatives)] <- correlationsRepresentatives
cor_matrixRepresentatives <- cor_matrixRepresentatives + t(cor_matrixRepresentatives) - diag(1, 3)
std_devsRepresentatives=c(2,2,3)
diag_std_devsRepresentatives <- diag(std_devsRepresentatives)
cov_matrixRepresentatives <- diag_std_devsRepresentatives %*% cor_matrixRepresentatives %*% diag_std_de

minvarianceRepresentatives=minvar(cov_matrixRepresentatives)


WeigthsRepresentatives=c(minvarianceRepresentatives[1:3])
varianceRepresentatives=t(WeigthsRepresentatives)%*%cov_matrixRepresentatives%*%WeigthsRepresentatives
ESRepresentatives=sqrt(varianceRepresentatives)*dnorm(qnorm(0.95))/(1-0.95)





##################
##Optimization 4##
##################
#Between:  #via optimization 2

weightsOpt4=c(Opt2Between[1]*c(1/3,1/3,1/3),Opt2Between[2]*c(1/3,1/3,1/3),Opt2Between[3]*c(1/2,1/2))
varianceOpt4=t(weightsOpt4)%*%cov_matrix%*%weightsOpt4
ESOpt4=sqrt(varianceOpt4)*dnorm(qnorm(0.95))/(1-0.95)

##################
##Optimization 5##
##################

weightsOpt5=c(1/3*c(cluster1), 1/3*c(cluster2), 1/3*c(cluster3)) #Via optimization 1
varianceOpt5=t(weightsOpt5)%*%cov_matrix%*%weightsOpt5
ESOpt5=sqrt(varianceOpt5)*dnorm(qnorm(0.95))/(1-0.95)
```

```r
######
##Summary
# Single step
show_results("Single-step: optimal weight and risk", singlestep, ESSingle)
```

```
##
##  Single-step: optimal weight and risk
## [1] 0.10 0.08 0.20 0.16 0.16 0.16 0.07 0.07
##       [,1]
## [1,] 2.364
```

```r
# Double step
show_results("Optimization 1: optimal weight and risk",
             c(cluster1*Opt1Between[1], cluster2*Opt1Between[2], cluster3*Opt1Between[3]),
             ESOpt1)
```

```
##
##  Optimization 1: optimal weight and risk
## [1] 0.16 0.07 0.16 0.16 0.16 0.16 0.06 0.06
##       [,1]
## [1,] 2.377
```

```r
show_results("Optimization 2: optimal weight and risk", weightsOpt2, ESOpt2)
```

```
##
##  Optimization 2: optimal weight and risk
## [1] 0.15 0.07 0.15 0.16 0.16 0.16 0.07 0.07
##       [,1]
## [1,] 2.377
```

```r
show_results("Optimization 3: optimal weight and risk", WeigthsRepresentatives, ESRepresentatives)
```

```
##
##  Optimization 3: optimal weight and risk
## [1] 0.42 0.42 0.16
##       [,1]
## [1,] 2.876
```

```r
show_results("Optimization 4: optimal weight and risk", weightsOpt4, ESOpt4, r_digits = 2)
```

```
##
##  Optimization 4: optimal weight and risk
## [1] 0.13 0.13 0.13 0.16 0.16 0.16 0.07 0.07
##      [,1]
## [1,] 2.39
```

```r
show_results("Optimization 5: optimal weight and risk", weightsOpt5, ESOpt5, r_digits = 2)
```

```
##
##  Optimization 5: optimal weight and risk
## [1] 0.14 0.06 0.14 0.11 0.11 0.11 0.17 0.17
##      [,1]
## [1,] 2.58
```

Setting 3: high within-cluster relatedness, high across-cluster relatedness

We define the setup, i.e. correlation matrix and standard deviations, as detailed in Table 1 for setting 3.

```r
between=0.1
group1=0.8
group2=0.7
group3=0.6
correlations=c(group1,group1,c(0.5,0.5,0.4,0.4,0.3),
               group1,rep(between,5),
               rep(between,5),
               group2,group2,rep(between,2),
               group2,rep(between,2),
               rep(between,2),
               group3)

# Number of variables (size of the matrix)
d <- 8
cor_matrix <- diag(1, d)

# Fill the lower triangular part
cor_matrix[lower.tri(cor_matrix)] <- correlations

# Mirror the lower triangle to the upper triangle
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)

# Display the correlation matrix
print(cor_matrix)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]  1.0  0.8  0.8  0.5  0.5  0.4  0.4  0.3
## [2,]  0.8  1.0  0.8  0.1  0.1  0.1  0.1  0.1
## [3,]  0.8  0.8  1.0  0.1  0.1  0.1  0.1  0.1
## [4,]  0.5  0.1  0.1  1.0  0.7  0.7  0.1  0.1
## [5,]  0.5  0.1  0.1  0.7  1.0  0.7  0.1  0.1
## [6,]  0.4  0.1  0.1  0.7  0.7  1.0  0.1  0.1
## [7,]  0.4  0.1  0.1  0.1  0.1  0.1  1.0  0.6
## [8,]  0.3  0.1  0.1  0.1  0.1  0.1  0.6  1.0
```

```r
eigen(cor_matrix)$values
```

```
## [1] 3.372987458 1.922137885 1.473799731 0.409575121 0.313758193 0.300000000
## [7] 0.200000000 0.007741612
```

```r
# Compute standard deviations from variances
std_devs=c(2,2.5,2,2,2,2,3,3)
# Compute the diagonal matrix of standard deviations
diag_std_devs <- diag(std_devs)

# Compute the covariance matrix
cov_matrix <- diag_std_devs %*% cor_matrix %*% diag_std_devs
```

For each of the Optimization methods (Single-step and Optimizations 1–5) we compute the true optimal portfolio and the corresponding risk using expected shortfall.

```r
##################
#single step    ##
##################
minvariance=minvar(cov_matrix)
singlestep=c(minvariance[1:8])
```

```
varianceSingle=t(singlestep)%*%cov_matrix%*%singlestep
ESSingle=sqrt(varianceSingle)*dnorm(qnorm(0.95))/(1-0.95)


##################
##Optimization 1##
##################
### cluster 1
d <- 3
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group1,3)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs1 <- c(std_devs[1:3])
diag_std_devs <- diag(std_devs1)
cov_matrix1 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix1)
cluster1=c(minvariance[1:3])


### cluster 2
d <- 3
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group2,3)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs2 <- c(std_devs[4:6])
diag_std_devs <- diag(std_devs2)
cov_matrix2 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix2)
cluster2=c(minvariance[1:3])



### cluster 3
d <- 2
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group3,1)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs2 <- c(std_devs[7:8])
diag_std_devs <- diag(std_devs2)
cov_matrix2 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix2)
cluster3=c(minvariance[1:2])


#between clusters
varS1=cluster1[1]^2*std_devs[1]^2+cluster1[2]^2*std_devs[2]^2+cluster1[3]^2*std_devs[3]^2+
  2*cluster1[1]*cluster1[2]*correlations[1]*std_devs[1]*std_devs[2]+
  2*cluster1[1]*cluster1[3]*correlations[2]*std_devs[1]*std_devs[3]+
  2*cluster1[2]*cluster1[3]*correlations[8]*std_devs[2]*std_devs[3]

varS2=cluster2[1]^2*std_devs[4]^2+cluster2[2]^2*std_devs[5]^2+cluster2[3]^2*std_devs[6]^2+
  2*cluster2[1]*cluster2[2]*correlations[19]*std_devs[4]*std_devs[5]+
  2*cluster2[1]*cluster2[3]*correlations[20]*std_devs[4]*std_devs[6]+
```

```r
  2*cluster2[2]*cluster2[3]*correlations[23]*std_devs[5]*std_devs[6]


varS3=cluster3[1]^2*std_devs[7]^2+cluster3[2]^2*std_devs[8]^2+2*cluster3[1]*cluster3[2]*correlations[len


weights=c(cluster1,cluster2)
combinations=expand.grid(c(1:3),c(4:6))
covarianceS1S2=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S2=covarianceS1S2+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster1,cluster3)
combinations=expand.grid(c(1:3),c(7:8))
weights=c(cluster1,rep(0,3),cluster3)

covarianceS1S3=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S3=covarianceS1S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster2,cluster3)
combinations=expand.grid(c(4:6),c(7:8))
covarianceS2S3=0
weights=c(rep(0,3),cluster2,cluster3)
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS2S3=covarianceS2S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}

cov_matrixBetweenClusters=matrix(c(varS1,covarianceS1S2,covarianceS1S3,
                                   covarianceS1S2,varS2,covarianceS2S3,
                                   covarianceS1S3,covarianceS2S3,varS3),nrow = 3, byrow = TRUE)
minvariance=minvar(cov_matrixBetweenClusters)
Opt1Between=c(minvariance[1:3])
varianceOpt1=t(Opt1Between)%*%cov_matrixBetweenClusters%*%Opt1Between
ESOpt1=sqrt(varianceOpt1)*dnorm(qnorm(0.95))/(1-0.95)




##################
##Optimization 2##
##################
#Equal weights in clusters
```

```
cluster1Eq=rep(1/3,3)
cluster2Eq=rep(1/3,3)
cluster3Eq=rep(1/2,2)

varS1=cluster1Eq[1]^2*std_devs[1]^2+cluster1Eq[2]^2*std_devs[2]^2+cluster1Eq[3]^2*std_devs[3]^2+
  2*cluster1Eq[1]*cluster1Eq[2]*correlations[1]*std_devs[1]*std_devs[2]+
  2*cluster1Eq[1]*cluster1Eq[3]*correlations[2]*std_devs[1]*std_devs[3]+
  2*cluster1Eq[2]*cluster1Eq[3]*correlations[8]*std_devs[2]*std_devs[3]

varS2=cluster2Eq[1]^2*std_devs[4]^2+cluster2Eq[2]^2*std_devs[5]^2+cluster2Eq[3]^2*std_devs[6]^2+
  2*cluster2Eq[1]*cluster2Eq[2]*correlations[19]*std_devs[4]*std_devs[5]+
  2*cluster2Eq[1]*cluster2Eq[3]*correlations[20]*std_devs[4]*std_devs[6]+
  2*cluster2Eq[2]*cluster2Eq[3]*correlations[23]*std_devs[5]*std_devs[6]


varS3=cluster3Eq[1]^2*std_devs[7]^2+cluster3Eq[2]^2*std_devs[8]^2+2*cluster3Eq[1]*cluster3Eq[2]*correla


weights=c(cluster1Eq,cluster2Eq)
combinations=expand.grid(c(1:3),c(4:6))
covarianceS1S2=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S2=covarianceS1S2+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster1Eq,cluster3Eq)
combinations=expand.grid(c(1:3),c(7:8))
weights=c(cluster1Eq,rep(0,3),cluster3Eq)

covarianceS1S3=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S3=covarianceS1S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster2Eq,cluster3Eq)
combinations=expand.grid(c(4:6),c(7:8))
covarianceS2S3=0
weights=c(rep(0,3),cluster2Eq,cluster3Eq)
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS2S3=covarianceS2S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}

cov_matrixBetweenClusters=matrix(c(varS1,covarianceS1S2,covarianceS1S3,
                                   covarianceS1S2,varS2,covarianceS2S3,
                                   covarianceS1S3,covarianceS2S3,varS3),nrow = 3, byrow = TRUE)
minvariance=minvar(cov_matrixBetweenClusters)
```

```r
Opt2Between=c(minvariance[1:3])
weightsOpt2=c(cluster1*Opt2Between[1],cluster2*Opt2Between[2],cluster3*Opt2Between[3])
varianceOpt2=t(weightsOpt2)%*%cov_matrix%*%weightsOpt2
ESOpt2=sqrt(varianceOpt2)*dnorm(qnorm(0.95))/(1-0.95)



##################
##Optimization 3##
##################
between=0.1
group1=0.8
group2=0.7
group3=0.6
correlations=c(group1,group1,c(0.5,0.5,0.4,0.4,0.3),
               group1,rep(between,5),
               rep(between,5),
               group2,group2,rep(between,2),
               group2,rep(between,2),
               rep(between,2),
               group3)

# Number of variables (size of the matrix)
d <- 8
cor_matrix <- diag(1, d)

# Fill the lower triangular part
cor_matrix[lower.tri(cor_matrix)] <- correlations

# Mirror the lower triangle to the upper triangle
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)

#representative cluster 1
toMinimize1=std_devs[1:3]
for (i in 1:3){
  sum=0
  for (j in 4:8){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize1[i]=toMinimize1[i]*sum
}
clusterRepresentative1=which.min(toMinimize1)


#representative cluster 2
toMinimize2=std_devs[4:6]
for (i in 4:6){
  sum=0
  for (j in c(1,2,3,7,8)){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize2[i-3]=toMinimize2[i-3]*sum
}
```

```
clusterRepresentative2=which.min(toMinimize2)+3


#representative cluster 3
toMinimize3=std_devs[7:8]
for (i in 7:8){
  sum=0
  for (j in c(1,2,3,4,5,6)){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize3[i-6]=toMinimize3[i-6]*sum
}
clusterRepresentative3=which.min(toMinimize3)+6





correlationsRepresentatives=c(0.1,0.1,0.1)
cor_matrixRepresentatives <- diag(1, 3)
cor_matrixRepresentatives[lower.tri(cor_matrixRepresentatives)] <- correlationsRepresentatives
cor_matrixRepresentatives <- cor_matrixRepresentatives + t(cor_matrixRepresentatives) - diag(1, 3)
std_devsRepresentatives=c(2,2,3)
diag_std_devsRepresentatives <- diag(std_devsRepresentatives)
cov_matrixRepresentatives <- diag_std_devsRepresentatives %*% cor_matrixRepresentatives %*% diag_std_de

minvarianceRepresentatives=minvar(cov_matrixRepresentatives)


WeigthsRepresentatives=c(minvarianceRepresentatives[1:3])
varianceRepresentatives=t(WeigthsRepresentatives)%*%cov_matrixRepresentatives%*%WeigthsRepresentatives
ESRepresentatives=sqrt(varianceRepresentatives)*dnorm(qnorm(0.95))/(1-0.95)










##################
##Optimization 4##
##################
weightsOpt4=c(Opt2Between[1]*c(1/3,1/3,1/3),Opt2Between[2]*c(1/3,1/3,1/3),Opt2Between[3]*c(1/2,1/2))
varianceOpt4=t(weightsOpt4)%*%cov_matrix%*%weightsOpt4
ESOpt4=sqrt(varianceOpt4)*dnorm(qnorm(0.95))/(1-0.95)

##################
##Optimization 5##
```

```r
##################
weightsOpt5=c(1/3*c(cluster1), 1/3*c(cluster2), 1/3*c(cluster3)) #Via optimization 1
varianceOpt5=t(weightsOpt5)%*%cov_matrix%*%weightsOpt5
ESOpt5=sqrt(varianceOpt5)*dnorm(qnorm(0.95))/(1-0.95)



######
##Summary
####

# Single step
show_results("Single-step: optimal weight and risk", singlestep, ESSingle)
```

```
##
##  Single-step: optimal weight and risk
## [1] 0.00 0.00 0.36 0.16 0.16 0.16 0.09 0.09
##       [,1]
## [1,] 2.719
```

```r
# Double step
show_results("Optimization 1: optimal weight and risk",
             c(cluster1*Opt1Between[1], cluster2*Opt1Between[2], cluster3*Opt1Between[3]),
             ESOpt1)
```

```
##
##  Optimization 1: optimal weight and risk
## [1] 0.17 0.00 0.17 0.16 0.16 0.16 0.09 0.09
##       [,1]
## [1,] 2.936
```

```r
show_results("Optimization 2: optimal weight and risk", weightsOpt2, ESOpt2)
```

```
##
##  Optimization 2: optimal weight and risk
## [1] 0.16 0.00 0.16 0.16 0.16 0.16 0.09 0.09
##       [,1]
## [1,] 2.937
```

```r
show_results("Optimization 3: optimal weight and risk", WeigthsRepresentatives, ESRepresentatives)
```

```
##
##  Optimization 3: optimal weight and risk
## [1] 0.42 0.42 0.16
##       [,1]
## [1,] 2.876
```

```r
show_results("Optimization 4: optimal weight and risk", weightsOpt4, ESOpt4, r_digits = 2)
```

```
##
##  Optimization 4: optimal weight and risk
## [1] 0.11 0.11 0.11 0.16 0.16 0.16 0.09 0.09
##      [,1]
## [1,] 2.91
```

```r
show_results("Optimization 5: optimal weight and risk", weightsOpt5, ESOpt5, r_digits = 2)
```

```
##
##  Optimization 5: optimal weight and risk
## [1] 0.17 0.00 0.17 0.11 0.11 0.11 0.17 0.17
##      [,1]
## [1,] 3.09
```

Setting 4: low within-cluster relatedness, high across-cluster relatedness

We define the setup, i.e. correlation matrix and standard deviations, as detailed in Table 1 for setting 4.

