High-dimensional copula-based Wasserstein dependence

This file discusses the R code and corresponding output that is used for creating the paper. Useful packages are the following.

```
# Packages
library(mvtnorm)
library(ggplot2)
library(ggpubr)
library(expm)
library(magic)
library(spcov)
library(MASS)
library(covglasso)
library(Matrix)
library(dplyr)
library(pracma)
library(plsgenomics)
library(sets)
library(hash)
library(gtools)
library(dendextend)
library(scales)
library(readxl)
library(SensoMineR)
library(FactoMineR)
library(corrplot)
library(copula)
library(gofCopula)
```

We also set our working directory

```
dir = dirname(rstudioapi::getActiveDocumentContext()$path)
setwd(dir)
```

The first lines of code are related to Example 3 of the paper.

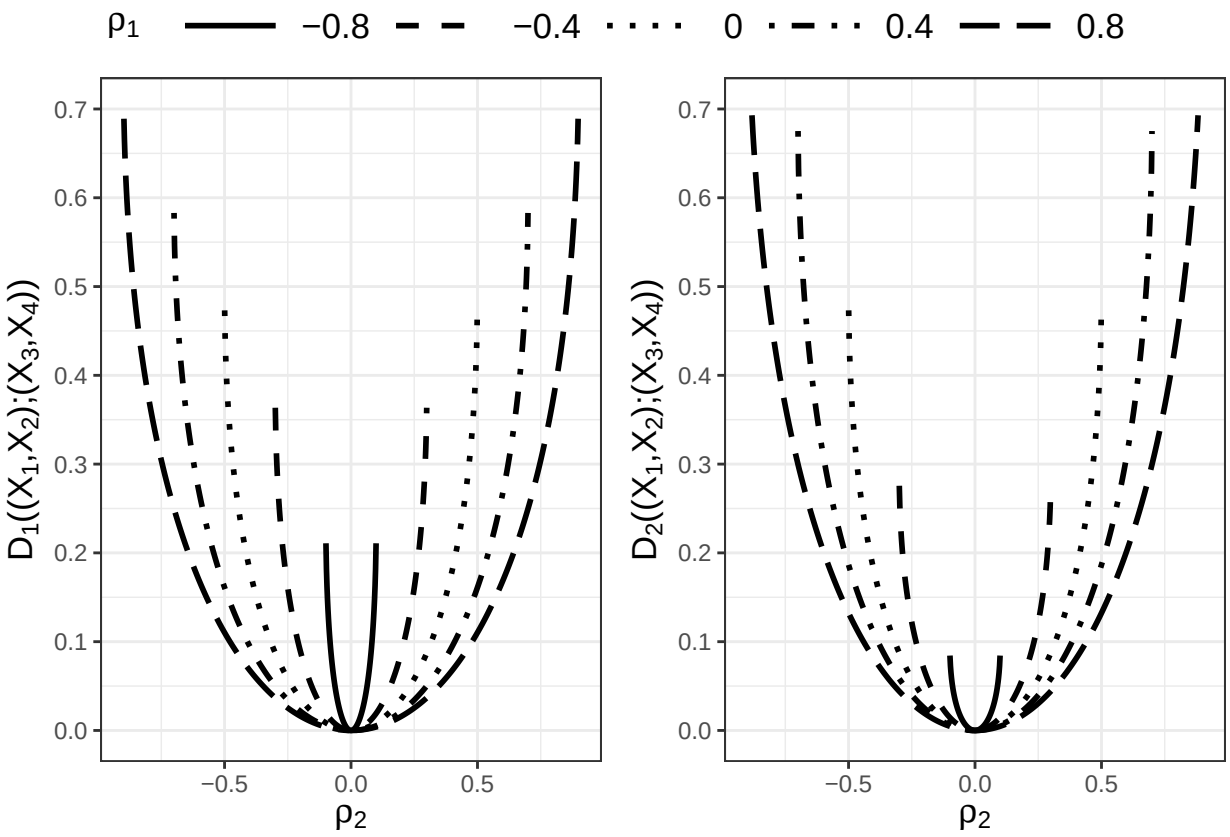
Example 3

We consider (X_1, X_2, X_3, X_4) having a Gaussian copula.

```
rho2 = seq(-0.99999,0.99999,len = 1000) # Sequence of rho2 values

BWD1_binormal = function(rho1,rho2){

  # This function computes equation (19) of the paper

  t1 = 2 * ((1 + rho1)^(1/2)) - ((rho1 - 2*rho2 + 1)^(1/2)) - ((rho1 + 2*rho2 + 1)^(1/2))
  t2 = (2-sqrt(2)) * (((1 + rho1)^(1/2)) + ((1 - rho1)^(1/2)))
  return(t1/t2)
}

values1 = integer(length(rho2))
values2 = integer(length(rho2))
values3 = integer(length(rho2))
values4 = integer(length(rho2))
```

```
values5 = integer(length(rho2))

for(i in 1:length(rho2)){ # Compute D1 under the constraint that rho1 >= 2 |rho2| - 1
  values1[i] = ifelse(-0.8 >= 2 *abs(rho2[i]) - 1, BWD1_binormal(-0.8,rho2[i]), NA)
  values2[i] = ifelse(-0.4 >= 2 *abs(rho2[i]) - 1, BWD1_binormal(-0.4,rho2[i]), NA)
  values3[i] = ifelse(0 >= 2 *abs(rho2[i]) - 1, BWD1_binormal(0,rho2[i]), NA)
  values4[i] = ifelse(0.4 >= 2 *abs(rho2[i]) - 1, BWD1_binormal(0.4,rho2[i]), NA)
  values5[i] = ifelse(0.8 >= 2 *abs(rho2[i]) - 1, BWD1_binormal(0.8,rho2[i]), NA)
}

frame1 = as.data.frame(cbind(rho2[-which(is.na(values1))],
                             values1[-which(is.na(values1))]))
frame2 = as.data.frame(cbind(rho2[-which(is.na(values2))],
                             values2[-which(is.na(values2))]))
frame3 = as.data.frame(cbind(rho2[-which(is.na(values3))],
                             values3[-which(is.na(values3))]))
frame4 = as.data.frame(cbind(rho2[-which(is.na(values4))],
                             values4[-which(is.na(values4))]))
frame5 = as.data.frame(cbind(rho2[-which(is.na(values5))],
                             values5[-which(is.na(values5))]))

# Make plot

g1 = ggplot() + geom_line(data = frame1,aes(x=V1,y=V2,lty = "a"),lwd = 1) +
  geom_line(data = frame2,aes(x=V1,y=V2,lty = "b"),lwd = 1) +
  geom_line(data = frame3,aes(x=V1,y=V2,lty = "c"),lwd = 1) +
  geom_line(data = frame4,aes(x=V1,y=V2,lty = "d"),lwd = 1) +
  geom_line(data = frame5,aes(x=V1,y=V2,lty = "e"),lwd = 1) +
  theme_bw() + xlab(expression(rho[2])) +
  ylab(expression("D"[1] * "((X"[1] * "," * "X"[2] * ");(X"[3] * "," * "X"[4] * "))")) +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=13),
        legend.position = "top",legend.text=element_text(size=13),
        legend.title=element_text(size=13), legend.key.width = unit(1.5, 'cm')) +
  scale_y_continuous(breaks = seq(0,0.7, by = 0.1),limits = c(0,0.7)) +
  scale_linetype_manual(values = c(1,2,3,4,5),
                        labels = c(-0.8,-0.4,0,0.4,0.8),
                        name = expression(rho[1] * "  "))

BWD2_binormal = function(rho1,rho2){

  # This function computes equation (20) of the paper

  t1 = 4 - 2*abs(1-rho1) - (rho1^2 + 2 * rho1 - 2*rho1*rho2 - 2*rho2 + 1)^(1/2) -