```
between=0.1
group1=0.3
group2=0.3
group3=0.2
correlations=c(group1,group1,c(0.5,0.5,0.4,0.4,0.3),
               group1,rep(between,5),
               rep(between,5),
               group2,group2,rep(between,2),
               group2,rep(between,2),
               rep(between,2),
               group3)

# Number of variables (size of the matrix)
d <- 8
cor_matrix <- diag(1, d)

# Fill the lower triangular part
cor_matrix[lower.tri(cor_matrix)] <- correlations

# Mirror the lower triangle to the upper triangle
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)

# Display the correlation matrix
print(cor_matrix)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]  1.0  0.3  0.3  0.5  0.5  0.4  0.4  0.3
## [2,]  0.3  1.0  0.3  0.1  0.1  0.1  0.1  0.1
## [3,]  0.3  0.3  1.0  0.1  0.1  0.1  0.1  0.1
## [4,]  0.5  0.1  0.1  1.0  0.3  0.3  0.1  0.1
## [5,]  0.5  0.1  0.1  0.3  1.0  0.3  0.1  0.1
## [6,]  0.4  0.1  0.1  0.3  0.3  1.0  0.1  0.1
## [7,]  0.4  0.1  0.1  0.1  0.1  0.1  1.0  0.2
## [8,]  0.3  0.1  0.1  0.1  0.1  0.1  0.2  1.0
```

```
eigen(cor_matrix)$values
```

```
## [1] 2.5682540 1.1824212 1.0430159 0.8064643 0.7094933 0.7000000 0.7000000
## [8] 0.2903512
```

```
# Compute standard deviations from variances
std_devs=c(2,2.5,2,2,2,2,3,3)
# Compute the diagonal matrix of standard deviations
diag_std_devs <- diag(std_devs)

# Compute the covariance matrix
```

```
cov_matrix <- diag_std_devs %*% cor_matrix %*% diag_std_devs
```

For each of the Optimization methods (Single-step and Optimizations 1–5) we compute the true optimal portfolio and the corresponding risk using expected shortfall.

```
##################
##Single step   ##
##################
minvariance=minvar(cov_matrix)
singlestep=c(minvariance[1:8])
varianceSingle=t(singlestep)%*%cov_matrix%*%singlestep
ESSingle=sqrt(varianceSingle)*dnorm(qnorm(0.95))/(1-0.95)


##################
##Optimization 1##
##################
### cluster 1
d <- 3
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group1,3)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs1 <- c(std_devs[1:3])
diag_std_devs <- diag(std_devs1)
cov_matrix1 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix1)
cluster1=c(minvariance[1:3])


### cluster 2
d <- 3
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group2,3)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs2 <- c(std_devs[4:6])
diag_std_devs <- diag(std_devs2)
cov_matrix2 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix2)
cluster2=c(minvariance[1:3])



### cluster 3
d <- 2
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group3,1)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs2 <- c(std_devs[7:8])
diag_std_devs <- diag(std_devs2)
cov_matrix2 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix2)
cluster3=c(minvariance[1:2])
```

```r
#between clusters
varS1=cluster1[1]^2*std_devs[1]^2+cluster1[2]^2*std_devs[2]^2+cluster1[3]^2*std_devs[3]^2+
  2*cluster1[1]*cluster1[2]*correlations[1]*std_devs[1]*std_devs[2]+
  2*cluster1[1]*cluster1[3]*correlations[2]*std_devs[1]*std_devs[3]+
  2*cluster1[2]*cluster1[3]*correlations[8]*std_devs[2]*std_devs[3]

varS2=cluster2[1]^2*std_devs[4]^2+cluster2[2]^2*std_devs[5]^2+cluster2[3]^2*std_devs[6]^2+
  2*cluster2[1]*cluster2[2]*correlations[19]*std_devs[4]*std_devs[5]+
  2*cluster2[1]*cluster2[3]*correlations[20]*std_devs[4]*std_devs[6]+
  2*cluster2[2]*cluster2[3]*correlations[23]*std_devs[5]*std_devs[6]


varS3=cluster3[1]^2*std_devs[7]^2+cluster3[2]^2*std_devs[8]^2+2*cluster3[1]*cluster3[2]*correlations[len


weights=c(cluster1,cluster2)
combinations=expand.grid(c(1:3),c(4:6))
covarianceS1S2=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S2=covarianceS1S2+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster1,cluster3)
combinations=expand.grid(c(1:3),c(7:8))
weights=c(cluster1,rep(0,3),cluster3)

covarianceS1S3=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S3=covarianceS1S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster2,cluster3)
combinations=expand.grid(c(4:6),c(7:8))
covarianceS2S3=0
weights=c(rep(0,3),cluster2,cluster3)
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS2S3=covarianceS2S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}

cov_matrixBetweenClusters=matrix(c(varS1,covarianceS1S2,covarianceS1S3,
                                   covarianceS1S2,varS2,covarianceS2S3,
                                   covarianceS1S3,covarianceS2S3,varS3),nrow = 3, byrow = TRUE)
minvariance=minvar(cov_matrixBetweenClusters)
Opt1Between=c(minvariance[1:3])
varianceOpt1=t(Opt1Between)%*%cov_matrixBetweenClusters%*%Opt1Between
ESOpt1=sqrt(varianceOpt1)*dnorm(qnorm(0.95))/(1-0.95)
```

```r
##################
##Optimization 2##
##################
#Equal weights in clusters
cluster1Eq=rep(1/3,3)
cluster2Eq=rep(1/3,3)
cluster3Eq=rep(1/2,2)

varS1=cluster1Eq[1]^2*std_devs[1]^2+cluster1Eq[2]^2*std_devs[2]^2+cluster1Eq[3]^2*std_devs[3]^2+
  2*cluster1Eq[1]*cluster1Eq[2]*correlations[1]*std_devs[1]*std_devs[2]+
  2*cluster1Eq[1]*cluster1Eq[3]*correlations[2]*std_devs[1]*std_devs[3]+
  2*cluster1Eq[2]*cluster1Eq[3]*correlations[8]*std_devs[2]*std_devs[3]

varS2=cluster2Eq[1]^2*std_devs[4]^2+cluster2Eq[2]^2*std_devs[5]^2+cluster2Eq[3]^2*std_devs[6]^2+
  2*cluster2Eq[1]*cluster2Eq[2]*correlations[19]*std_devs[4]*std_devs[5]+
  2*cluster2Eq[1]*cluster2Eq[3]*correlations[20]*std_devs[4]*std_devs[6]+
  2*cluster2Eq[2]*cluster2Eq[3]*correlations[23]*std_devs[5]*std_devs[6]


varS3=cluster3Eq[1]^2*std_devs[7]^2+cluster3Eq[2]^2*std_devs[8]^2+2*cluster3Eq[1]*cluster3Eq[2]*correla


weights=c(cluster1Eq,cluster2Eq)
combinations=expand.grid(c(1:3),c(4:6))
covarianceS1S2=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S2=covarianceS1S2+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster1Eq,cluster3Eq)
combinations=expand.grid(c(1:3),c(7:8))
weights=c(cluster1Eq,rep(0,3),cluster3Eq)

covarianceS1S3=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S3=covarianceS1S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster2Eq,cluster3Eq)
combinations=expand.grid(c(4:6),c(7:8))
covarianceS2S3=0
weights=c(rep(0,3),cluster2Eq,cluster3Eq)
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
```

```
  covarianceS2S3=covarianceS2S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}

cov_matrixBetweenClusters=matrix(c(varS1,covarianceS1S2,covarianceS1S3,
                                   covarianceS1S2,varS2,covarianceS2S3,
                                   covarianceS1S3,covarianceS2S3,varS3),nrow = 3, byrow = TRUE)
minvariance=minvar(cov_matrixBetweenClusters)
Opt2Between=c(minvariance[1:3])
weightsOpt2=c(cluster1*Opt2Between[1],cluster2*Opt2Between[2],cluster3*Opt2Between[3])
varianceOpt2=t(weightsOpt2)%*%cov_matrix%*%weightsOpt2
ESOpt2=sqrt(varianceOpt2)*dnorm(qnorm(0.95))/(1-0.95)




##################
##Optimization 3##
##################
between=0.1
group1=0.3
group2=0.3
group3=0.2
correlations=c(group1,group1,c(0.5,0.5,0.4,0.4,0.3),
               group1,rep(between,5),
               rep(between,5),
               group2,group2,rep(between,2),
               group2,rep(between,2),
               rep(between,2),
               group3)

# Number of variables (size of the matrix)
d <- 8
cor_matrix <- diag(1, d)

# Fill the lower triangular part
cor_matrix[lower.tri(cor_matrix)] <- correlations

# Mirror the lower triangle to the upper triangle
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)

#representative cluster 1
toMinimize1=std_devs[1:3]
for (i in 1:3){
  sum=0
  for (j in 4:8){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize1[i]=toMinimize1[i]*sum
}
clusterRepresentative1=which.min(toMinimize1)


#representative cluster 2
```

```
toMinimize2=std_devs[4:6]
for (i in 4:6){
  sum=0
  for (j in c(1,2,3,7,8)){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize2[i-3]=toMinimize2[i-3]*sum
}
clusterRepresentative2=which.min(toMinimize2)+3


#representative cluster 3
toMinimize3=std_devs[7:8]
for (i in 7:8){
  sum=0
  for (j in c(1,2,3,4,5,6)){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize3[i-6]=toMinimize3[i-6]*sum
}
clusterRepresentative3=which.min(toMinimize3)+6





correlationsRepresentatives=c(0.1,0.1,0.1)
cor_matrixRepresentatives <- diag(1, 3)
cor_matrixRepresentatives[lower.tri(cor_matrixRepresentatives)] <- correlationsRepresentatives
cor_matrixRepresentatives <- cor_matrixRepresentatives + t(cor_matrixRepresentatives) - diag(1, 3)
std_devsRepresentatives=c(2,2,3)
diag_std_devsRepresentatives <- diag(std_devsRepresentatives)
cov_matrixRepresentatives <- diag_std_devsRepresentatives %*% cor_matrixRepresentatives %*% diag_std_dev

minvarianceRepresentatives=minvar(cov_matrixRepresentatives)


WeigthsRepresentatives=c(minvarianceRepresentatives[1:3])
varianceRepresentatives=t(WeigthsRepresentatives)%*%cov_matrixRepresentatives%*%WeigthsRepresentatives
ESRepresentatives=sqrt(varianceRepresentatives)*dnorm(qnorm(0.95))/(1-0.95)





##################
##Optimization 4##
##################
weightsOpt4=c(Opt2Between[1]*c(1/3,1/3,1/3),Opt2Between[2]*c(1/3,1/3,1/3),Opt2Between[3]*c(1/2,1/2))
varianceOpt4=t(weightsOpt4)%*%cov_matrix%*%weightsOpt4
ESOpt4=sqrt(varianceOpt4)*dnorm(qnorm(0.95))/(1-0.95)

##################
##Optimization 5##
```

```r
##################
weightsOpt5=c(1/3*c(cluster1), 1/3*c(cluster2), 1/3*c(cluster3)) #Via optimization 1
varianceOpt5=t(weightsOpt5)%*%cov_matrix%*%weightsOpt5
ESOpt5=sqrt(varianceOpt5)*dnorm(qnorm(0.95))/(1-0.95)


######
##Summary
####

# Single step
show_results("Single-step: optimal weight and risk", singlestep, ESSingle)
```

```
##
##  Single-step: optimal weight and risk
## [1] 0.00 0.10 0.22 0.17 0.17 0.17 0.08 0.08
##       [,1]
## [1,] 2.393
```

```r
# Double step
show_results("Optimization 1: optimal weight and risk",
             c(cluster1*Opt1Between[1], cluster2*Opt1Between[2], cluster3*Opt1Between[3]),
             ESOpt1)
```

```
##
##  Optimization 1: optimal weight and risk
## [1] 0.14 0.06 0.14 0.17 0.17 0.17 0.07 0.07
##       [,1]
## [1,] 2.545
```

```r
show_results("Optimization 2: optimal weight and risk", weightsOpt2, ESOpt2)
```

```
##
##  Optimization 2: optimal weight and risk
## [1] 0.14 0.06 0.14 0.17 0.17 0.17 0.07 0.07
##       [,1]
## [1,] 2.545
```

```r
show_results("Optimization 3: optimal weight and risk", WeigthsRepresentatives, ESRepresentatives)
```

```
##
##  Optimization 3: optimal weight and risk
## [1] 0.42 0.42 0.16
##       [,1]
## [1,] 2.876
```

```r
show_results("Optimization 4: optimal weight and risk", weightsOpt4, ESOpt4, r_digits = 2)
```

```
##
##  Optimization 4: optimal weight and risk
## [1] 0.11 0.11 0.11 0.17 0.17 0.17 0.07 0.07
##      [,1]
## [1,] 2.52
```

```r
show_results("Optimization 5: optimal weight and risk", weightsOpt5, ESOpt5, r_digits = 2)
```

```
##
```

```
##  Optimization 5: optimal weight and risk
## [1] 0.14 0.06 0.14 0.11 0.11 0.11 0.17 0.17
##      [,1]
## [1,] 2.72
```

Setting 5

We define the setup, i.e. correlation matrix and standard deviations, as detailed in Table 1 for setting 5.

```
between=0.1
group1=0.8
group2=0.7
group3=0.6
correlations=c(group1,group1,c(0.2,0.2,0.2,0.2,0.2),
               group1,rep(between,5),
               rep(between,5),
               group2,group2,rep(between,2),
               group2,rep(between,2),
               rep(between,2),
               group3)



# Number of variables (size of the matrix)
d <- 8
cor_matrix <- diag(1, d)

# Fill the lower triangular part
cor_matrix[lower.tri(cor_matrix)] <- correlations

# Mirror the lower triangle to the upper triangle
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)

# Display the correlation matrix
print(cor_matrix)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]  1.0  0.8  0.8  0.2  0.2  0.2  0.2  0.2
## [2,]  0.8  1.0  0.8  0.1  0.1  0.1  0.1  0.1
## [3,]  0.8  0.8  1.0  0.1  0.1  0.1  0.1  0.1
## [4,]  0.2  0.1  0.1  1.0  0.7  0.7  0.1  0.1
## [5,]  0.2  0.1  0.1  0.7  1.0  0.7  0.1  0.1
## [6,]  0.2  0.1  0.1  0.7  0.7  1.0  0.1  0.1
## [7,]  0.2  0.1  0.1  0.1  0.1  0.1  1.0  0.6
## [8,]  0.2  0.1  0.1  0.1  0.1  0.1  0.6  1.0
```

```
eigen(cor_matrix)$values
```

```
## [1] 3.0336198 2.0942368 1.4892529 0.4000000 0.3000000 0.3000000 0.2000000
## [8] 0.1828905
```

```
# Compute standard deviations from variances
std_devs=c(2,2.5,2,5,2,2,3,3)
# Compute the diagonal matrix of standard deviations
diag_std_devs <- diag(std_devs)
```

```
# Compute the covariance matrix
cov_matrix <- diag_std_devs %*% cor_matrix %*% diag_std_devs
```

For each of the Optimization methods (Single-step and Optimizations 1–5) we compute the true optimal portfolio and the corresponding risk using expected shortfall.

```
##################
##single step   ##
##################
minvariance=minvar(cov_matrix)
singlestep=c(minvariance[1:8])
varianceSingle=t(singlestep)%*%cov_matrix%*%singlestep
ESSingle=sqrt(varianceSingle)*dnorm(qnorm(0.95))/(1-0.95)


##################
##Optimization 1##
##################
### cluster 1
d <- 3
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group1,3)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs1 <- c(std_devs[1:3])
diag_std_devs <- diag(std_devs1)
cov_matrix1 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix1)
cluster1=c(minvariance[1:3])


### cluster 2
d <- 3
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group2,3)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs2 <- c(std_devs[4:6])
diag_std_devs <- diag(std_devs2)
cov_matrix2 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix2)
cluster2=c(minvariance[1:3])



### cluster 3
d <- 2
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group3,1)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs2 <- c(std_devs[7:8])
diag_std_devs <- diag(std_devs2)
cov_matrix2 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix2)
cluster3=c(minvariance[1:2])
```

```
#between clusters
varS1=cluster1[1]^2*std_devs[1]^2+cluster1[2]^2*std_devs[2]^2+cluster1[3]^2*std_devs[3]^2+
  2*cluster1[1]*cluster1[2]*correlations[1]*std_devs[1]*std_devs[2]+
  2*cluster1[1]*cluster1[3]*correlations[2]*std_devs[1]*std_devs[3]+
  2*cluster1[2]*cluster1[3]*correlations[8]*std_devs[2]*std_devs[3]

varS2=cluster2[1]^2*std_devs[4]^2+cluster2[2]^2*std_devs[5]^2+cluster2[3]^2*std_devs[6]^2+
  2*cluster2[1]*cluster2[2]*correlations[19]*std_devs[4]*std_devs[5]+
  2*cluster2[1]*cluster2[3]*correlations[20]*std_devs[4]*std_devs[6]+
  2*cluster2[2]*cluster2[3]*correlations[23]*std_devs[5]*std_devs[6]


varS3=cluster3[1]^2*std_devs[7]^2+cluster3[2]^2*std_devs[8]^2+2*cluster3[1]*cluster3[2]*correlations[le


weights=c(cluster1,cluster2)
combinations=expand.grid(c(1:3),c(4:6))
covarianceS1S2=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S2=covarianceS1S2+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster1,cluster3)
combinations=expand.grid(c(1:3),c(7:8))
weights=c(cluster1,rep(0,3),cluster3)

covarianceS1S3=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S3=covarianceS1S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster2,cluster3)
combinations=expand.grid(c(4:6),c(7:8))
covarianceS2S3=0
weights=c(rep(0,3),cluster2,cluster3)
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS2S3=covarianceS2S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}

cov_matrixBetweenClusters=matrix(c(varS1,covarianceS1S2,covarianceS1S3,
                                   covarianceS1S2,varS2,covarianceS2S3,
                                   covarianceS1S3,covarianceS2S3,varS3),nrow = 3, byrow = TRUE)
minvariance=minvar(cov_matrixBetweenClusters)
Opt1Between=c(minvariance[1:3])
varianceOpt1=t(Opt1Between)%*%cov_matrixBetweenClusters%*%Opt1Between
ESOpt1=sqrt(varianceOpt1)*dnorm(qnorm(0.95))/(1-0.95)
```

```
##################
##Optimization 2##
##################
#Equal weights in clusters
cluster1Eq=rep(1/3,3)
cluster2Eq=rep(1/3,3)
cluster3Eq=rep(1/2,2)

varS1=cluster1Eq[1]^2*std_devs[1]^2+cluster1Eq[2]^2*std_devs[2]^2+cluster1Eq[3]^2*std_devs[3]^2+
  2*cluster1Eq[1]*cluster1Eq[2]*correlations[1]*std_devs[1]*std_devs[2]+
  2*cluster1Eq[1]*cluster1Eq[3]*correlations[2]*std_devs[1]*std_devs[3]+
  2*cluster1Eq[2]*cluster1Eq[3]*correlations[8]*std_devs[2]*std_devs[3]

varS2=cluster2Eq[1]^2*std_devs[4]^2+cluster2Eq[2]^2*std_devs[5]^2+cluster2Eq[3]^2*std_devs[6]^2+
  2*cluster2Eq[1]*cluster2Eq[2]*correlations[19]*std_devs[4]*std_devs[5]+
  2*cluster2Eq[1]*cluster2Eq[3]*correlations[20]*std_devs[4]*std_devs[6]+
  2*cluster2Eq[2]*cluster2Eq[3]*correlations[23]*std_devs[5]*std_devs[6]


varS3=cluster3Eq[1]^2*std_devs[7]^2+cluster3Eq[2]^2*std_devs[8]^2+2*cluster3Eq[1]*cluster3Eq[2]*correla


weights=c(cluster1Eq,cluster2Eq)
combinations=expand.grid(c(1:3),c(4:6))
covarianceS1S2=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S2=covarianceS1S2+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster1Eq,cluster3Eq)
combinations=expand.grid(c(1:3),c(7:8))
weights=c(cluster1Eq,rep(0,3),cluster3Eq)

covarianceS1S3=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S3=covarianceS1S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster2Eq,cluster3Eq)
combinations=expand.grid(c(4:6),c(7:8))
covarianceS2S3=0
weights=c(rep(0,3),cluster2Eq,cluster3Eq)
for (i in 1:dim(combinations)[1]){
```

```r
  row=as.numeric(combinations[i,])
  covarianceS2S3=covarianceS2S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}

cov_matrixBetweenClusters=matrix(c(varS1,covarianceS1S2,covarianceS1S3,
                                   covarianceS1S2,varS2,covarianceS2S3,
                                   covarianceS1S3,covarianceS2S3,varS3),nrow = 3, byrow = TRUE)
minvariance=minvar(cov_matrixBetweenClusters)
Opt2Between=c(minvariance[1:3])
weightsOpt2=c(cluster1*Opt2Between[1],cluster2*Opt2Between[2],cluster3*Opt2Between[3])
varianceOpt2=t(weightsOpt2)%*%cov_matrix%*%weightsOpt2
ESOpt2=sqrt(varianceOpt2)*dnorm(qnorm(0.95))/(1-0.95)




##################
##Optimization 3##
##################
between=0.1
group1=0.8
group2=0.7
group3=0.6
correlations=c(group1,group1,c(0.2,0.2,0.2,0.2,0.2),
               group1,rep(between,5),
               rep(between,5),
               group2,group2,rep(between,2),
               group2,rep(between,2),
               rep(between,2),
               group3)



# Number of variables (size of the matrix)
d <- 8
cor_matrix <- diag(1, d)

# Fill the lower triangular part
cor_matrix[lower.tri(cor_matrix)] <- correlations

# Mirror the lower triangle to the upper triangle
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)

#representative cluster 1
toMinimize1=std_devs[1:3]
for (i in 1:3){
  sum=0
  for (j in 4:8){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize1[i]=toMinimize1[i]*sum
}
clusterRepresentative1=which.min(toMinimize1)
```

```
#representative cluster 2
toMinimize2=std_devs[4:6]
for (i in 4:6){
  sum=0
  for (j in c(1,2,3,7,8)){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize2[i-3]=toMinimize2[i-3]*sum
}
clusterRepresentative2=which.min(toMinimize2)+3


#representative cluster 3
toMinimize3=std_devs[7:8]
for (i in 7:8){
  sum=0
  for (j in c(1,2,3,4,5,6)){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize3[i-6]=toMinimize3[i-6]*sum
}
clusterRepresentative3=which.min(toMinimize3)+6





correlationsRepresentatives=c(0.1,0.1,0.1)
cor_matrixRepresentatives <- diag(1, 3)
cor_matrixRepresentatives[lower.tri(cor_matrixRepresentatives)] <- correlationsRepresentatives
cor_matrixRepresentatives <- cor_matrixRepresentatives + t(cor_matrixRepresentatives) - diag(1, 3)
std_devsRepresentatives=c(2,2,3)
diag_std_devsRepresentatives <- diag(std_devsRepresentatives)
cov_matrixRepresentatives <- diag_std_devsRepresentatives %*% cor_matrixRepresentatives %*% diag_std_de

minvarianceRepresentatives=minvar(cov_matrixRepresentatives)


WeigthsRepresentatives=c(minvarianceRepresentatives[1:3])
varianceRepresentatives=t(WeigthsRepresentatives)%*%cov_matrixRepresentatives%*%WeigthsRepresentatives
ESRepresentatives=sqrt(varianceRepresentatives)*dnorm(qnorm(0.95))/(1-0.95)



##################
##Optimization 4##
##################
weightsOpt4=c(Opt2Between[1]*c(1/3,1/3,1/3),Opt2Between[2]*c(1/3,1/3,1/3),Opt2Between[3]*c(1/2,1/2))
varianceOpt4=t(weightsOpt4)%*%cov_matrix%*%weightsOpt4
ESOpt4=sqrt(varianceOpt4)*dnorm(qnorm(0.95))/(1-0.95)

##################
```

```r
##Optimization 5##
##################
weightsOpt5=c(1/3*c(cluster1), 1/3*c(cluster2), 1/3*c(cluster3)) #Via optimization 1
varianceOpt5=t(weightsOpt5)%*%cov_matrix%*%weightsOpt5
ESOpt5=sqrt(varianceOpt5)*dnorm(qnorm(0.95))/(1-0.95)


######
##Summary
####

# Single step
show_results("Single-step: optimal weight and risk", singlestep, ESSingle)
```

```
##
##  Single-step: optimal weight and risk
## [1] 0.01 0.00 0.37 0.00 0.22 0.22 0.09 0.09
##       [,1]
## [1,] 2.752
```

```r
# Double step
show_results("Optimization 1: optimal weight and risk",
             c(cluster1*Opt1Between[1], cluster2*Opt1Between[2], cluster3*Opt1Between[3]),
             ESOpt1)
```

```
##
##  Optimization 1: optimal weight and risk
## [1] 0.20 0.00 0.20 0.00 0.22 0.22 0.08 0.08
##      [,1]
## [1,] 2.79
```

```r
show_results("Optimization 2: optimal weight and risk", weightsOpt2, ESOpt2)
```

```
##
##  Optimization 2: optimal weight and risk
## [1] 0.25 0.00 0.25 0.00 0.12 0.12 0.13 0.13
##       [,1]
## [1,] 2.934
```

```r
show_results("Optimization 3: optimal weight and risk", WeigthsRepresentatives, ESRepresentatives)
```

```
##
##  Optimization 3: optimal weight and risk
## [1] 0.42 0.42 0.16
##       [,1]
## [1,] 2.876
```

```r
show_results("Optimization 4: optimal weight and risk", weightsOpt4, ESOpt4, r_digits = 2)
```

```
##
##  Optimization 4: optimal weight and risk
## [1] 0.17 0.17 0.17 0.08 0.08 0.08 0.13 0.13
##      [,1]
## [1,] 3.24
```

```r
show_results("Optimization 5: optimal weight and risk", weightsOpt5, ESOpt5, r_digits = 2)
```

```
##
##  Optimization 5: optimal weight and risk
## [1] 0.17 0.00 0.17 0.00 0.17 0.17 0.17 0.17
##      [,1]
## [1,] 2.95
```

Setting 1 with misspecification

Here we use the wrong clusering $(X_1, X_5, X_7); (X_2, X_4, X_6); (X_3, X_8)$. We define the setup, i.e. correlation matrix and standard deviations, as detailed in the Supplementary Material.

```
between=0.1
group1=0.8
group2=0.7
group3=0.6
correlations=c(group1,group1,c(0.2,0.2,0.2,0.2,0.2),
               group1,rep(between,5),
               rep(between,5),
               group2,group2,rep(between,2),
               group2,rep(between,2),
               rep(between,2),
               group3)

# Number of variables (size of the matrix)
d <- 8
cor_matrix <- diag(1, d)

# Fill the lower triangular part
cor_matrix[lower.tri(cor_matrix)] <- correlations

# Mirror the lower triangle to the upper triangle
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
cor_matrixBig=cor_matrix
# Display the correlation matrix
print(cor_matrix)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]  1.0  0.8  0.8  0.2  0.2  0.2  0.2  0.2
## [2,]  0.8  1.0  0.8  0.1  0.1  0.1  0.1  0.1
## [3,]  0.8  0.8  1.0  0.1  0.1  0.1  0.1  0.1
## [4,]  0.2  0.1  0.1  1.0  0.7  0.7  0.1  0.1
## [5,]  0.2  0.1  0.1  0.7  1.0  0.7  0.1  0.1
## [6,]  0.2  0.1  0.1  0.7  0.7  1.0  0.1  0.1
## [7,]  0.2  0.1  0.1  0.1  0.1  0.1  1.0  0.6
## [8,]  0.2  0.1  0.1  0.1  0.1  0.1  0.6  1.0
```

```
eigen(cor_matrix)$values
```

```
## [1] 3.0336198 2.0942368 1.4892529 0.4000000 0.3000000 0.3000000 0.2000000
## [8] 0.1828905
```

```
# Compute standard deviations from variances
std_devs=c(2,2.5,2,2,2,2,3,3)
# Compute the diagonal matrix of standard deviations
diag_std_devs <- diag(std_devs)
```

```r
# Compute the covariance matrix
cov_matrix <- diag_std_devs %*% cor_matrix %*% diag_std_devs
```

For each of the Optimization methods (Single-step and Optimizations 1–5) we compute the true optimal portfolio and the corresponding risk using expected shortfall.

```r
##################
##Single step   ##
##################
minvariance=minvar(cov_matrix)
singlestep=c(minvariance[1:8])
varianceSingle=t(singlestep)%*%cov_matrix%*%singlestep
ESSingle=sqrt(varianceSingle)*dnorm(qnorm(0.95))/(1-0.95)


##################
##Optimization 1##
##################
### cluster 1
d <- 3
cor_matrix <- diag(1, d)
cor_matrix[upper.tri(cor_matrix)] <- c(cor_matrixBig[1,5],cor_matrixBig[1,7],cor_matrixBig[5,7])
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs1 <- c(std_devs[1],std_devs[5],std_devs[7])
diag_std_devs <- diag(std_devs1)
cov_matrix1 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix1)
cluster1=c(minvariance[1:3])


### cluster 2
d <- 3
cor_matrix <- diag(1, d)
cor_matrix[upper.tri(cor_matrix)] <- c(cor_matrixBig[2,4],cor_matrixBig[2,6],cor_matrixBig[4,6])
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs2 <- c(std_devs[2],std_devs[4],std_devs[6])
diag_std_devs <- diag(std_devs2)
cov_matrix2 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix2)
cluster2=c(minvariance[1:3])



### cluster 3
d <- 2
cor_matrix <- diag(1, d)
cor_matrix[upper.tri(cor_matrix)] <- c(cor_matrixBig[3,8])
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs2 <- c(std_devs[3],std_devs[8])
diag_std_devs <- diag(std_devs2)
cov_matrix2 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix2)
cluster3=c(minvariance[1:2])
```

```r
#between clusters
varS1=cluster1[1]^2*std_devs[1]^2+cluster1[2]^2*std_devs[5]^2+cluster1[3]^2*std_devs[7]^2+
  2*cluster1[1]*cluster1[2]*cov_matrix[1,5]+
  2*cluster1[1]*cluster1[3]*cov_matrix[1,7]+
  2*cluster1[2]*cluster1[3]*cov_matrix[5,7]

varS2=cluster2[1]^2*std_devs[2]^2+cluster2[2]^2*std_devs[4]^2+cluster2[3]^2*std_devs[6]^2+
  2*cluster2[1]*cluster2[2]*cov_matrix[2,4]+
  2*cluster2[1]*cluster2[3]*cov_matrix[2,6]+
  2*cluster2[2]*cluster2[3]*cov_matrix[4,6]


varS3=cluster3[1]^2*std_devs[3]^2+cluster3[2]^2*std_devs[8]^2+2*cluster3[1]*cluster3[2]*cov_matrix[3,8]