       (rho1^2 + 2 * rho1 + 2*rho1*rho2 + 2*rho2 + 1)^(1/2)
  t2 = 4 - sqrt(2) * (abs(1-rho1) + 1 + rho1)

  return(t1/t2)
}

values1 = integer(length(rho2))
values2 = integer(length(rho2))
values3 = integer(length(rho2))
```

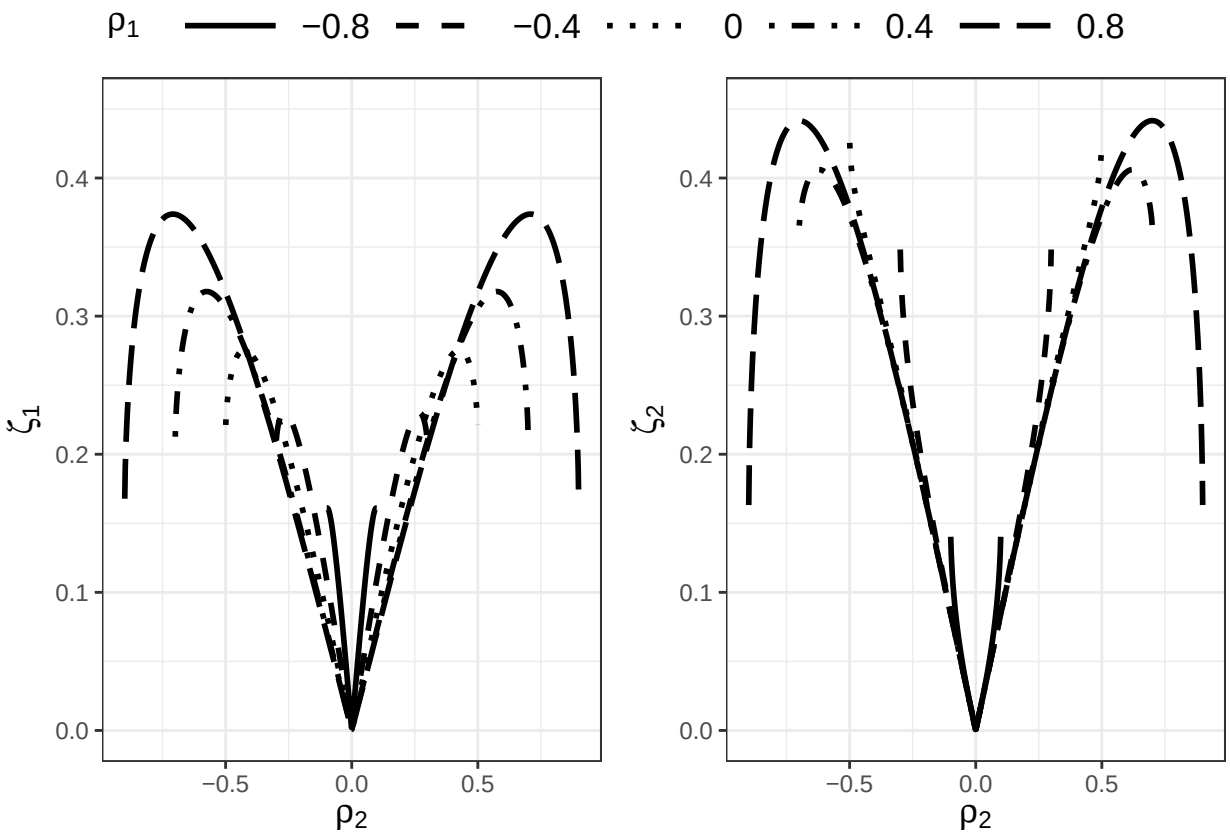
```
values4 = integer(length(rho2))
values5 = integer(length(rho2))

for(i in 1:length(rho2)){ # Compute D2 under the constraint that rho1 >= 2 |rho2| - 1
  values1[i] = ifelse(-0.8 >= 2 *abs(rho2[i]) - 1, BWD2_binormal(-0.8,rho2[i]), NA)
  values2[i] = ifelse(-0.4 >= 2 *abs(rho2[i]) - 1, BWD2_binormal(-0.4,rho2[i]), NA)
  values3[i] = ifelse(0 >= 2 *abs(rho2[i]) - 1, BWD2_binormal(0,rho2[i]), NA)
  values4[i] = ifelse(0.4 >= 2 *abs(rho2[i]) - 1, BWD2_binormal(0.4,rho2[i]), NA)
  values5[i] = ifelse(0.8 >= 2 *abs(rho2[i]) - 1, BWD2_binormal(0.8,rho2[i]), NA)
}

frame1 = as.data.frame(cbind(rho2[-which(is.na(values1))],
                             values1[-which(is.na(values1))]))
frame2= as.data.frame(cbind(rho2[-which(is.na(values2))],
                            values2[-which(is.na(values2))]))
frame3 = as.data.frame(cbind(rho2[-which(is.na(values3))],
                             values3[-which(is.na(values3))]))
frame4 = as.data.frame(cbind(rho2[-which(is.na(values4))],
                             values4[-which(is.na(values4))]))
frame5 = as.data.frame(cbind(rho2[-which(is.na(values5))],
                             values5[-which(is.na(values5))]))

# Make plot

g2 = ggplot() + geom_line(data = frame1,aes(x=V1,y=V2,lty = "a"),lwd = 1) +
  geom_line(data = frame2,aes(x=V1,y=V2,lty = "b"),lwd = 1) +
  geom_line(data = frame3,aes(x=V1,y=V2,lty = "c"),lwd = 1) +
  geom_line(data = frame4,aes(x=V1,y=V2,lty = "d"),lwd = 1) +
  geom_line(data = frame5,aes(x=V1,y=V2,lty = "e"),lwd = 1) +
  theme_bw() + xlab(expression(rho[2])) +
  ylab(expression("D"[2] * "((X"[1] * "," * "X"[2] * ");(X"[3] * "," * "X"[4] * "))")) +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=13),
        legend.position = "top",legend.text=element_text(size=13),
        legend.title=element_text(size=13),
        legend.key.width = unit(1.5, 'cm')) +
  scale_y_continuous(breaks = seq(0,0.7, by = 0.1),limits = c(0,0.7)) +
  scale_linetype_manual(values = c(1,2,3,4,5),
                        labels = c(-0.8,-0.4,0,0.4,0.8),
                        name = expression(rho[1] * "  "))

# cairo_pdf("Plot1.pdf", width = 8, height = 3) # Plot together and save

ggarrange(g1, g2, ncol=2, nrow=1, common.legend = TRUE, legend="top")
```




```
# dev.off()

create_R = function(rho1,rho2){

  # Function that constructs correlation matrix given rho1 and rho2

  R = cbind(c(1,rho1,rho2,rho2),c(rho1,1,rho2,rho2),c(rho2,rho2,1,rho1),
            c(rho2,rho2,rho1,1))

  return(R)
}

values1 = integer(length(rho2))
values2 = integer(length(rho2))
values3 = integer(length(rho2))
values4 = integer(length(rho2))
values5 = integer(length(rho2))

for(i in 1:length(rho2)){ # Asymptotic standard deviation of D1
                          # Filter such that rho1 >= 2 abs(rho2) - 1

  if(-0.8 >= (2 * abs(rho2[i]) - 1)){
    values1[i] = BWD1_Avar(create_R(-0.8,rho2[i]),c(2,2))
  }
  if(-0.4 >= (2 * abs(rho2[i]) - 1)){
    values2[i] = BWD1_Avar(create_R(-0.4,rho2[i]),c(2,2))
```

```
  }
  if(0 >= (2 * abs(rho2[i]) - 1)){
    values3[i] = BWD1_Avar(create_R(0,rho2[i]),c(2,2))
  }
  if(0.4 >= (2 * abs(rho2[i]) - 1)){
    values4[i] = BWD1_Avar(create_R(0.4,rho2[i]),c(2,2))
  }
  if(0.8 >= (2 * abs(rho2[i]) - 1)){
    values5[i] = BWD1_Avar(create_R(0.8,rho2[i]),c(2,2))
  }
}

frame1 = as.data.frame(cbind(rho2[-which(values1 == 0)],values1[-which(values1 == 0)]))
frame2 = as.data.frame(cbind(rho2[-which(values2 == 0)],values2[-which(values2 == 0)]))
frame3 = as.data.frame(cbind(rho2[-which(values3 == 0)],values3[-which(values3 == 0)]))
frame4 = as.data.frame(cbind(rho2[-which(values4 == 0)],values4[-which(values4 == 0)]))
frame5 = as.data.frame(cbind(rho2[-which(values5 == 0)],values5[-which(values5 == 0)]))


# Make plot

g1 = ggplot() + geom_line(data = frame1,aes(x=V1,y=V2,lty = "a"),lwd = 1) +
  geom_line(data = frame2,aes(x=V1,y=V2,lty = "b"),lwd = 1) +
  geom_line(data = frame3,aes(x=V1,y=V2,lty = "c"),lwd = 1) +
  geom_line(data = frame4,aes(x=V1,y=V2,lty = "d"),lwd = 1) +
  geom_line(data = frame5,aes(x=V1,y=V2,lty = "e"),lwd = 1) +
  theme_bw() + xlab(expression(rho[2])) + ylab(expression(zeta[1])) +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=13),
        legend.position = "top",legend.text=element_text(size=13),
        legend.title=element_text(size=13), legend.key.width = unit(1.5, 'cm')) +
  scale_y_continuous(breaks = seq(0,0.45, by = 0.1),limits = c(0,0.45)) +
  scale_linetype_manual(values = c(1,2,3,4,5), labels = c(-0.8,-0.4,0,0.4,0.8),
                        name = expression(rho[1] * "  "))

values1 = integer(length(rho2))
values2 = integer(length(rho2))
values3 = integer(length(rho2))
values4 = integer(length(rho2))
values5 = integer(length(rho2))

for(i in 1:length(rho2)){ # Asymptotic standard deviation of D2
                          # Filter such that rho1 >= 2 abs(rho2) - 1

  if(-0.8 >= (2 * abs(rho2[i]) - 1)){
    values1[i] = BWD2_Avar(create_R(-0.8,rho2[i]),c(2,2))
  }
  if(-0.4 >= (2 * abs(rho2[i]) - 1)){
    values2[i] = BWD2_Avar(create_R(-0.4,rho2[i]),c(2,2))
  }
  if(0 >= (2 * abs(rho2[i]) - 1)){
    values3[i] = BWD2_Avar(create_R(0,rho2[i]),c(2,2))
  }
  if(0.4 >= (2 * abs(rho2[i]) - 1)){
```

```
    values4[i] = BWD2_Avar(create_R(0.4,rho2[i]),c(2,2))
  }
  if(0.8 >= (2 * abs(rho2[i]) - 1)){
    values5[i] = BWD2_Avar(create_R(0.8,rho2[i]),c(2,2))
  }
}


frame1 = as.data.frame(cbind(rho2[-which(values1 == 0)],values1[-which(values1 == 0)]))
frame2 = as.data.frame(cbind(rho2[-which(values2 == 0)],values2[-which(values2 == 0)]))
frame3 = as.data.frame(cbind(rho2[-which(values3 == 0)],values3[-which(values3 == 0)]))
frame4 = as.data.frame(cbind(rho2[-which(values4 == 0)],values4[-which(values4 == 0)]))
frame5 = as.data.frame(cbind(rho2[-which(values5 == 0)],values5[-which(values5 == 0)]))


# Make plot

g2 = ggplot() + geom_line(data = frame1,aes(x=V1,y=V2,lty = "a"),lwd = 1) +
  geom_line(data = frame2,aes(x=V1,y=V2,lty = "b"),lwd = 1) +
  geom_line(data = frame3,aes(x=V1,y=V2,lty = "c"),lwd = 1) +
  geom_line(data = frame4,aes(x=V1,y=V2,lty = "d"),lwd = 1) +
  geom_line(data = frame5,aes(x=V1,y=V2,lty = "e"),lwd = 1) +
  theme_bw() + xlab(expression(rho[2])) + ylab(expression(zeta[2])) +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=13),
        legend.position = "top",legend.text=element_text(size=13),
        legend.title=element_text(size=13),
        legend.key.width = unit(1.5, 'cm')) +
  scale_y_continuous(breaks = seq(0,0.45, by = 0.1),limits = c(0,0.45)) +
  scale_linetype_manual(values = c(1,2,3,4,5),
                        labels = c(-0.8,-0.4,0,0.4,0.8),
                        name = expression(rho[1] * "  "))

# cairo_pdf("Plot2.pdf", width = 8, height = 3) # Plot together and save

ggarrange(g1, g2, ncol=2, nrow=1, common.legend = TRUE, legend="top")
```




```
# dev.off()
```

Next, we show the R code that was used for performing the simulations. Figure 3 of Example 4 and Figure 4 are also developed here.