#Between cluster 1 and cluster 2 Sums
combinations=expand.grid(c(1,5,7),c(2,4,6))
covarianceS1S2=0
weights=c(cluster1[1],cluster2[1],0,cluster2[2],cluster1[2],cluster2[3],cluster1[3])
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S2=covarianceS1S2+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



#Between cluster 1 and cluster 3 Sums
combinations=expand.grid(c(1,5,7),c(3,8))
weights=c(cluster1[1],0,cluster3[1],0,cluster1[2],0,cluster1[3],cluster3[2])

covarianceS1S3=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S3=covarianceS1S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}


#Between cluster 2 and cluster 3 Sums
combinations=expand.grid(c(2,4,6),c(3,8))
covarianceS2S3=0
weights=c(0,cluster2[1],cluster3[1],cluster2[2],0,cluster2[3],0,cluster3[2])
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS2S3=covarianceS2S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}

cov_matrixBetweenClusters=matrix(c(varS1,covarianceS1S2,covarianceS1S3,
                                   covarianceS1S2,varS2,covarianceS2S3,
                                   covarianceS1S3,covarianceS2S3,varS3),nrow = 3, byrow = TRUE)
minvariance=minvar(cov_matrixBetweenClusters)
Opt1Between=c(minvariance[1:3])
varianceOpt1=t(Opt1Between)%*%cov_matrixBetweenClusters%*%Opt1Between
ESOpt1=sqrt(varianceOpt1)*dnorm(qnorm(0.95))/(1-0.95)
```

```r
weightsOrderedByCluster=c(cluster1*Opt1Between[1],cluster2*Opt1Between[2],cluster3*Opt1Between[3])
weightsOpt1=c(weightsOrderedByCluster[1],weightsOrderedByCluster[4],weightsOrderedByCluster[7],
              weightsOrderedByCluster[5],weightsOrderedByCluster[2],weightsOrderedByCluster[6],
              weightsOrderedByCluster[3],weightsOrderedByCluster[8])




##################
##Optimization 2##
##################
#Equal weights in clusters
cluster1Eq=rep(1/3,3)
cluster2Eq=rep(1/3,3)
cluster3Eq=rep(1/2,2)
#between clusters
varS1=cluster1Eq[1]^2*std_devs[1]^2+cluster1Eq[2]^2*std_devs[5]^2+cluster1Eq[3]^2*std_devs[7]^2+
  2*cluster1Eq[1]*cluster1Eq[2]*cov_matrix[1,5]+
  2*cluster1Eq[1]*cluster1Eq[3]*cov_matrix[1,7]+
  2*cluster1Eq[2]*cluster1Eq[3]*cov_matrix[5,7]

varS2=cluster2Eq[1]^2*std_devs[2]^2+cluster2Eq[2]^2*std_devs[4]^2+cluster2Eq[3]^2*std_devs[6]^2+
  2*cluster2Eq[1]*cluster2Eq[2]*cov_matrix[2,4]+
  2*cluster2Eq[1]*cluster2Eq[3]*cov_matrix[2,6]+
  2*cluster2Eq[2]*cluster2Eq[3]*cov_matrix[4,6]


varS3=cluster3Eq[1]^2*std_devs[3]^2+cluster3Eq[2]^2*std_devs[8]^2+2*cluster3Eq[1]*cluster3Eq[2]*cov_mat


#Between cluster 1 and cluster 2 Sums
combinations=expand.grid(c(1,5,7),c(2,4,6))
covarianceS1S2=0
weights=c(cluster1Eq[1],cluster2Eq[1],0,cluster2Eq[2],cluster1Eq[2],cluster2Eq[3],cluster1Eq[3])
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S2=covarianceS1S2+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



#Between cluster 1 and cluster 3 Sums
combinations=expand.grid(c(1,5,7),c(3,8))
weights=c(cluster1Eq[1],0,cluster3Eq[1],0,cluster1Eq[2],0,cluster1Eq[3],cluster3Eq[2])

covarianceS1S3=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S3=covarianceS1S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}
```

```r
#Between cluster 2 and cluster 3 Sums
combinations=expand.grid(c(2,4,6),c(3,8))
covarianceS2S3=0
weights=c(0,cluster2Eq[1],cluster3Eq[1],cluster2Eq[2],0,cluster2Eq[3],0,cluster3Eq[2])
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS2S3=covarianceS2S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}

cov_matrixBetweenClusters=matrix(c(varS1,covarianceS1S2,covarianceS1S3,
                                   covarianceS1S2,varS2,covarianceS2S3,
                                   covarianceS1S3,covarianceS2S3,varS3),nrow = 3, byrow = TRUE)
minvariance=minvar(cov_matrixBetweenClusters)
Opt2Between=c(minvariance[1:3])
weightsOrderedByCluster=c(cluster1*Opt2Between[1],cluster2*Opt2Between[2],cluster3*Opt2Between[3])
weightsOpt2=c(weightsOrderedByCluster[1],weightsOrderedByCluster[4],weightsOrderedByCluster[7],
              weightsOrderedByCluster[5],weightsOrderedByCluster[2],weightsOrderedByCluster[6],
              weightsOrderedByCluster[3],weightsOrderedByCluster[8])
varianceOpt2=t(weightsOpt2)%*%cov_matrix%*%weightsOpt2
ESOpt2=sqrt(varianceOpt2)*dnorm(qnorm(0.95))/(1-0.95)




##################
##Optimization 3##
##################
correlations=c(group1,group1,c(0.2,0.2,0.2,0.2,0.2),
               group1,rep(between,5),
               rep(between,5),
               group2,group2,rep(between,2),
               group2,rep(between,2),
               rep(between,2),
               group3)

# Number of variables (size of the matrix)
d <- 8
cor_matrix <- diag(1, d)

# Fill the lower triangular part
cor_matrix[lower.tri(cor_matrix)] <- correlations

# Mirror the lower triangle to the upper triangle
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)

#representative cluster 1
toMinimize1=std_devs[c(1,5,7)]
cluster1Numbers=c(1,5,7)
for (i in 1:3){
  m=cluster1Numbers[i]
  sum=0
  for (j in c(2,3,4,6,8)){
```

```r
    sum=sum+abs(asin(cor_matrix[m,j])*2/pi)
  }
  toMinimize1[i]=toMinimize1[i]*sum
}
clusterRepresentative1=cluster1Numbers[which.min(toMinimize1)]


#representative cluster 2
toMinimize2=std_devs[c(2,4,6)]
cluster2Numbers=c(2,4,6)
for (i in 1:3){
  m=cluster2Numbers[i]
  sum=0
  for (j in c(1,3,5,7,8)){
    sum=sum+abs(asin(cor_matrix[m,j])*2/pi)
  }
  toMinimize2[i]=toMinimize2[i]*sum
}
clusterRepresentative2=cluster2Numbers[which.min(toMinimize2)]



#representative cluster 3
toMinimize3=std_devs[c(3,8)]
cluster3Numbers=c(3,8)
for (i in 1:2){
  m=cluster3Numbers[i]
  sum=0
  for (j in c(1,2,4,5,6,7)){
    sum=sum+abs(asin(cor_matrix[m,j])*2/pi)
  }
  toMinimize3[i]=toMinimize3[i]*sum
}
clusterRepresentative3=cluster3Numbers[which.min(toMinimize3)]


#representatives: 7,4,8


correlationsRepresentatives=c(0.1,0.6,0.1)
cor_matrixRepresentatives <- diag(1, 3)
cor_matrixRepresentatives[lower.tri(cor_matrixRepresentatives)] <- correlationsRepresentatives
cor_matrixRepresentatives <- cor_matrixRepresentatives + t(cor_matrixRepresentatives) - diag(1, 3)
std_devsRepresentatives=c(3,2,3)
diag_std_devsRepresentatives <- diag(std_devsRepresentatives)
cov_matrixRepresentatives <- diag_std_devsRepresentatives %*% cor_matrixRepresentatives %*% diag_std_dev

minvarianceRepresentatives=minvar(cov_matrixRepresentatives)


WeigthsRepresentatives=c(minvarianceRepresentatives[1:3])
varianceRepresentatives=t(WeigthsRepresentatives)%*%cov_matrixRepresentatives%*%WeigthsRepresentatives
ESRepresentatives=sqrt(varianceRepresentatives)*dnorm(qnorm(0.95))/(1-0.95)
```

```
WeightsRepresentativesOpt3=c(0,0,0,WeigthsRepresentatives[2],0,0,WeigthsRepresentatives[1],WeigthsRepres



##################
##Optimization 4##
##################
weightsOrderedByCluster=c(c(1/3,1/3,1/3)*Opt2Between[1],c(1/3,1/3,1/3)*Opt2Between[2],c(1/2,1/2)*Opt2Bet
weightsOpt4=c(weightsOrderedByCluster[1],weightsOrderedByCluster[4],weightsOrderedByCluster[7],
              weightsOrderedByCluster[5],weightsOrderedByCluster[2],weightsOrderedByCluster[6],
              weightsOrderedByCluster[3],weightsOrderedByCluster[8])
varianceOpt4=t(weightsOpt4)%*%cov_matrix%*%weightsOpt4
ESOpt4=sqrt(varianceOpt4)*dnorm(qnorm(0.95))/(1-0.95)

round(weightsOpt4,digits=2)
```

```
## [1] 0.10 0.16 0.11 0.16 0.10 0.16 0.10 0.11
```

```
##################
##Optimization 5##
##################
weightsOrderedByClusterOpt5=c(1/3*c(cluster1), 1/3*c(cluster2), 1/3*c(cluster3)) #Via optimization 1
weightsOpt5=c(weightsOrderedByClusterOpt5[1],weightsOrderedByClusterOpt5[4],weightsOrderedByClusterOpt5
              weightsOrderedByClusterOpt5[5],weightsOrderedByClusterOpt5[2],weightsOrderedByClusterOpt5
              weightsOrderedByClusterOpt5[3],weightsOrderedByClusterOpt5[8])

varianceOpt5=t(weightsOpt5)%*%cov_matrix%*%weightsOpt5
ESOpt5=sqrt(varianceOpt5)*dnorm(qnorm(0.95))/(1-0.95)


######
##Summary
####single step
round(singlestep,digits=2)
```

```
## [1] 0.00 0.00 0.36 0.16 0.16 0.16 0.09 0.09
```

```
round(ESSingle,digits = 3)
```

```
##       [,1]
## [1,] 2.719
```

```
#double step: Optimization 1
round(weightsOpt1,digits=2)
```

```
## [1] 0.16 0.11 0.19 0.11 0.17 0.11 0.06 0.08
```

```
round(ESOpt1,digits=3)
```

```
##       [,1]
## [1,] 2.821
```

```
#double step: Optimization 2
round(weightsOpt2,digits=2)
```

```
## [1] 0.12 0.16 0.15 0.16 0.13 0.16 0.05 0.06
```

```r
round(ESOpt2,digits=3)
```

```
##        [,1]
## [1,] 2.841
```

```r
#double step: Optimization 3
round(WeigthsRepresentatives,digits = 2)
```

```
## [1] 0.17 0.66 0.17
```

```r
round(ESRepresentatives,digits=3)
```

```
##        [,1]
## [1,] 3.479
```

Misspecification 2

We define the setup, i.e. correlation matrix and standard deviations, as detailed in the Supplementary Material.

```r
between=0.5
group1=0.5
group2=0.5
group3=0.5
correlations=c(group1,group1,rep(between,5),
               group1,rep(between,5),
               rep(between,5),
               group2,group2,rep(between,2),
               group2,rep(between,2),
               rep(between,2),
               group3)

# Number of variables (size of the matrix)
d <- 8
cor_matrix <- diag(1, d)

# Fill the lower triangular part
cor_matrix[lower.tri(cor_matrix)] <- correlations

# Mirror the lower triangle to the upper triangle
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)

# Display the correlation matrix
print(cor_matrix)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]  1.0  0.5  0.5  0.5  0.5  0.5  0.5  0.5
## [2,]  0.5  1.0  0.5  0.5  0.5  0.5  0.5  0.5
## [3,]  0.5  0.5  1.0  0.5  0.5  0.5  0.5  0.5
## [4,]  0.5  0.5  0.5  1.0  0.5  0.5  0.5  0.5
## [5,]  0.5  0.5  0.5  0.5  1.0  0.5  0.5  0.5
## [6,]  0.5  0.5  0.5  0.5  0.5  1.0  0.5  0.5
## [7,]  0.5  0.5  0.5  0.5  0.5  0.5  1.0  0.5
## [8,]  0.5  0.5  0.5  0.5  0.5  0.5  0.5  1.0
```

```r
eigen(cor_matrix)$values
```

```
## [1] 4.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
```

```r
# Compute standard deviations from variances
std_devs=c(2,2.5,2,2,2,2,3,3)
# Compute the diagonal matrix of standard deviations
diag_std_devs <- diag(std_devs)

# Compute the covariance matrix
cov_matrix <- diag_std_devs %*% cor_matrix %*% diag_std_devs
```

For each of the Optimization methods (Single-step and Optimizations 1–5) we compute the true optimal portfolio and the corresponding risk using expected shortfall.

```r
##################
##Single step   ##
##################

minvariance=minvar(cov_matrix)
singlestep=c(minvariance[1:8])
varianceSingle=t(singlestep)%*%cov_matrix%*%singlestep
ESSingle=sqrt(varianceSingle)*dnorm(qnorm(0.95))/(1-0.95)


##################
##Optimization 1##
##################
### cluster 1
d <- 3
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group1,3)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs1 <- c(std_devs[1:3])
diag_std_devs <- diag(std_devs1)
cov_matrix1 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix1)
cluster1=c(minvariance[1:3])


### cluster 2
d <- 3
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group2,3)
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs2 <- c(std_devs[4:6])
diag_std_devs <- diag(std_devs2)
cov_matrix2 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix2)
cluster2=c(minvariance[1:3])



### cluster 3
d <- 2
cor_matrix <- diag(1, d)
cor_matrix[lower.tri(cor_matrix)] <- rep(group3,1)
```

```
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
std_devs2 <- c(std_devs[7:8])
diag_std_devs <- diag(std_devs2)
cov_matrix2 <- diag_std_devs %*% cor_matrix %*% diag_std_devs
minvariance=minvar(cov_matrix2)
cluster3=c(minvariance[1:2])


#between clusters
varS1=cluster1[1]^2*std_devs[1]^2+cluster1[2]^2*std_devs[2]^2+cluster1[3]^2*std_devs[3]^2+
  2*cluster1[1]*cluster1[2]*correlations[1]*std_devs[1]*std_devs[2]+
  2*cluster1[1]*cluster1[3]*correlations[2]*std_devs[1]*std_devs[3]+
  2*cluster1[2]*cluster1[3]*correlations[8]*std_devs[2]*std_devs[3]

varS2=cluster2[1]^2*std_devs[4]^2+cluster2[2]^2*std_devs[5]^2+cluster2[3]^2*std_devs[6]^2+
  2*cluster2[1]*cluster2[2]*correlations[19]*std_devs[4]*std_devs[5]+
  2*cluster2[1]*cluster2[3]*correlations[20]*std_devs[4]*std_devs[6]+
  2*cluster2[2]*cluster2[3]*correlations[23]*std_devs[5]*std_devs[6]


varS3=cluster3[1]^2*std_devs[7]^2+cluster3[2]^2*std_devs[8]^2+2*cluster3[1]*cluster3[2]*correlations[len


weights=c(cluster1,cluster2)
combinations=expand.grid(c(1:3),c(4:6))
covarianceS1S2=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S2=covarianceS1S2+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster1,cluster3)
combinations=expand.grid(c(1:3),c(7:8))
weights=c(cluster1,rep(0,3),cluster3)

covarianceS1S3=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S3=covarianceS1S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster2,cluster3)
combinations=expand.grid(c(4:6),c(7:8))
covarianceS2S3=0
weights=c(rep(0,3),cluster2,cluster3)
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS2S3=covarianceS2S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}
```