Simulations

Finite-sample performance of the estimator of Section 3

We start with Section 5.1. Note that the R data file SD_emp.Rdata is loaded.

```
# Asymptotic normality of D1 (QQ-plots) in Gaussian copula case, 4 different settings

# Simulation for AR(1) model with rho = 0.25 and k = 2, d1 = d2 = 2,
# standard normal marginals

svalues = c(4) # Dimension(s), we have s = 4 in the paper
nvalues = c(50,200,1000,5000) # Sample sizes
M = 1000 # Number of Monte Carlo replications

true_val = matrix(0,length(nvalues),length(svalues)) # True values
est_val = array(0, dim = c(length(nvalues),length(svalues),M)) # Estimates

set.seed(123)

for(k in 1:length(nvalues)){
  n = nvalues[k]
```

```
  for(i in 1:length(svalues)){
    s = svalues[i]

    if(s < n){
      dim = rep(2,s/2)
      R = 0.25^(abs(matrix(1:s-1,nrow = s, ncol = s, byrow = TRUE) - (1:s-1))) # True R
      true = BWD1(R,dim) # True value of D1
      true_val[k,i] = true
      est = integer(M)

      for(j in 1:M){
        sample = mvtnorm::rmvnorm(n,rep(0,s),R,method = "chol") # Gaussian sample
        R_est = est_R(sample,1) # Estimated R
        BWD1_est = BWD1(R_est,dim) # Estimated D1
        Avar_est = BWD1_Avar(R_est,dim) # Estimated asymptotic standard deviation
        est[j] = (sqrt(n) * (BWD1_est - true))/Avar_est # Studentized estimate
      }
      est_val[k,i,] = est
    }
  }
}

# Make plots

g1 = ggplot() + geom_qq(aes(sample = est_val[1,1,])) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1.5, alpha = 0.8) +
  ggtitle("n = 50") + xlab("") + ylab("Setting 1") + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  annotate("text", x = 1, y = -5,vjust = 0, label = c("Med = 0.64"),col = c("blue"))

median(est_val[1,1,])
```

```
## [1] 0.643759
```

```
g2 = ggplot() + geom_qq(aes(sample = est_val[2,1,])) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1.5, alpha = 0.8) +
  ggtitle("n = 200") + xlab("") + ylab("") + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  annotate("text", x = 1, y = -5,vjust = 0, label = c("Med = 0.37"),col = c("blue"))

median(est_val[2,1,])
```

```
## [1] 0.3689585
```

```
g3 = ggplot() + geom_qq(aes(sample = est_val[3,1,])) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1.5, alpha = 0.8) +
  ggtitle("n = 1000") + xlab("") + ylab("") + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  annotate("text", x = 1, y = -5,vjust = 0, label = c("Med = 0.12"),col = c("blue"))

median(est_val[3,1,])
```

```
## [1] 0.1221971
```

```r
g4 = ggplot() + geom_qq(aes(sample = est_val[4,1,])) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1.5, alpha = 0.8) +
  ggtitle("n = 5000") + xlab("") + ylab("") + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  annotate("text", x = 1, y = -5,vjust = 0, label = c("Med = 0.06"),col = c("blue"))

median(est_val[4,1,])
```

```
## [1] 0.05657409
```

```r
# Simulation for AR(1) model with rho = 0.25 and k = 2, d1 = d2 = 2
# with different marginals

svalues = c(4)
nvalues = c(50,200,1000,5000)
M = 1000

true_val2 = matrix(0, length(nvalues),length(svalues))
est_val2 = array(0, dim = c(length(nvalues),length(svalues),M))

set.seed(123)

for(k in 1:length(nvalues)){
  n = nvalues[k]

  for(i in 1:length(svalues)){
    s = svalues[i]

    if(s < n){
      dim = rep(2,s/2)
      R = 0.25^(abs(matrix(1:s-1,nrow = s, ncol = s, byrow = TRUE) - (1:s-1)))
      true = BWD1(R,dim)
      true_val2[k,i] = true
      est = integer(M)

      for(j in 1:M){
        sample = pnorm(mvtnorm::rmvnorm(n,rep(0,s),R,method = "chol"))
        # Gaussian copula sample with various marginals
        sample = cbind(qt(sample[,1],3),qexp(sample[,2],1),qbeta(sample[,3],2,2),
                       qf(sample[,4],2,6))
        R_est = est_R(sample,1)
        BWD1_est = BWD1(R_est,dim)
        Avar_est = BWD1_Avar(R_est,dim)
        est[j] = (sqrt(n) * (BWD1_est - true))/Avar_est
      }
      est_val2[k,i,] = est
    }
  }
}

# Make plots

g5 = ggplot() + geom_qq(aes(sample = est_val2[1,1,])) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1.5, alpha = 0.8) +
```

```
  ggtitle("") + xlab("") + ylab("Setting 2") + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  annotate("text", x = 1, y = -5,vjust = 0, label = c("Med = 0.64"),col = c("blue"))

median(est_val2[1,1,])
```

```
## [1] 0.643759
```

```
g6 = ggplot() + geom_qq(aes(sample = est_val2[2,1,])) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1.5, alpha = 0.8) +
  ggtitle("") + xlab("") + ylab("") + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  annotate("text", x = 1, y = -5,vjust = 0, label = c("Med = 0.37"),col = c("blue"))

median(est_val2[2,1,])
```

```
## [1] 0.3689585
```

```
g7 = ggplot() + geom_qq(aes(sample = est_val2[3,1,])) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1.5, alpha = 0.8) +
  ggtitle("") + xlab("") + ylab("") + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  annotate("text", x = 1, y = -5,vjust = 0, label = c("Med = 0.12"),col = c("blue"))

median(est_val2[3,1,])
```

```
## [1] 0.1221971
```

```
g8 = ggplot() + geom_qq(aes(sample = est_val2[4,1,])) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1.5, alpha = 0.8) +
  ggtitle("") + xlab("") + ylab("") + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  annotate("text", x = 1, y = -5,vjust = 0, label = c("Med = 0.06"),col = c("blue"))

median(est_val2[4,1,])
```

```
## [1] 0.05657409
```

```
# Simulation for AR(1) model with rho = 0.8 and k = 2, d1 = d2 = 2
# standard normal marginals

svalues = c(4)
nvalues = c(50,200,1000,5000)
M = 1000

true_val3 = matrix(0, length(nvalues),length(svalues))
est_val3 = array(0, dim = c(length(nvalues),length(svalues),M))

set.seed(123)

for(k in 1:length(nvalues)){
  n = nvalues[k]

  for(i in 1:length(svalues)){
    s = svalues[i]
```

```
    if(s < n){
      dim = rep(2,s/2)
      R = 0.8^(abs(matrix(1:s-1,nrow = s, ncol = s, byrow = TRUE) - (1:s-1)))
      true = BWD1(R,dim)
      true_val3[k,i] = true
      est = integer(M)

      for(j in 1:M){
        sample = mvtnorm::rmvnorm(n,rep(0,s),R,method = "chol")
        R_est = est_R(sample,1)
        BWD1_est = BWD1(R_est,dim)
        Avar_est = BWD1_Avar(R_est,dim)
        est[j] = (sqrt(n) * (BWD1_est - true))/Avar_est
      }
      est_val3[k,i,] = est
    }
  }
}

# Make plots

g9 = ggplot() + geom_qq(aes(sample = est_val3[1,1,])) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1.5, alpha = 0.8) +
  ggtitle("") + xlab("") + ylab("Setting 3") + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  annotate("text", x = 1, y = -5,vjust = 0, label = c("Med = -0.04"),col = c("blue"))

median(est_val3[1,1,])
```

```
## [1] -0.03562429
```

```
g10 = ggplot() + geom_qq(aes(sample = est_val3[2,1,])) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1.5, alpha = 0.8) +
  ggtitle("") + xlab("") + ylab("") + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  annotate("text", x = 1, y = -5,vjust = 0, label = c("Med = 0.00"),col = c("blue"))

median(est_val3[2,1,])
```

```
## [1] -0.004730001
```

```
g11 = ggplot() + geom_qq(aes(sample = est_val3[3,1,])) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1.5, alpha = 0.8) +
  ggtitle("") + xlab("") + ylab("") + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  annotate("text", x = 1, y = -5,vjust = 0, label = c("Med = -0.05"),col = c("blue"))

median(est_val3[3,1,])
```

```
## [1] -0.05182635
```

```
g12 = ggplot() + geom_qq(aes(sample = est_val3[4,1,])) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1.5, alpha = 0.8) +
  ggtitle("") + xlab("") + ylab("") + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  annotate("text", x = 1, y = -5,vjust = 0, label = c("Med = 0.08"),col = c("blue"))
```

```r
median(est_val3[4,1,])
```

```
## [1] 0.07703065
```

```r
# Simulation for equicorrelated model with rho = 0.5 and
# k = 5, d1 = 4, d2 = 5, d3 = 3,d4 = 1, d5 = 2
# standard normal marginals

s = 15
dim = c(4,5,3,1,2)
nvalues = c(50,200,1000,5000)
M = 1000

true_val4 = integer(length(nvalues))
est_val4 = matrix(0,length(nvalues),M)