```r
cov_matrixBetweenClusters=matrix(c(varS1,covarianceS1S2,covarianceS1S3,
                                   covarianceS1S2,varS2,covarianceS2S3,
                                   covarianceS1S3,covarianceS2S3,varS3),nrow = 3, byrow = TRUE)
minvariance=minvar(cov_matrixBetweenClusters)
Opt1Between=c(minvariance[1:3])
varianceOpt1=t(Opt1Between)%*%cov_matrixBetweenClusters%*%Opt1Between
ESOpt1=sqrt(varianceOpt1)*dnorm(qnorm(0.95))/(1-0.95)




##################
##Optimization 2##
##################
#Equal weights in clusters
cluster1Eq=rep(1/3,3)
cluster2Eq=rep(1/3,3)
cluster3Eq=rep(1/2,2)

varS1=cluster1Eq[1]^2*std_devs[1]^2+cluster1Eq[2]^2*std_devs[2]^2+cluster1Eq[3]^2*std_devs[3]^2+
  2*cluster1Eq[1]*cluster1Eq[2]*correlations[1]*std_devs[1]*std_devs[2]+
  2*cluster1Eq[1]*cluster1Eq[3]*correlations[2]*std_devs[1]*std_devs[3]+
  2*cluster1Eq[2]*cluster1Eq[3]*correlations[8]*std_devs[2]*std_devs[3]

varS2=cluster2Eq[1]^2*std_devs[4]^2+cluster2Eq[2]^2*std_devs[5]^2+cluster2Eq[3]^2*std_devs[6]^2+
  2*cluster2Eq[1]*cluster2Eq[2]*correlations[19]*std_devs[4]*std_devs[5]+
  2*cluster2Eq[1]*cluster2Eq[3]*correlations[20]*std_devs[4]*std_devs[6]+
  2*cluster2Eq[2]*cluster2Eq[3]*correlations[23]*std_devs[5]*std_devs[6]


varS3=cluster3Eq[1]^2*std_devs[7]^2+cluster3Eq[2]^2*std_devs[8]^2+2*cluster3Eq[1]*cluster3Eq[2]*correla


weights=c(cluster1Eq,cluster2Eq)
combinations=expand.grid(c(1:3),c(4:6))
covarianceS1S2=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S2=covarianceS1S2+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}



weights=c(cluster1Eq,cluster3Eq)
combinations=expand.grid(c(1:3),c(7:8))
weights=c(cluster1Eq,rep(0,3),cluster3Eq)

covarianceS1S3=0
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS1S3=covarianceS1S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}
```

```
weights=c(cluster2Eq,cluster3Eq)
combinations=expand.grid(c(4:6),c(7:8))
covarianceS2S3=0
weights=c(rep(0,3),cluster2Eq,cluster3Eq)
for (i in 1:dim(combinations)[1]){
  row=as.numeric(combinations[i,])
  covarianceS2S3=covarianceS2S3+weights[row[1]]*weights[row[2]]*cov_matrix[row[1],row[2]]
}

cov_matrixBetweenClusters=matrix(c(varS1,covarianceS1S2,covarianceS1S3,
                                   covarianceS1S2,varS2,covarianceS2S3,
                                   covarianceS1S3,covarianceS2S3,varS3),nrow = 3, byrow = TRUE)
minvariance=minvar(cov_matrixBetweenClusters)
Opt2Between=c(minvariance[1:3])
weightsOpt2=c(cluster1*Opt2Between[1],cluster2*Opt2Between[2],cluster3*Opt2Between[3])
varianceOpt2=t(weightsOpt2)%*%cov_matrix%*%weightsOpt2
ESOpt2=sqrt(varianceOpt2)*dnorm(qnorm(0.95))/(1-0.95)
```

```
##################
##Optimization 3##
#################
##representative cluster 1
correlations=c(group1,group1,rep(between,5),
               group1,rep(between,5),
               rep(between,5),
               group2,group2,rep(between,2),
               group2,rep(between,2),
               rep(between,2),
               group3)

# Number of variables (size of the matrix)
d <- 8
cor_matrix <- diag(1, d)

# Fill the lower triangular part
cor_matrix[lower.tri(cor_matrix)] <- correlations

# Mirror the lower triangle to the upper triangle
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
```

```
toMinimize1=std_devs[1:3]
for (i in 1:3){
  sum=0
  for (j in 4:8){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize1[i]=toMinimize1[i]*sum
}
clusterRepresentative1=which.min(toMinimize1)


#representative cluster 2
toMinimize2=std_devs[4:6]
for (i in 4:6){
  sum=0
  for (j in c(1,2,3,7,8)){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize2[i-3]=toMinimize2[i-3]*sum
}
clusterRepresentative2=which.min(toMinimize2)+3


#representative cluster 3
toMinimize3=std_devs[7:8]
for (i in 7:8){
  sum=0
  for (j in c(1,2,3,4,5,6)){
    sum=sum+asin(cor_matrix[i,j])*2/pi
  }
  toMinimize3[i-6]=toMinimize3[i-6]*sum
}
clusterRepresentative3=which.min(toMinimize3)+6




#representatives 1,4,7
correlationsRepresentatives=c(0.5,0.5,0.5)
cor_matrixRepresentatives <- diag(1, 3)
cor_matrixRepresentatives[lower.tri(cor_matrixRepresentatives)] <- correlationsRepresentatives
cor_matrixRepresentatives <- cor_matrixRepresentatives + t(cor_matrixRepresentatives) - diag(1, 3)
std_devsRepresentatives=c(2,2,3)
diag_std_devsRepresentatives <- diag(std_devsRepresentatives)
cov_matrixRepresentatives <- diag_std_devsRepresentatives %*% cor_matrixRepresentatives %*% diag_std_dev

minvarianceRepresentatives=minvar(cov_matrixRepresentatives)


WeigthsRepresentatives=c(minvarianceRepresentatives[1:3])
varianceRepresentatives=t(WeigthsRepresentatives)%*%cov_matrixRepresentatives%*%WeigthsRepresentatives
```

```
ESRepresentatives=sqrt(varianceRepresentatives)*dnorm(qnorm(0.95))/(1-0.95)




##################
##Optimization 4##
##################
weightsOpt4=c(Opt2Between[1]*c(1/3,1/3,1/3),Opt2Between[2]*c(1/3,1/3,1/3),Opt2Between[3]*c(1/2,1/2))
varianceOpt4=t(weightsOpt4)%*%cov_matrix%*%weightsOpt4
ESOpt4=sqrt(varianceOpt4)*dnorm(qnorm(0.95))/(1-0.95)


##################
##Optimization 5##
##################

weightsOpt5=c(1/3*c(cluster1), 1/3*c(cluster2), 1/3*c(cluster3)) #Via optimization 1
varianceOpt5=t(weightsOpt5)%*%cov_matrix%*%weightsOpt5
ESOpt5=sqrt(varianceOpt5)*dnorm(qnorm(0.95))/(1-0.95)



######
##Summary
####

# Single step
show_results("Single-step: optimal weight and risk", singlestep, ESSingle)
```

```
##
##  Single-step: optimal weight and risk
## [1] 0.2 0.0 0.2 0.2 0.2 0.2 0.0 0.0
##       [,1]
## [1,] 3.196
```

```
# Double step
show_results("Optimization 1: optimal weight and risk",
             c(cluster1*Opt1Between[1], cluster2*Opt1Between[2], cluster3*Opt1Between[3]),
             ESOpt1)
```

```
##
##  Optimization 1: optimal weight and risk
## [1] 0.18 0.05 0.18 0.20 0.20 0.20 0.00 0.00
##       [,1]
## [1,] 3.208
```

```
show_results("Optimization 2: optimal weight and risk", weightsOpt2, ESOpt2)
```

```
##
##  Optimization 2: optimal weight and risk
## [1] 0.15 0.04 0.15 0.22 0.22 0.22 0.00 0.00
##       [,1]
## [1,] 3.213
```

```r
show_results("Optimization 3: optimal weight and risk", WeigthsRepresentatives, ESRepresentatives)
```

```
##
##  Optimization 3: optimal weight and risk
## [1] 0.5 0.5 0.0
##       [,1]
## [1,] 3.573
```

```r
show_results("Optimization 4: optimal weight and risk", weightsOpt4, ESOpt4, r_digits = 2)
```

```
##
##  Optimization 4: optimal weight and risk
## [1] 0.11 0.11 0.11 0.22 0.22 0.22 0.00 0.00
##      [,1]
## [1,] 3.26
```

```r
show_results("Optimization 5: optimal weight and risk", weightsOpt5, ESOpt5, r_digits = 2)
```

```
##
##  Optimization 5: optimal weight and risk
## [1] 0.15 0.04 0.15 0.11 0.11 0.11 0.17 0.17
##      [,1]
## [1,] 3.68
```

Example 1: multivariate normal distribution - Statistical inference

This R script corresponds to Section 7.1 (Example 1: multivariate normal distribution). We consider the simulation study for setting 1. First, we define the setup, i.e. correlation matrix and standard deviations. Next, for each of the Optimization methods (Single-step and Optimizations 1–5) we estimate for each of 500 Monte Carlo runs the optimal portfolio weights and the corresponding risk.

We define the setup, i.e. correlation matrix and standard deviations, as detailed in Table 1 for setting 1.

```r
between=0.1
group1=0.8
group2=0.7
group3=0.6
correlations=c(group1,group1,c(0.2,0.2,0.2,0.2,0.2),
               group1,rep(between,5),
               rep(between,5),
               group2,group2,rep(between,2),
               group2,rep(between,2),
               rep(between,2),
               group3)

# Number of variables (size of the matrix)
d <- 8
cor_matrix <- diag(1, d)

# Fill the lower triangular part
cor_matrix[lower.tri(cor_matrix)] <- correlations

# Mirror the lower triangle to the upper triangle
cor_matrix <- cor_matrix + t(cor_matrix) - diag(1, d)
```

```r
# Display the correlation matrix
print(cor_matrix)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]  1.0  0.8  0.8  0.2  0.2  0.2  0.2  0.2
## [2,]  0.8  1.0  0.8  0.1  0.1  0.1  0.1  0.1
## [3,]  0.8  0.8  1.0  0.1  0.1  0.1  0.1  0.1
## [4,]  0.2  0.1  0.1  1.0  0.7  0.7  0.1  0.1
## [5,]  0.2  0.1  0.1  0.7  1.0  0.7  0.1  0.1
## [6,]  0.2  0.1  0.1  0.7  0.7  1.0  0.1  0.1
## [7,]  0.2  0.1  0.1  0.1  0.1  0.1  1.0  0.6
## [8,]  0.2  0.1  0.1  0.1  0.1  0.1  0.6  1.0
```

```r
eigen(cor_matrix)$values
```

```
## [1] 3.0336198 2.0942368 1.4892529 0.4000000 0.3000000 0.3000000 0.2000000
## [8] 0.1828905
```

```r
# Compute standard deviations from variances
std_devs=c(2,2.5,2,2,2,2,3,3)
# Compute the diagonal matrix of standard deviations
diag_std_devs <- diag(std_devs)

# Compute the covariance matrix
cov_matrix <- diag_std_devs %*% cor_matrix %*% diag_std_devs
```

Single-step Optimization

```r
set.seed(1)
SingleStepWeights=matrix(NA,nrow=500,ncol=8)
SingleStepES=rep(NA,500)

for (j in 1:500){
  sample=mvrnorm(n=50,mu=rep(0,8),Sigma=cov_matrix)
  EstimatedCov=cov(sample)
  minvariance=minvar(EstimatedCov)
  singlestep=c(minvariance[1:8])
  varianceSingle=t(singlestep)%*%EstimatedCov%*%singlestep
  ESSingle=sqrt(varianceSingle)*dnorm(qnorm(0.95))/(1-0.95)
  SingleStepWeights[j,]=singlestep
  SingleStepES[j]=ESSingle
}

round(colMeans(SingleStepWeights),digits=2)
```

```
## [1] 0.05 0.03 0.29 0.15 0.15 0.17 0.09 0.08
```

```r
mean(SingleStepES)
```

```
## [1] 2.556909
```

Optimization 1: with parametric stage 2

```r
set.seed(1)
Opt1Weights=matrix(NA,nrow=500,ncol=8)
```

```r
Opt1ES=rep(NA,500)

for (j in 1:500){
  sample=mvrnorm(n=50,mu=rep(0,8),Sigma=cov_matrix)

  ### cluster 1
  cov_matrix1 <- cov(sample[,1:3])
  minvariance=minvar(cov_matrix1)
  cluster1=c(minvariance[1:3])

  ### cluster 2
  cov_matrix2 <- cov(sample[,4:6])
  minvariance=minvar(cov_matrix2)
  cluster2=c(minvariance[1:3])

  ### cluster 3
  cov_matrix3 <- cov(sample[,7:8])
  minvariance=minvar(cov_matrix3)
  cluster3=c(minvariance[1:2])

  #between clusters
  variable1=rowSums(t(cluster1*t(sample[,1:3])))
  variable2=rowSums(t(cluster2*t(sample[,4:6])))
  variable3=rowSums(t(cluster3*t(sample[,7:8])))

  cov_matrixBetweenClusters=cov(cbind(variable1,variable2,variable3))
  minvariance=minvar(cov_matrixBetweenClusters)
  Opt1Between=c(minvariance[1:3])
  varianceOpt1=t(Opt1Between)%*%cov_matrixBetweenClusters%*%Opt1Between
  ESOpt1=sqrt(varianceOpt1)*dnorm(qnorm(0.95))/(1-0.95)

  Opt1Weights[j,]=c(cluster1*Opt1Between[1],cluster2*Opt1Between[2],cluster3*Opt1Between[3])
  Opt1ES[j]=ESOpt1
}

round(colMeans(Opt1Weights),digits=2)
```

```
## [1] 0.18 0.01 0.19 0.15 0.15 0.16 0.08 0.08
```

```r
mean(Opt1ES)
```

```
## [1] 2.638718
```

Optimization 1: with nonparametric stage 2

```r
###### Generate weigths
generate_combinations <- function(current, remaining, max_sum) {
  if (length(remaining) == 0) {
    last_value <- max_sum - sum(current)
    if (last_value > 0) {
      return(list(c(current, last_value)))
    } else {
      return(list())
    }
```

```
  }

  result <- list()
  w <- seq(0.01, 0.99, 0.01)
  for (value in w) {
    new_sum <- sum(current) + value
    if (new_sum < max_sum) {
      new_current <- c(current, value)
      result <- c(result, generate_combinations(new_current, remaining[-1], max_sum))
    }
  }

  return(result)
}

# Generate combinations
combinations <- generate_combinations(numeric(0), rep(1, 2), 1)
combinations <- do.call(rbind, combinations)
combinations <- as.matrix(combinations)
Grid=combinations
dim(Grid)





apply_ES <- function(tau, weight_matrix, GiantSampleMatrix) {
  results <- apply(weight_matrix, 1, function(weight) {
    Zcol <- rowSums(t(t(GiantSampleMatrix) * weight))

    # Compute empirical CDF
    FecdfEvaluated <- rank(Zcol) / (length(Zcol)+1)

    # Define d_tau, partialLoss, lambda, and findroot
    d_tau=function(x,tau){1/(1-tau)*as.numeric(x>=tau)*as.numeric(x<=1)*as.numeric(x>=0)}


    partialLoss <- function(z, t) {
      -2 * (z - t)
    }

    DFZ <- matrix(d_tau(FecdfEvaluated, tau), nrow = 1)

    lambda <- function(t, tau) {
      return(1 / length(DFZ) * DFZ %*% matrix(partialLoss(Zcol, t), ncol = 1))
    }

    findroot <- function(tau) {
      uniroot(function(t) {
        lambda(t, tau)
      }, lower = min(Zcol), upper = max(Zcol), extendInt = "yes")$root
    }
```

```
    findroot(tau)
  })
  return(results)
}


set.seed(1)
Opt1Weights=matrix(NA,nrow=500,ncol=8)
Opt1ES=rep(NA,500)


nCores <- detectCores() - 1
cl_outer <- makeCluster(nCores)
registerDoParallel(cl_outer)

for (j in 1:500){
  print(j)
  sample=mvrnorm(n=400,mu=rep(0,8),Sigma=cov_matrix)