set.seed(123)

for(k in 1:length(nvalues)){
  n = nvalues[k]
  R = 0.5 * diag(s) + 0.5 * matrix(1,s,s)
  true = BWD1(R,dim)
  true_val4[k] = true
  est = integer(M)

  for(j in 1:M){
    sample = mvtnorm::rmvnorm(n,rep(0,s),R,method = "chol")
    sorted = sort_OT(sample,dim) # Sort such that dim is in ascending order
    sample_sorted = sorted$sample
    dim_sorted = sorted$dim
    R_est = est_R(sample_sorted,1)
    BWD1_est = BWD1(R_est,dim_sorted)
    Avar_est = BWD1_Avar(R_est,dim_sorted)
    est[j] = (sqrt(n) * (BWD1_est - true))/Avar_est
  }
  est_val4[k,] = est
}

# Make plots

g13 = ggplot() + geom_qq(aes(sample = est_val4[1,])) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1.5, alpha = 0.8) +
  ggtitle("") + xlab("") + ylab("Setting 4") + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  annotate("text", x = 1, y = -5,vjust = 0, label = c("Med = 1.45"),col = c("blue"))

median(est_val4[1,])
```

```
## [1] 1.445519
```

```r
g14 = ggplot() + geom_qq(aes(sample = est_val4[2,])) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1.5, alpha = 0.8) +
  ggtitle("") + xlab("") + ylab("") + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  annotate("text", x = 1, y = -5,vjust = 0, label = c("Med = 0.54"),col = c("blue"))
```

```r
median(est_val4[2,])
```

```
## [1] 0.5402598
```

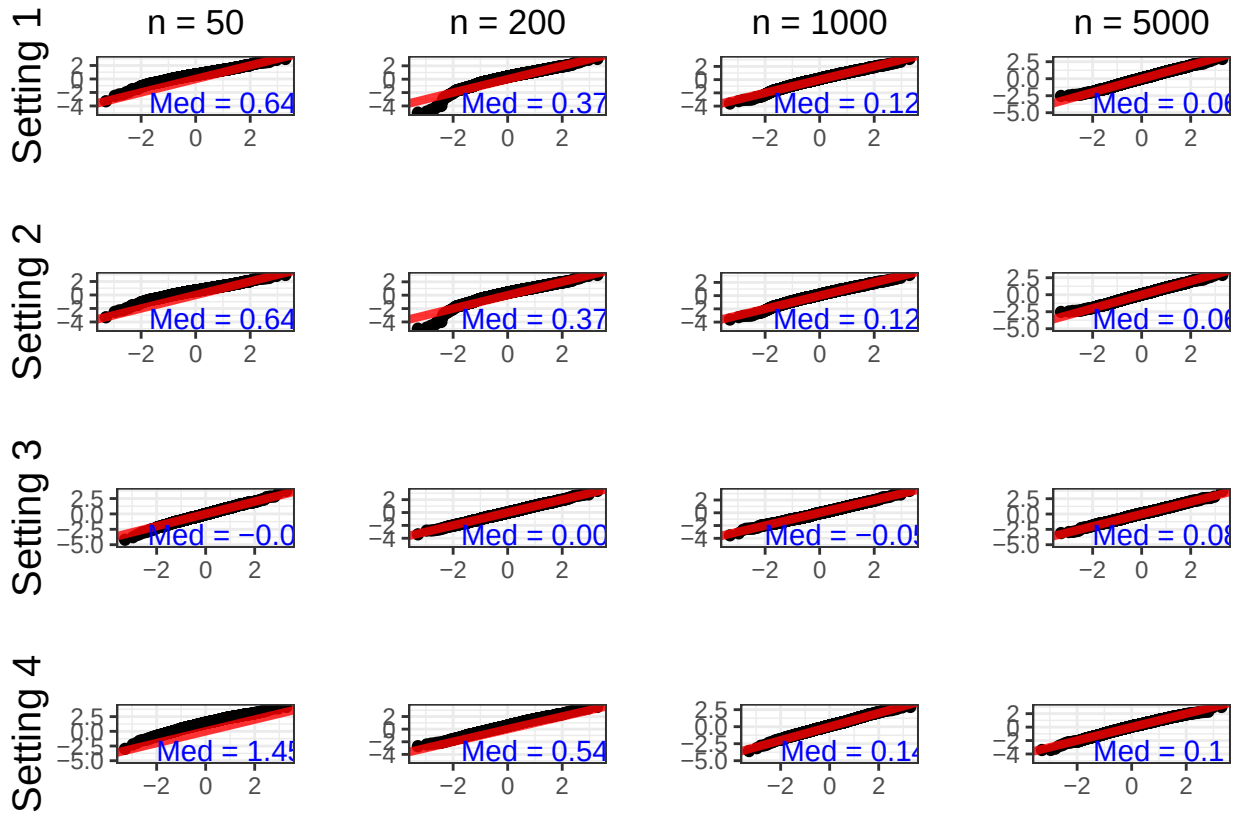
```r
g15 = ggplot() + geom_qq(aes(sample = est_val4[3,])) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1.5, alpha = 0.8) +
  ggtitle("") + xlab("") + ylab("") + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  annotate("text", x = 1, y = -5,vjust = 0, label = c("Med = 0.14"),col = c("blue"))

median(est_val4[3,])
```

```
## [1] 0.1426558
```

```r
g16 = ggplot() + geom_qq(aes(sample = est_val4[4,])) +
  geom_abline(intercept = 0, slope = 1, color = "red", size = 1.5, alpha = 0.8) +
  ggtitle("") + xlab("") + ylab("") + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=15)) +
  annotate("text", x = 1, y = -5,vjust = 0, label = c("Med = 0.1"),col = c("blue"))

median(est_val4[4,])
```

```
## [1] 0.1002728
```

```r
# Put all plots together

ggarrange(g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g16, nrow = 4, ncol = 4)
```



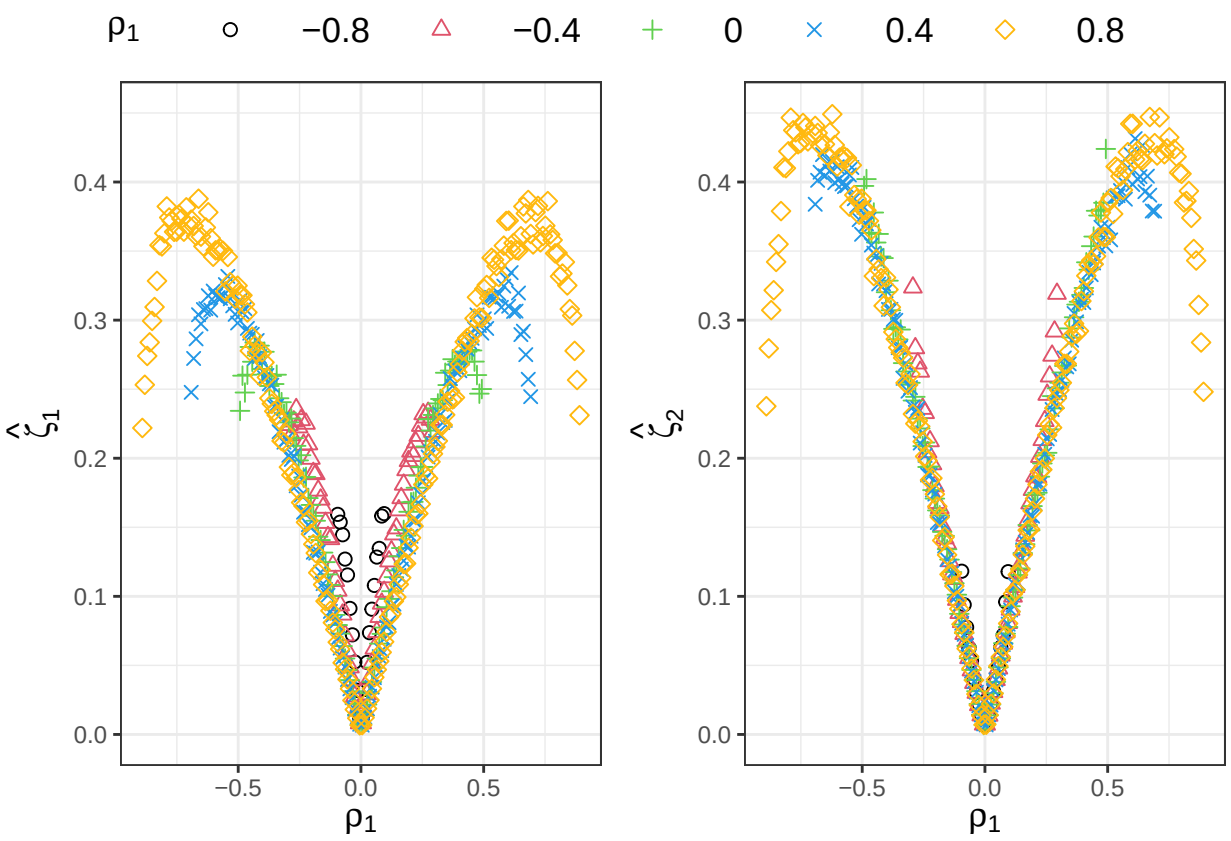
```r
# Empirical verification for asymptotic standard deviation of D1 (or D2)
# in Example 3

rho2 = seq(-0.99,0.99,len = 200) # Grid of rho2 values

values1 = integer(length(rho2))
values2 = integer(length(rho2))
values3 = integer(length(rho2))
values4 = integer(length(rho2))
values5 = integer(length(rho2))

n = 10000 # Sample size
M = 1000  # Number of Monte Carlo Replications
```

```r
set.seed(123)

for(i in 1:length(rho2)){

    if(-0.8 >= (2 * abs(rho2[i]) - 1)){ # If rho1 = -0.8 and rho1 >= 2 |rho2| - 1
    R = create_R(-0.8,rho2[i]) # True R
    est = integer(M) # Dependence estimates

    for(l in 1:M){
      sample = mvtnorm::rmvnorm(n,rep(0,4),R, method = "chol") # Gaussian sample
      est[l] = BWD1(est_R(sample,1),c(2,2)) # Estimated D1, or BWD2 for D2
    }
    values1[i] = sqrt(n) * sqrt(var(est)) # Approximated asymptotic standard deviation
    }

  if(-0.4 >= (2 * abs(rho2[i]) - 1)){ # If rho1 = -0.4
    R = create_R(-0.4,rho2[i])
    est = integer(M)

    for(l in 1:M){
      sample = mvtnorm::rmvnorm(n,rep(0,4),R, method = "chol")
      est[l] = BWD1(est_R(sample,1),c(2,2))
    }
    values2[i] = sqrt(n) * sqrt(var(est))
  }

  if(0 >= (2 * abs(rho2[i]) - 1)){# If rho1 = 0
    R = create_R(0,rho2[i])
    est = integer(M)

    for(l in 1:M){
      sample = mvtnorm::rmvnorm(n,rep(0,4),R,method = "chol")
      est[l] = BWD1(est_R(sample,1),c(2,2))
    }
    values3[i] = sqrt(n) * sqrt(var(est))
  }

  if(0.4 >= (2 * abs(rho2[i]) - 1)){ # If rho1 = 0.4
    R = create_R(0.4,rho2[i])
    est = integer(M)
```

```
    for(l in 1:M){
      sample = mvtnorm::rmvnorm(n,rep(0,4),R,method = "chol")
      est[l] = BWD1(est_R(sample,1),c(2,2))
    }
    values4[i] = sqrt(n) * sqrt(var(est))
  }

  if(0.8 >= (2 * abs(rho2[i]) - 1)){ # If rho1 = 0.8
    R = create_R(0.8,rho2[i])
    est = integer(M)

    for(l in 1:M){
      sample = mvtnorm::rmvnorm(n,rep(0,4),R,method = "chol")
      est[l] = BWD1(est_R(sample,1),c(2,2))
    }
    values5[i] = sqrt(n) * sqrt(var(est))
  }
}

# Save variables

# VTS = list("SD_emp1" = 0, "SD_emp2" = 0)
# values = cbind(values1,values2,values3,values4,values5)
# VTS$SD_emp1 = values (or VTS$SD_emp2 = values)
# save(VTS, file = "SD_emp.Rdata")
```

```
# Load variables

load("SD_emp.Rdata")

values = VTS$SD_emp1 # For D1

values1 = values[,1]
values2 = values[,2]
values3 = values[,3]
values4 = values[,4]
values5 = values[,5]

frame1 = as.data.frame(cbind(rho2[-which(values1 == 0)],values1[-which(values1 == 0)]))
frame2 = as.data.frame(cbind(rho2[-which(values2 == 0)],values2[-which(values2 == 0)]))
frame3 = as.data.frame(cbind(rho2[-which(values3 == 0)],values3[-which(values3 == 0)]))
frame4 = as.data.frame(cbind(rho2[-which(values4 == 0)],values4[-which(values4 == 0)]))
frame5 = as.data.frame(cbind(rho2[-which(values5 == 0)],values5[-which(values5 == 0)]))

# Make plot for D1

g1 = ggplot() + geom_point(data = frame1,aes(x=V1,y=V2,col = "a"),pch = 1, size = 2) +
  geom_point(data = frame2,aes(x=V1,y=V2,col = "b"),pch = 2, size = 2) +
  geom_point(data = frame3,aes(x=V1,y=V2,col = "c"),pch = 3, size = 2) +
  geom_point(data = frame4,aes(x=V1,y=V2,col = "d"),pch = 4, size = 2) +
  geom_point(data = frame5,aes(x=V1,y=V2,col = "e"),pch = 5, size = 2) +
  theme_bw() + xlab(expression(rho[1])) + ylab(expression(hat(zeta)[1])) +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=13),
```

```
        legend.position = "top",legend.text=element_text(size=13),
        legend.title=element_text(size=13),
        legend.key.width = unit(1.5, 'cm')) +
  scale_y_continuous(breaks = seq(0,0.45, by = 0.1),limits = c(0,0.45)) +
  scale_colour_manual(values = c(1,2,3,4,"darkgoldenrod1"),
                      labels = c(-0.8,-0.4,0,0.4,0.8),
                      name = expression(rho[1] * "  ")) +
  guides(color = guide_legend(override.aes=list(shape = c(1,2,3,4,5))))

values = VTS$SD_emp2 # For D2

values1 = values[,1]
values2 = values[,2]
values3 = values[,3]
values4 = values[,4]
values5 = values[,5]

frame1 = as.data.frame(cbind(rho2[-which(values1 == 0)],values1[-which(values1 == 0)]))
frame2 = as.data.frame(cbind(rho2[-which(values2 == 0)],values2[-which(values2 == 0)]))
frame3 = as.data.frame(cbind(rho2[-which(values3 == 0)],values3[-which(values3 == 0)]))
frame4 = as.data.frame(cbind(rho2[-which(values4 == 0)],values4[-which(values4 == 0)]))
frame5 = as.data.frame(cbind(rho2[-which(values5 == 0)],values5[-which(values5 == 0)]))

# Make plot for D2

g2 = ggplot() + geom_point(data = frame1,aes(x=V1,y=V2,col = "a"),pch = 1, size = 2) +
  geom_point(data = frame2,aes(x=V1,y=V2,col = "b"),pch = 2, size = 2) +
  geom_point(data = frame3,aes(x=V1,y=V2,col = "c"),pch = 3, size = 2) +
  geom_point(data = frame4,aes(x=V1,y=V2,col = "d"),pch = 4, size = 2) +
  geom_point(data = frame5,aes(x=V1,y=V2,col = "e"),pch = 5, size = 2) +
  theme_bw() + xlab(expression(rho[1])) + ylab(expression(hat(zeta)[2])) +
  theme(plot.title = element_text(hjust=0.5),axis.title = element_text(size=13),
        legend.position = "top",legend.text=element_text(size=13),
        legend.title=element_text(size=13),
        legend.key.width = unit(1.5, 'cm')) +
  scale_y_continuous(breaks = seq(0,0.45, by = 0.1),limits = c(0,0.45)) +
  scale_colour_manual(values = c(1,2,3,4,"darkgoldenrod1"),
                      labels = c(-0.8,-0.4,0,0.4,0.8),
                      name = expression(rho[1] * "  ")) +
  guides(color = guide_legend(override.aes=list(shape = c(1,2,3,4,5))))

# cairo_pdf("Plot4.pdf", width = 10, height = 5) # Plot together and save

ggarrange(g1, g2, ncol = 2, nrow = 1, common.legend = TRUE, legend = "top")
```




```
# dev.off()
```

Penalization techniques

Next, we go to Section 5.2. Note that the R data files RidgeSimNormalized.Rdata and LassoSimNew.Rdata are loaded.