  ### cluster 1
  cov_matrix1 <- cov(sample[,1:3])
  minvariance=minvar(cov_matrix1)
  cluster1=c(minvariance[1:3])

  ### cluster 2
  cov_matrix2 <- cov(sample[,4:6])
  minvariance=minvar(cov_matrix2)
  cluster2=c(minvariance[1:3])

  ### cluster 3
  cov_matrix3 <- cov(sample[,7:8])
  minvariance=minvar(cov_matrix3)
  cluster3=c(minvariance[1:2])



  #between clusters
  variable1=rowSums(t(cluster1*t(sample[,1:3])))
  variable2=rowSums(t(cluster2*t(sample[,4:6])))
  variable3=rowSums(t(cluster3*t(sample[,7:8])))

  u5=pobs(variable1)
  u6=pobs(variable2)
  u7=pobs(variable3)
  betaCop  <- empCopula(cbind(u5,u6,u7),smoothing="beta")
  CopulaSample  <-  rCopula(30000, copula = betaCop)
  GiantSampleMatrix=toEmpMargins(CopulaSample,x=cbind(variable1,variable2,variable3))
  weight_matrix <- Grid


  riskPerWeight <- foreach(j = 1:dim(Grid)[1], .combine = rbind, .packages = c("copula", "stats")) %dopa
    c(apply_ES(0.95, matrix(Grid[j,],nrow=1), GiantSampleMatrix))}
```

```
  Opt1Between <- Grid[which.min(riskPerWeight), ]
  Opt1Weights[j,]=c(cluster1*Opt1Between[1],cluster2*Opt1Between[2],cluster3*Opt1Between[3])
  ESOpt1=min(riskPerWeight)
  Opt1ES[j]=ESOpt1

}
```

Optimization 2

```
set.seed(1)
Opt2Weights=matrix(NA,nrow=500,ncol=8)
Opt2ES=rep(NA,500)

for (j in 1:500){
  sample=mvrnorm(n=50,mu=rep(0,8),Sigma=cov_matrix)
  cov_matrixTotal=cov(sample)
  ### cluster 1
  cov_matrix1 <- cov(sample[,1:3])
  minvariance=minvar(cov_matrix1)
  cluster1=c(minvariance[1:3])

  ### cluster 2
  cov_matrix2 <- cov(sample[,4:6])
  minvariance=minvar(cov_matrix2)
  cluster2=c(minvariance[1:3])

  ### cluster 3
  cov_matrix3 <- cov(sample[,7:8])
  minvariance=minvar(cov_matrix3)
  cluster3=c(minvariance[1:2])

  cluster1Eq=rep(1/3,3)
  cluster2Eq=rep(1/3,3)
  cluster3Eq=rep(1/2,2)


  #between clusters
  variable1=rowSums(t(cluster1Eq*t(sample[,1:3])))
  variable2=rowSums(t(cluster2Eq*t(sample[,4:6])))
  variable3=rowSums(t(cluster3Eq*t(sample[,7:8])))

  cov_matrixBetweenClusters=cov(cbind(variable1,variable2,variable3))
  minvariance=minvar(cov_matrixBetweenClusters)
  Opt2Between=c(minvariance[1:3])
  Opt2Weights[j,]=c(cluster1*Opt2Between[1],cluster2*Opt2Between[2],cluster3*Opt2Between[3])
  varianceOpt2=t(Opt2Weights[j,])%*%cov_matrixTotal%*%Opt2Weights[j,]
  Opt2ES[j]=sqrt(varianceOpt2)*dnorm(qnorm(0.95))/(1-0.95)
}

round(colMeans(Opt2Weights),digits=2)
```

```
## [1] 0.16 0.01 0.17 0.16 0.15 0.17 0.09 0.09
```

```r
mean(Opt2ES)
```

```
## [1] 2.646532
```

Optimization 3

```r
set.seed(1)
Opt3RepresentativesWeights=matrix(0,nrow=500,ncol=8)
Opt3RepresentativesES=rep(NA,500)
Opt3RepresentativesESCluster1=rep(NA,500)
Opt3RepresentativesESCluster2=rep(NA,500)
Opt3RepresentativesESCluster3=rep(NA,500)

for (j in 1:500){
  sample=mvrnorm(n=50,mu=rep(0,8),Sigma=cov_matrix)
  cov_matrixTotal=cov(sample)
  cor_matrix=cor(sample)
  std_devs=apply(sample,2,sd)
  cor_matrixKendall=cor(sample,method="kendall")


  #representative cluster 1
  toMinimize1=std_devs[1:3]
  for (i in 1:3){
    sum=0
    for (k in 4:8){
      sum=sum+abs(cor_matrixKendall[i,k])
    }
    toMinimize1[i]=toMinimize1[i]*sum
  }
  clusterRepresentative1=which.min(toMinimize1)
  Opt3RepresentativesESCluster1[j]=clusterRepresentative1

  #representative cluster 2
  toMinimize2=std_devs[4:6]
  for (i in 4:6){
    sum=0
    for (k in c(1,2,3,7,8)){
      sum=sum+abs(cor_matrixKendall[i,k])
    }
    toMinimize2[i-3]=toMinimize2[i-3]*sum
  }
  clusterRepresentative2=which.min(toMinimize2)+3
  Opt3RepresentativesESCluster2[j]=clusterRepresentative2


  #representative cluster 3
  toMinimize3=std_devs[7:8]
  for (i in 7:8){
    sum=0
    for (k in c(1,2,3,4,5,6)){
      sum=sum+abs(cor_matrixKendall[i,k])
    }
```

```
    toMinimize3[i-6]=toMinimize3[i-6]*sum
  }
  clusterRepresentative3=which.min(toMinimize3)+6
  Opt3RepresentativesESCluster3[j]=clusterRepresentative3



  correlationsRepresentatives=c(cor_matrix[clusterRepresentative1,clusterRepresentative2],
                              cor_matrix[clusterRepresentative1,clusterRepresentative3],
                              cor_matrix[clusterRepresentative2,clusterRepresentative3])
  cor_matrixRepresentatives <- diag(1, 3)
  cor_matrixRepresentatives[lower.tri(cor_matrixRepresentatives)] <- correlationsRepresentatives
  cor_matrixRepresentatives <- cor_matrixRepresentatives + t(cor_matrixRepresentatives) - diag(1, 3)
  std_devsRepresentatives=std_devs[c(clusterRepresentative1,clusterRepresentative2,clusterRepresentativ
  diag_std_devsRepresentatives <- diag(std_devsRepresentatives)
  cov_matrixRepresentatives <- diag_std_devsRepresentatives %*% cor_matrixRepresentatives %*% diag_std_d
  minvarianceRepresentatives=minvar(cov_matrixRepresentatives)


  WeigthsRepresentatives=c(minvarianceRepresentatives[1:3])
  varianceRepresentatives=t(WeigthsRepresentatives)%*%cov_matrixRepresentatives%*%WeigthsRepresentative
  ESRepresentatives=sqrt(varianceRepresentatives)*dnorm(qnorm(0.95))/(1-0.95)

  Opt3RepresentativesWeights[j,clusterRepresentative1]=WeigthsRepresentatives[1]
  Opt3RepresentativesWeights[j,clusterRepresentative2]=WeigthsRepresentatives[2]
  Opt3RepresentativesWeights[j,clusterRepresentative3]=WeigthsRepresentatives[3]
  Opt3RepresentativesES[j]=ESRepresentatives
}
```

Optimization 4

```
set.seed(1)
Opt4Weights=matrix(NA,nrow=500,ncol=8)
Opt4ES=rep(NA,500)

for (j in 1:500){
  sample=mvrnorm(n=50,mu=rep(0,8),Sigma=cov_matrix)
  cov_matrixTotal=cov(sample)
  cluster1Eq=rep(1/3,3)
  cluster2Eq=rep(1/3,3)
  cluster3Eq=rep(1/2,2)


  #between clusters
  variable1=rowSums(t(cluster1Eq*t(sample[,1:3])))
  variable2=rowSums(t(cluster2Eq*t(sample[,4:6])))
  variable3=rowSums(t(cluster3Eq*t(sample[,7:8])))

  cov_matrixBetweenClusters=cov(cbind(variable1,variable2,variable3))
  minvariance=minvar(cov_matrixBetweenClusters)
  Opt2Between=c(minvariance[1:3])
  Opt4Weights[j,]=c(cluster1Eq*Opt2Between[1],cluster2Eq*Opt2Between[2],cluster3Eq*Opt2Between[3])
  varianceOpt4=t(Opt4Weights[j,])%*%cov_matrixTotal%*%Opt4Weights[j,]
```

```
  Opt4ES[j]=sqrt(varianceOpt4)*dnorm(qnorm(0.95))/(1-0.95)
}
```

Optimization 5

```
set.seed(1)
Opt5Weights=matrix(NA,nrow=500,ncol=8)
Opt5ES=rep(NA,500)

for (j in 1:500){
  sample=mvrnorm(n=50,mu=rep(0,8),Sigma=cov_matrix)
  cov_matrixTotal=cov(sample)
  ### cluster 1
  cov_matrix1 <- cov(sample[,1:3])
  minvariance=minvar(cov_matrix1)
  cluster1=c(minvariance[1:3])

  ### cluster 2
  cov_matrix2 <- cov(sample[,4:6])
  minvariance=minvar(cov_matrix2)
  cluster2=c(minvariance[1:3])

  ### cluster 3
  cov_matrix3 <- cov(sample[,7:8])
  minvariance=minvar(cov_matrix3)
  cluster3=c(minvariance[1:2])




  Opt5Weights[j,]=c(cluster1*1/3,cluster2*1/3,cluster3*1/3)
  varianceOpt5=t(Opt5Weights[j,])%*%cov_matrixTotal%*%Opt5Weights[j,]
  Opt5ES[j]=sqrt(varianceOpt5)*dnorm(qnorm(0.95))/(1-0.95)

}
```

Stock portfolio via sector clustering

Finally, we turn to the code provided for Section 9.2 (Stock Portfolio Application). Our focus lies on the two-step optimization procedure. We begin by loading stock data from Yahoo Finance. Afterwards, we explore the data: we apply the Ljung–Box test to assess the independence of returns and examine a correlation plot. Following this exploration, we proceed to estimation. Specifically, we compute the optimal portfolio weights and the associated risk for Optimizations 1–5. These estimated weights are then evaluated through a backtest, where we assess portfolio performance outside the sample used for optimization.

##Load data Load stock data from Yahoo finance for the period 2024-01-01 until 2025-01-01.

```
start_date <- "2024-01-01"
end_date <- "2025-01-01"
```

Technology sector

```
#Apple
apple_df <- getSymbols('AAPL', src='yahoo', auto.assign=FALSE)
```

```
apple <- exp(as.numeric(diff(log(apple_df$AAPL.Close[start_date <= index(apple_df) & index(apple_df) < e

#Microsoft
microsoft_df <- getSymbols('MSFT', src='yahoo', auto.assign=FALSE)
microsoft <- exp(as.numeric(diff(log(microsoft_df$MSFT.Close[start_date <= index(microsoft_df) & index(m

#Google
google_df <- getSymbols('GOOGL', src='yahoo', auto.assign=FALSE)
google <- exp(as.numeric(diff(log(google_df$GOOGL.Close[start_date <= index(google_df) & index(google_d

#Adobe
adobe_df <- getSymbols('ADBE', src='yahoo', auto.assign=FALSE)
adobe <- exp(as.numeric(diff(log(adobe_df$ADBE.Close[start_date <= index(adobe_df) & index(adobe_df) < e
```

Healthcare sector

```
#Johnson & Johnson
johnson_df <- getSymbols('JNJ', src='yahoo', auto.assign=FALSE)
johnson <- exp(as.numeric(diff(log(johnson_df$JNJ.Close[start_date <= index(johnson_df) & index(johnson_

#Pfizer
pfizer_df <- getSymbols('PFE', src='yahoo', auto.assign=FALSE)
pfizer <- exp(as.numeric(diff(log(pfizer_df$PFE.Close[start_date <= index(pfizer_df) & index(pfizer_df)

#Merck
merck_df <- getSymbols('MRK', src='yahoo', auto.assign=FALSE)
merck <- exp(as.numeric(diff(log(merck_df$MRK.Close[start_date <= index(merck_df) & index(merck_df) < en

#abbvie
abbvie_df <- getSymbols('ABBV', src='yahoo', auto.assign=FALSE)
abbvie <- exp(as.numeric(diff(log(abbvie_df$ABBV.Close[start_date <= index(abbvie_df) & index(abbvie_df)
```

Finance sector

```
#JPMorgan Chase
jpmorgan_df <- getSymbols('JPM', src='yahoo', auto.assign=FALSE)
jpmorgan<- exp(as.numeric(diff(log(jpmorgan_df$JPM.Close[start_date <= index(jpmorgan_df) & index(jpmorg

#Goldman Sachs
goldman_df <- getSymbols('GS', src='yahoo', auto.assign=FALSE)
goldman <- exp(as.numeric(diff(log(goldman_df$GS.Close[start_date <= index(goldman_df) & index(goldman_d

#Bank of America
bank_df <- getSymbols('BAC', src='yahoo', auto.assign=FALSE)
bank <- exp(as.numeric(diff(log(bank_df$BAC.Close[start_date <= index(bank_df) & index(bank_df) < end_da

#Morgan Stanley
stanley_df <- getSymbols('MS', src='yahoo', auto.assign=FALSE)
stanley <- exp(as.numeric(diff(log(stanley_df$MS.Close[start_date <= index(stanley_df) & index(stanley_d
```

Ljung-Box test

```
Box.test(apple,lag=3,type="Ljung-Box")
```

```
##
```

```
##  Box-Ljung test
##
## data:  apple
## X-squared = 3.022, df = 3, p-value = 0.3882
```

```
Box.test(microsoft,lag=3,type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  microsoft
## X-squared = 4.9549, df = 3, p-value = 0.1751
```

```
Box.test(google,lag=3,type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  google
## X-squared = 4.3396, df = 3, p-value = 0.227
```

```
Box.test(adobe,lag=3,type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  adobe
## X-squared = 0.82667, df = 3, p-value = 0.8431
```

```
Box.test(johnson,lag=3,type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  johnson
## X-squared = 2.3389, df = 3, p-value = 0.5051
```

```
Box.test(pfizer,lag=3,type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  pfizer
## X-squared = 1.5774, df = 3, p-value = 0.6645
```

```
Box.test(merck,lag=3,type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  merck
## X-squared = 0.37212, df = 3, p-value = 0.9459
```

```
Box.test(abbvie,lag=3,type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  abbvie
```

```
## X-squared = 2.4874, df = 3, p-value = 0.4776
```

```r
Box.test(jpmorgan,lag=3,type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  jpmorgan
## X-squared = 2.7976, df = 3, p-value = 0.4239
```

```r
Box.test(goldman,lag=3,type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  goldman
## X-squared = 2.92, df = 3, p-value = 0.4041
```

```r
Box.test(bank,lag=3,type="Ljung-Box")
```

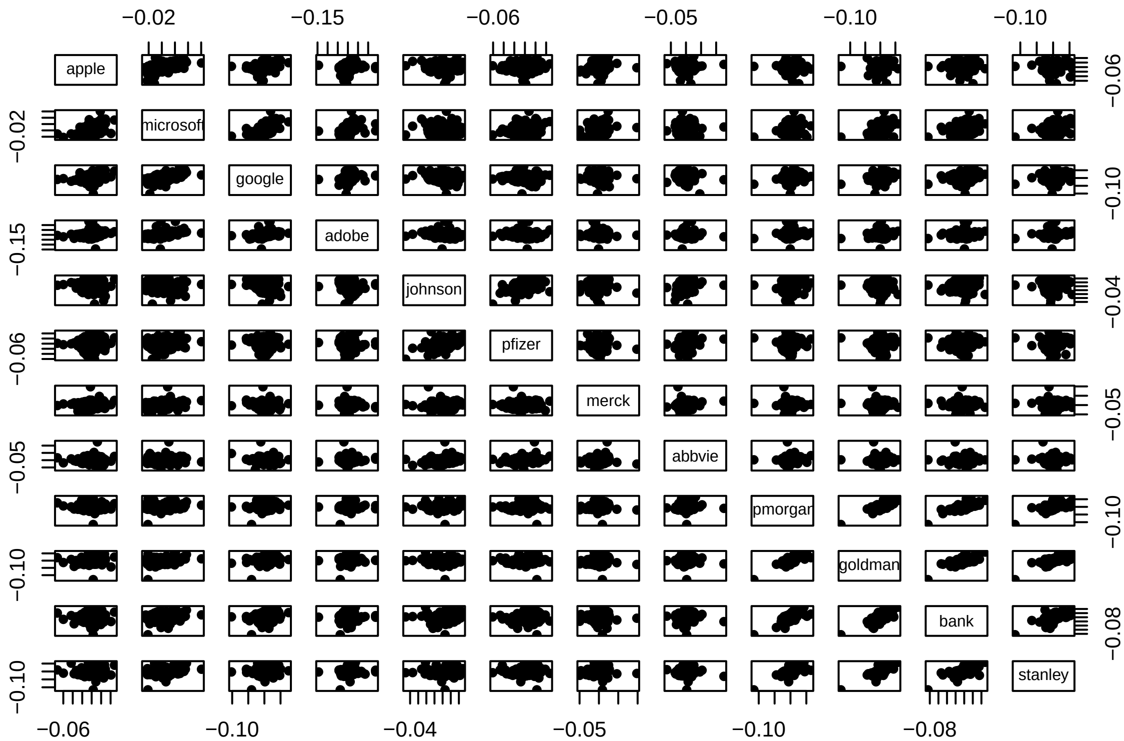
```
##
##  Box-Ljung test
##
## data:  bank
## X-squared = 1.5849, df = 3, p-value = 0.6628
```

```r
Box.test(stanley,lag=3,type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  stanley
## X-squared = 4.1951, df = 3, p-value = 0.2412
```

Correlation plot

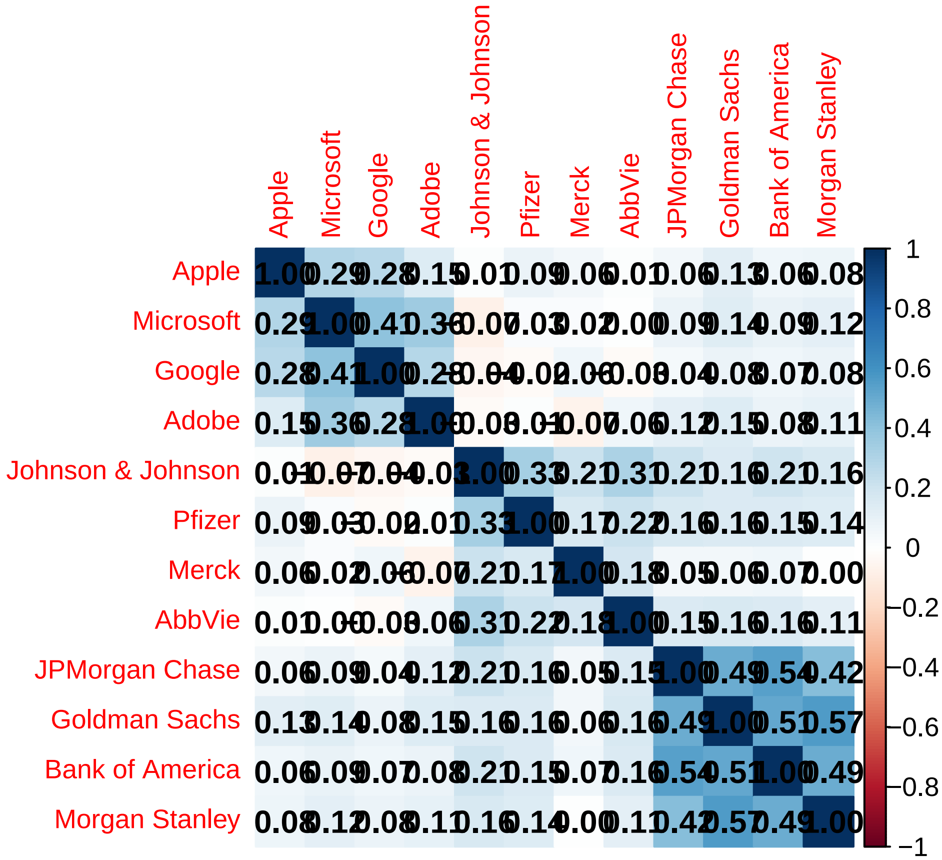
```r
df=data.frame(apple,microsoft,google,adobe,johnson,pfizer,merck,abbvie,jpmorgan,goldman,bank,stanley)
pairs(1-df,pch=16)
```




```
cor_matrix=cor(df,method="kendall")
colnames(cor_matrix)=c("Apple","Microsoft","Google","Adobe","Johnson & Johnson","Pfizer","Merck","AbbVi
rownames(cor_matrix)=c("Apple","Microsoft","Google","Adobe","Johnson & Johnson","Pfizer","Merck","AbbVi
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
corrplot(cor_matrix, method="color", tl.cex=0.8, addCoef.col="black")
```



##Estimation

The following function implements the full nonparametric estimator (NP) from Debrauwer and Gijbels (2024) in the case of Extremiles. Here tau = risk level (0.95 here), and the data should be the portfolio weighted returns. The output is the estimated extremile risk. We also define a helper function called "risk", this function only requires portfolio weights. This latter function then calls ExtrNPNP using tau=0.95 and our dataset of stocks weighted by the given inputvector x.

```
ExtrNPNP=function(tau,data){
  library(copula)
  set.seed(1)
  X=data
  u=pobs(X)
  ecop.orig  <- empCopula(u,smoothing="beta")   #Empirical Beta copula
  U.orig  <-  rCopula(100000, copula = ecop.orig)
  GiantSampleMatrix=toEmpMargins(U.orig,x=X)
  Zcol=rowSums(GiantSampleMatrix)
  FecdfEvaluated=rank(Zcol)/(length(Zcol)+1)
  d_tau=function(x,tau){((log(0.5)/log(1-tau)*(1-x)^(log(0.5)/log(1-tau)-1))*as.numeric(tau>0)*as.numer
  partialLoss=function(z,t){-2*(z-t)}
  lambda=function(t,tau){
    FecdfEvaluated[which(FecdfEvaluated==1)]=0
    DFZ=matrix(d_tau(FecdfEvaluated,tau),nrow=1)
    return(1/length(DFZ)*DFZ%*%matrix(partialLoss(Zcol,t),ncol=1))}
  findroot=function(tau){
    return(uniroot(function(t){lambda(t,tau)},lower=min(Zcol),upper=max(Zcol),extendInt = "yes")$root)
  }
```

```
  EstimatedExtr=findroot(tau)
  return(EstimatedExtr)

}


risk=function(x){
  return(ExtrNPNP(0.95,data=as.matrix(-df)%*%diag(x)))}
```

We implement every Optimization (1–5) separately. Starting with Optimization 1

To work with $\widetilde{R}_t$ as explained in the paper, we define:

```
df=data.frame(apple,microsoft,google,adobe,johnson,pfizer,merck,abbvie,jpmorgan,goldman,bank,stanley)
df=df-1
```

Optimization 1

```
###### Generate weights for within cluster optimization
generate_combinations <- function(current, remaining, max_sum) {
  if (length(remaining) == 0) {
    last_value <- max_sum - sum(current)
    if (last_value > 0) {
      return(list(c(current, last_value)))
    } else {
      return(list())
    }
  }

  result <- list()
  w <- seq(0.01, 0.99, 0.01)
  for (value in w) {
    new_sum <- sum(current) + value
    if (new_sum < max_sum) {
      new_current <- c(current, value)
      result <- c(result, generate_combinations(new_current, remaining[-1], max_sum))
    }
  }

  return(result)
}

# Generate combinations
combinations <- generate_combinations(numeric(0), rep(1, 3), 1)
combinations <- do.call(rbind, combinations)
combinations <- as.matrix(combinations)
Grid=combinations


##Per cluster we determine the optimal weights (stage 1 of Optimization 1)


################################ cluster 1
risk=function(x){
```

```
  return(ExtrNPNP(0.95,data=as.matrix(-df[,1:4])%*%diag(x)))}

tic()
nCores=parallel::detectCores()-2
cl <- parallel::makeCluster(nCores)
doParallel::registerDoParallel(cl)
NPNPrisk=foreach(i=1:dim(Grid)[1], .combine='c',.export=ls(envir=globalenv())) %dopar% risk(Grid[i,])
parallel::stopCluster(cl)
toc()
cluster1=Grid[which.min(NPNPrisk),]
cluster1#0.55 0.40 0.02 0.03
min(NPNPrisk)#0.02026971




################################ cluster 2
risk=function(x){
  return(ExtrNPNP(0.95,data=as.matrix(-df[,5:8])%*%diag(x)))}

tic()
nCores=parallel::detectCores()-2
cl <- parallel::makeCluster(nCores)
doParallel::registerDoParallel(cl)
NPNPrisk=foreach(i=1:dim(Grid)[1], .combine='c',.export=ls(envir=globalenv())) %dopar% risk(Grid[i,])
parallel::stopCluster(cl)
toc()
cluster2=Grid[which.min(NPNPrisk),]
#cluster 2=0.59 0.15 0.21 0.05
cluster2
min(NPNPrisk)#0.0142816



################################ cluster 3
risk=function(x){
  return(ExtrNPNP(0.95,data=as.matrix(-df[,9:12])%*%diag(x)))}

tic()
nCores=parallel::detectCores()-2
cl <- parallel::makeCluster(nCores)
doParallel::registerDoParallel(cl)
NPNPrisk=foreach(i=1:dim(Grid)[1], .combine='c',.export=ls(envir=globalenv())) %dopar% risk(Grid[i,])
parallel::stopCluster(cl)
toc()
cluster3=Grid[which.min(NPNPrisk),]
cluster3# 0.18 0.11 0.56 0.15
min(NPNPrisk)#0.02048599





### Generate grid of weights for across cluster optimization
```

```r
###### Generate weigths
generate_combinations <- function(current, remaining, max_sum) {
  if (length(remaining) == 0) {
    last_value <- max_sum - sum(current)
    if (last_value > 0) {
      return(list(c(current, last_value)))
    } else {
      return(list())
    }
  }

  result <- list()
  w <- seq(0.01, 0.99, 0.01)
  for (value in w) {
    new_sum <- sum(current) + value
    if (new_sum < max_sum) {
      new_current <- c(current, value)
      result <- c(result, generate_combinations(new_current, remaining[-1], max_sum))
    }
  }

  return(result)
}

# Generate combinations
combinations <- generate_combinations(numeric(0), rep(1, 2), 1)
combinations <- do.call(rbind, combinations)
combinations <- as.matrix(combinations)
Grid=combinations



##Across cluster optimization (stage 2 of Optimization 1)

variable1=rowSums(t(cluster1*t(df[,1:4])))
variable2=rowSums(t(cluster2*t(df[,5:8])))
variable3=rowSums(t(cluster3*t(df[,9:12])))

risk=function(x){
  return(ExtrNPNP(0.95,data=cbind(-variable1,-variable2,-variable3)%*%diag(x)))}

tic()
nCores=parallel::detectCores()-2
cl <- parallel::makeCluster(nCores)
doParallel::registerDoParallel(cl)
NPNPrisk=foreach(i=1:dim(Grid)[1], .combine='c',.export=ls(envir=globalenv())) %dopar% risk(Grid[i,])
parallel::stopCluster(cl)
toc()
Opt1Between=Grid[which.min(NPNPrisk),]#0.28 0.60 0.12
ExtrOpt1=min(NPNPrisk)#0.01192556
Opt1Between
ExtrOpt1
```

```r
#Final weight:
#c(0.28*c(0.55,0.40,0.02,0.03), 0.60*c(0.59,0.15,0.21,0.05) ,0.12*c(0.18, 0.11, 0.56, 0.15))

#Risk: 0.01192556
```

Optimization 2

```r
#Stage 1 of Optimization 2: uniform within cluster weights
variable1=rowSums(t(rep(1/4,4)*t(df[,1:4])))
variable2=rowSums(t(rep(1/4,4)*t(df[,5:8])))
variable3=rowSums(t(rep(1/4,4)*t(df[,9:12])))


##Determine across cluster weights
risk=function(x){
  return(ExtrNPNP(0.95,data=cbind(-variable1,-variable2,-variable3)%*%diag(x)))}

tic()
nCores=parallel::detectCores()-2
cl <- parallel::makeCluster(nCores)
doParallel::registerDoParallel(cl)
NPNPrisk=foreach(i=1:dim(Grid)[1], .combine='c',.export=ls(envir=globalenv())) %dopar% risk(Grid[i,])
parallel::stopCluster(cl)
toc()
Opt2Between=Grid[which.min(NPNPrisk),]#0.21 0.59 0.20
ExtrOpt2=min(NPNPrisk)#0.0130436


##Determine final risk, using again the optimized within cluster weights
variable1=rowSums(t(cluster1*t(df[,1:4])))
variable2=rowSums(t(cluster2*t(df[,5:8])))
variable3=rowSums(t(cluster3*t(df[,9:12])))


risk=function(x){
  return(ExtrNPNP(0.95,data=cbind(-variable1,-variable2,-variable3)%*%diag(x)))}
ExtrOpt2FinalRisk=risk(c(Opt2Between))#0.01206293

FinalWeightOpt2=c(cluster1*Opt2Between[1],cluster2*Opt2Between[2],cluster3*Opt2Between[3])
# 0.1155 0.0840 0.0042 0.0063 0.3481 0.0885 0.1239 0.0295 0.0360 0.0220 0.1120 0.0300
```

Optimization 3

```r
#Estimate the 95% Extremile of the margins
ExtrMargins=function(data){
  tau=0.95
  d_tau <- function(x, tau) {
    (log(0.5) / log(tau) * x^(log(0.5) / log(tau) - 1) * as.numeric(tau > 0.5) * as.numeric(tau <= 1))
  }

  partialLoss <- function(z, t) {
    -2 * (z - t)
```

```
  }
  cdfEvaluated=rank(data)/(length(data)+1)

  lambda=function(t,tau){
    cdfEvaluated[which(cdfEvaluated==1)]=0
    DFZ=matrix(d_tau(cdfEvaluated,tau),nrow=1)
    return(1/length(DFZ)*DFZ%*%matrix(partialLoss(data,t),ncol=1))}
  findroot=function(tau){
    return(uniroot(function(t){lambda(t,tau)},lower=min(data),upper=max(data),extendInt = "yes")$root)
  }
  EstimatedExtr=findroot(tau)
  return(EstimatedExtr)
}


##Find the representative for each cluster.


#representative cluster 1
toMinimize1=c(ExtrMargins(-df[,1]),ExtrMargins(-df[,2]),ExtrMargins(-df[,3]),ExtrMargins(-df[,4]))
for (i in 1:4){
  sum=0
  for (k in 5:12){
    sum=sum+abs(cor(df[,i],df[,k],method="kendall"))
  }
  toMinimize1[i]=toMinimize1[i]*sum
}
clusterRepresentative1=which.min(toMinimize1)
#1




#representative cluster 2
toMinimize2=c(ExtrMargins(-df[,5]),ExtrMargins(-df[,6]),ExtrMargins(-df[,7]),ExtrMargins(-df[,8]))
for (i in 5:8){
  sum=0
  for (k in c(1:4,9:12)){
    sum=sum+abs(cor(df[,i],df[,k],method="kendall"))
  }
  toMinimize2[i-4]=toMinimize2[i-4]*sum
}
clusterRepresentative2=which.min(toMinimize2)+4
#7




#representative cluster 3
toMinimize3=c(ExtrMargins(-df[,9]),ExtrMargins(-df[,10]),ExtrMargins(-df[,11]),ExtrMargins(-df[,12]))
for (i in 9:12){
  sum=0
  for (k in c(1:8)){
```

```
    sum=sum+abs(cor(df[,i],df[,k],method="kendall"))
  }
  toMinimize3[i-8]=toMinimize3[i-8]*sum
}
clusterRepresentative3=which.min(toMinimize3)+8
#11


##Optimized portfolio using the representatives
risk=function(x){
  return(ExtrNPNP(0.95,data=cbind(-df[,clusterRepresentative1],-df[,clusterRepresentative2],-df[,cluste

tic()
nCores=parallel::detectCores()-2
cl <- parallel::makeCluster(nCores)
doParallel::registerDoParallel(cl)
NPNPrisk=foreach(i=1:dim(Grid)[1], .combine='c',.export=ls(envir=globalenv())) %dopar% risk(Grid[i,])
parallel::stopCluster(cl)
toc()
weightsOpt3=Grid[which.min(NPNPrisk),]#0.35 0.30 0.35
min(NPNPrisk)#0.0146372
#Total portfolio weight: 0.35  0  0  0  0  0  0.30  0  0  0  0.35  0
```

Optimization 4

```
#From the first stage of Optimization 2 we know the optimal across cluster weights and corresponding ri
#when using uniform within cluster weights.
c(0.21*rep(1/4,4),0.59*rep(1/4,4),0.20*rep(1/4,4))
#Final weight: 0.0525 0.0525 0.0525 0.0525 0.1475 0.1475 0.1475 0.1475 0.0500 0.0500 0.0500 0.0500
#with risk 0.0130436
```

Optimization 5

```
#From stage 1 of optimization 1 we know the optimal within cluster weights
cluster1=c(0.55,0.40,0.02,0.03)
cluster2=c(0.59,0.15,0.21,0.05)
cluster3=c(0.18,0.11,0.56,0.15)




variable1=rowSums(t(cluster1*t(df[,1:4])))
variable2=rowSums(t(cluster2*t(df[,5:8])))
variable3=rowSums(t(cluster3*t(df[,9:12])))



##determine across cluster weights
risk=function(x){
  return(ExtrNPNP(0.95,data=cbind(-variable1,-variable2,-variable3)%*%diag(x)))}

ExtrOpt5=risk(c(1/3,1/3,1/3))#0.013001

#weight:
```

```r
round(c(1/3*cluster1 ,1/3*cluster2, 1/3*cluster3),digits=3)
#0.183333333 0.133333333 0.006666667 0.010000000 0.196666667 0.050000000 0.070000000
#0.016666667 0.060000000 0.036666667 0.186666667 0.050000000
```

Backtest

We apply the apply portfolio weights (Opt1–Opt5 and single-step) to the out-of-sample period Jan–Jun 2025. Returns are plotted as 'losses' (positive values) vs 'gains' (negative).

First we load the data.

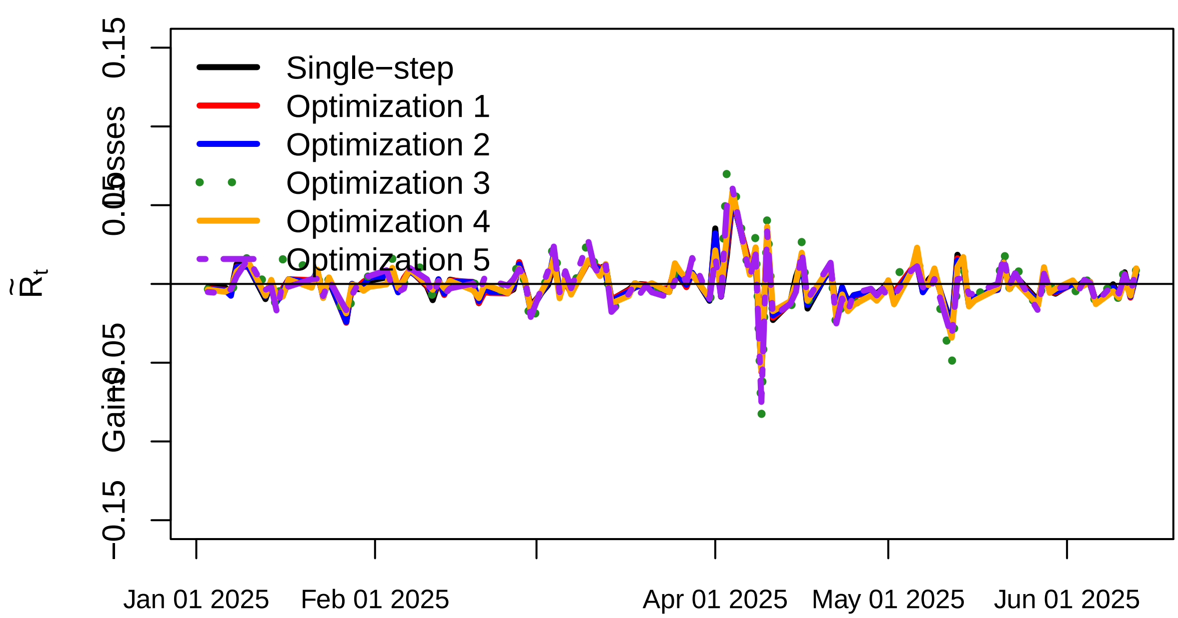
```r
start_date <- "2025-01-01"
end_date <- "2025-06-13"



################################################################################
##########################     Technology        ##############################
################################################################################

#Apple
apple_df <- getSymbols('AAPL', src='yahoo', auto.assign=FALSE)
apple <- exp(as.numeric(diff(log(apple_df$AAPL.Close[start_date <= index(apple_df) & index(apple_df) <=

apple_close <- Cl(apple_df)[paste0(start_date, "/", end_date)]
gross_returns <- apple_close[-1] / lag(apple_close, k = 1)[-1]

#Microsoft
microsoft_df <- getSymbols('MSFT', src='yahoo', auto.assign=FALSE)
microsoft <- exp(as.numeric(diff(log(microsoft_df$MSFT.Close[start_date <= index(microsoft_df) & index(

#Google
google_df <- getSymbols('GOOGL', src='yahoo', auto.assign=FALSE)
google <- exp(as.numeric(diff(log(google_df$GOOGL.Close[start_date <= index(google_df) & index(google_d

#Adobe
adobe_df <- getSymbols('ADBE', src='yahoo', auto.assign=FALSE)
adobe <- exp(as.numeric(diff(log(adobe_df$ADBE.Close[start_date <= index(adobe_df) & index(adobe_df) <=



################################################################################
##########################     Healthcare        ##############################
################################################################################


#Johnson & Johnson
johnson_df <- getSymbols('JNJ', src='yahoo', auto.assign=FALSE)
johnson <- exp(as.numeric(diff(log(johnson_df$JNJ.Close[start_date <= index(johnson_df) & index(johnson_

#Pfizer
pfizer_df <- getSymbols('PFE', src='yahoo', auto.assign=FALSE)
pfizer <- exp(as.numeric(diff(log(pfizer_df$PFE.Close[start_date <= index(pfizer_df) & index(pfizer_df)
```

```
#Merck
merck_df <- getSymbols('MRK', src='yahoo', auto.assign=FALSE)
merck <- exp(as.numeric(diff(log(merck_df$MRK.Close[start_date <= index(merck_df) & index(merck_df) <= e

#abbvie
abbvie_df <- getSymbols('ABBV', src='yahoo', auto.assign=FALSE)
abbvie <- exp(as.numeric(diff(log(abbvie_df$ABBV.Close[start_date <= index(abbvie_df) & index(abbvie_df)




#################################################################################
##################              Finance              ##########################
#################################################################################

#JPMorgan Chase
jpmorgan_df <- getSymbols('JPM', src='yahoo', auto.assign=FALSE)
jpmorgan<- exp(as.numeric(diff(log(jpmorgan_df$JPM.Close[start_date <= index(jpmorgan_df) & index(jpmorg

#Goldman Sachs
goldman_df <- getSymbols('GS', src='yahoo', auto.assign=FALSE)
goldman <- exp(as.numeric(diff(log(goldman_df$GS.Close[start_date <= index(goldman_df) & index(goldman_d

#Bank of America
bank_df <- getSymbols('BAC', src='yahoo', auto.assign=FALSE)
bank <- exp(as.numeric(diff(log(bank_df$BAC.Close[start_date <= index(bank_df) & index(bank_df) <= end_d

#Morgan Stanley
stanley_df <- getSymbols('MS', src='yahoo', auto.assign=FALSE)
stanley <- exp(as.numeric(diff(log(stanley_df$MS.Close[start_date <= index(stanley_df) & index(stanley_d





df=data.frame(apple,microsoft,google,adobe,johnson,pfizer,merck,abbvie,jpmorgan,goldman,bank,stanley)

##Fill in appropriate weights
weightSingle=c(0.14,0.06,0.03,0.04,0.42,0,0.13,0.05,0.06,0,0,0.07)
weightOpt1=c(0.1540,0.1120,0.0056,0.0084,0.3540,0.0900,0.1260,0.0300,0.0216,0.0132,0.0672,0.0180)
weightOpt2=c(0.1155,0.0840,0.0042,0.0063,0.3481,0.0885,0.1239,0.0295,0.0360,0.0220,0.1120,0.0300)
weightOpt3=c(0.35,0,0,0,0,0,0.30,0,0,0,0.35,0)
weightOpt4=c(0.0525,0.0525,0.0525,0.0525,0.1475,0.1475,0.1475,0.1475,0.0500,0.0500,0.0500,0.0500)

cluster1=c(0.55,0.40,0.02,0.03)
cluster2=c(0.59,0.15,0.21,0.05)
cluster3=c(0.18,0.11,0.56,0.15)
weightOpt5=c(1/3*cluster1 ,1/3*cluster2, 1/3*cluster3)


returnsSingle=1-as.matrix(df)%*%matrix(weightSingle,nrow=12,ncol=1)
returns1=1-as.matrix(df)%*%matrix(weightOpt1,nrow=12,ncol=1)
returns2=1-as.matrix(df)%*%matrix(weightOpt2,nrow=12,ncol=1)
```

```
returns3=1-as.matrix(df)%*%matrix(weightOpt3,nrow=12,ncol=1)
returns4=1-as.matrix(df)%*%matrix(weightOpt4,nrow=12,ncol=1)
returns5=1-as.matrix(df)%*%matrix(weightOpt5,nrow=12,ncol=1)
```

We construct a Figure with performance of the different portfolios. Positive values are losses.

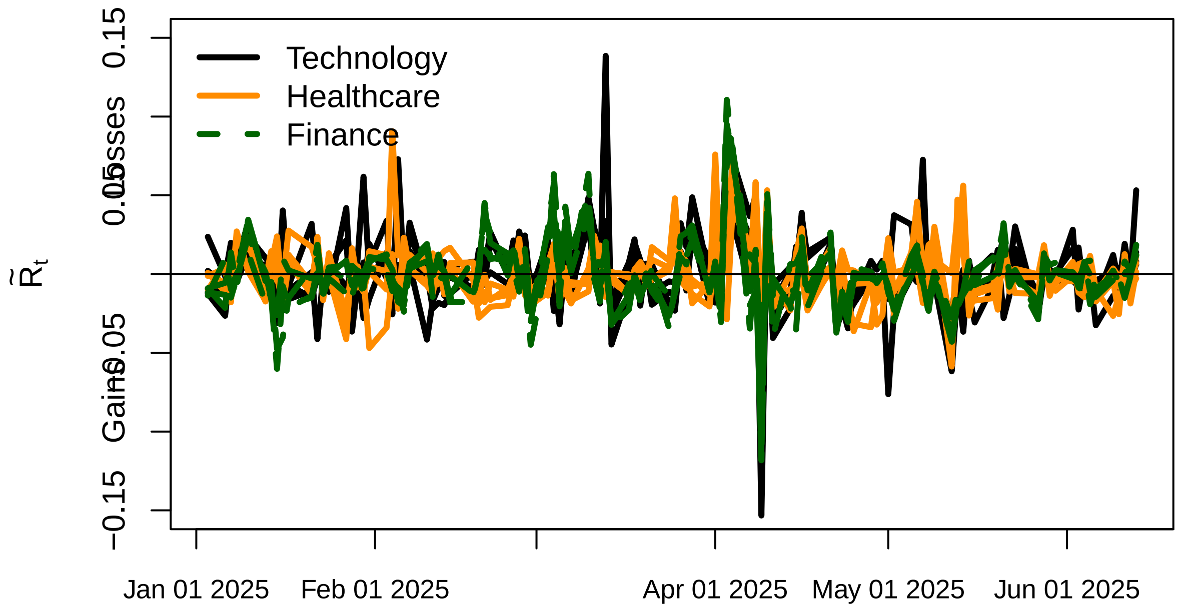
```
par(mar = c(5.1, 4.3, 4.1, 2.1))
plot(index(gross_returns),returnsSingle,type="l",ylim=c(-0.15,0.15),lwd=3,xaxt="n",xlab="",ylab=TeX("$\
pretty_dates <- pretty(index(gross_returns))
axis(1, at = pretty_dates, labels = format(pretty_dates, "%b %d %Y"), cex.axis = 0.8)
ylims <- par("usr")[3:4]
xpos <- par("usr")[1] -10
# "Gains" (for negative values)
text(x = xpos, y = mean(c(ylims[1], 0)), labels = "Gains", srt = 90, adj = 0.5, xpd = TRUE)

# "Losses" (for positive values)
text(x = xpos,  y = mean(c(0, ylims[2])), labels = "Losses", srt = 90, adj = 0.5, xpd = TRUE)
lines(index(gross_returns),returns1,col="red",lwd=3)
lines(index(gross_returns),returns2,col="blue",lwd=3)
lines(index(gross_returns),returns3,col="forestgreen",lwd=4,lty=3)
lines(index(gross_returns),returns4,col="orange",lwd=3)
lines(index(gross_returns),returns5,col="purple",lwd=3,lty=6)
legend("topleft",col=c("black","red","blue","forestgreen","orange","purple"),
       legend=c("Single-step","Optimization 1","Optimization 2","Optimization 3","Optimization 4","Opti
       lty=c(1,1,1,3,1,6),lwd=c(3,3,3,4,3,3),bty="n")
abline(h=0)
```



```r
par(mar = c(5.1, 4.1, 4.1, 2.1))#default
```

Construction of Figure with performance of individual stocks. Stocks from the same cluster have the same color.

```r
par(mar = c(5.1, 4.3, 4.1, 2.1))
plot(index(gross_returns), 1-coredata(gross_returns), type = "l",
     ylim = c(-0.15, 0.15), lwd = 3, ylab = TeX("$\\tilde{R}_t$"), xlab = "", col = "black", xaxt = "n")
pretty_dates <- pretty(index(gross_returns))
axis(1, at = pretty_dates, labels = format(pretty_dates, "%b %d %Y"), cex.axis = 0.8)
ylims <- par("usr")[3:4]
xpos <- par("usr")[1] -10
# "Gains" (for negative values)
text(x = xpos, y = mean(c(ylims[1], 0)), labels = "Gains", srt = 90, adj = 0.5, xpd = TRUE)

# "Losses" (for positive values)
text(x = xpos,  y = mean(c(0, ylims[2])), labels = "Losses", srt = 90, adj = 0.5, xpd = TRUE)


lines(index(gross_returns),1-df[,2], col = "black",lwd=3)
lines(index(gross_returns),1-df[,3], col = "black",lwd=3)
lines(index(gross_returns),1-df[,4], col = "black",lwd=3)

lines(index(gross_returns),1-df[,5], col = "#FF8C00",lwd=3)
lines(index(gross_returns),1-df[,6], col = "#FF8C00",lwd=3)
lines(index(gross_returns),1-df[,7], col = "#FF8C00",lwd=3)
lines(index(gross_returns),1-df[,8], col = "#FF8C00",lwd=3)

lines(index(gross_returns),1-df[,9], col = "#006400",lwd=3,lty=5)
lines(index(gross_returns),1-df[,10], col = "#006400",lwd=3,lty=5)
lines(index(gross_returns),1-df[,11], col = "#006400",lwd=3,lty=5)
lines(index(gross_returns),1-df[,12], col = "#006400",lwd=3,lty=5)
legend("topleft",col=c("black","#FF8C00","#006400"),lwd=c(3,3,3),lty=c(1,1,2),legend=c("Technology","Hea
abline(h=0)
```




```
par(mar = c(5.1, 4.1, 4.1, 2.1))#default
```