```
# Simulation for Ridge estimation

# VTS = list("MI" = list("Setting1" = list("NonPenal" = list("bias" = 0, "sd" = 0),
#                                          "Penal" = list("bias" = 0, "sd" = 0, "omegas" = 0)),
#                        "Setting2" = list("NonPenal" = list("bias" = 0, "sd" = 0),
#                                          "Penal" = list("bias" = 0, "sd" = 0, "omegas" = 0))),
#            "Hel" = list("Setting1" = list("NonPenal" = list("bias" = 0, "sd" = 0),
#                                           "Penal" = list("bias" = 0, "sd" = 0)),
#                         "Setting2" = list("NonPenal" = list("bias" = 0, "sd" = 0),
#                                           "Penal" = list("bias" = 0, "sd" = 0))),
#            "BWD1" = list("Setting1" = list("NonPenal" = list("bias" = 0, "sd" = 0),
#                                            "Penal" = list("bias" = 0, "sd" = 0)),
#                          "Setting2" = list("NonPenal" = list("bias" = 0, "sd" = 0),
#                                            "Penal" = list("bias" = 0, "sd" = 0))),
#            "BWD2" = list("Setting1" = list("NonPenal" = list("bias" = 0, "sd" = 0),
#                                            "Penal" = list("bias" = 0, "sd" = 0)),
#                          "Setting2" = list("NonPenal" = list("bias" = 0, "sd" = 0),
#                                            "Penal" = list("bias" = 0, "sd" = 0))))

normalize = function(x){sqrt(1-exp(-2*x))} # Normalization for mutual information
```

```
# Design 1, no penalization

svalues = seq(4,98,by = 2) # Dimensions
nvalues = c(50,100,500) # Sample sizes
M = 1000 # Number of Monte Carlo replications

true_val = array(0, dim = c(4,length(nvalues),length(svalues))) # True values
est_val = array(0, dim = c(4,length(nvalues),length(svalues),M)) # Estimated values
```

```
set.seed(123)

for(k in 1:length(nvalues)){
  n = nvalues[k]

  for(i in 1:length(svalues)){
    s = svalues[i]

    if(s < n){
      dim = rep(2,s/2)
      R = 0.5^(abs(matrix(1:s-1,nrow = s, ncol = s, byrow = TRUE) - (1:s-1)))
      true_val[1,k,i] = normalize(MI_normal(R,dim)) ; true_val[2,k,i] = Hel_normal(R,dim)
      true_val[3,k,i] = BWD1(R,dim) ; true_val[4,k,i] = BWD2(R,dim)
      est = matrix(0,4,M)

      for(j in 1:M){
        sample = rmvnorm(n,rep(0,s),R, method = "chol")
        R_est = est_R(sample,1) # Estimated R, no penalization
        est[1,j] = normalize(MI_normal(R_est,dim)) ; est[2,j] = Hel_normal(R_est,dim)
        est[3,j] = BWD1(R_est,dim) ; est[4,j] = BWD2(R_est,dim)
      }
      est_val[1,k,i,] = est[1,] ; est_val[2,k,i,] = est[2,]
      est_val[3,k,i,] = est[3,] ; est_val[4,k,i,] = est[4,]
    }
  }
}

# Compute Monte Carlo biases and standard deviations

biases = array(0, dim = c(4,length(nvalues),length(svalues)))
sds = array(0, dim = c(4,length(nvalues),length(svalues)))

for(k in 1:length(nvalues)){
  for(i in 1:length(svalues)){
    if(svalues[i] < nvalues[k]){
      biases[1,k,i] = mean(est_val[1,k,i,]) - true_val[1,k,i]
      biases[2,k,i] = mean(est_val[2,k,i,]) - true_val[2,k,i]
      biases[3,k,i] = mean(est_val[3,k,i,]) - true_val[3,k,i]
      biases[4,k,i] = mean(est_val[4,k,i,]) - true_val[4,k,i]
      sds[1,k,i] = sd(est_val[1,k,i,])
      sds[2,k,i] = sd(est_val[2,k,i,])
      sds[3,k,i] = sd(est_val[3,k,i,])
      sds[4,k,i] = sd(est_val[4,k,i,])
    }
```

```
  }
}

VTS$MI$Setting1$NonPenal$bias = biases[1,,]
VTS$Hel$Setting1$NonPenal$bias = biases[2,,]
VTS$BWD1$Setting1$NonPenal$bias = biases[3,,]
VTS$BWD2$Setting1$NonPenal$bias = biases[4,,]

VTS$MI$Setting1$NonPenal$sd = sds[1,,]
VTS$Hel$Setting1$NonPenal$sd = sds[2,,]
VTS$BWD1$Setting1$NonPenal$sd = sds[3,,]
VTS$BWD2$Setting1$NonPenal$sd = sds[4,,]

# save(VTS, file = "RidgeSimNormalized.Rdata")  # Save data
```

```
# Design 2, no penalization

svalues = seq(4,98,by = 2)
nvalues = c(50,100,500)
M = 1000

true_val = array(0, dim = c(4,length(nvalues),length(svalues)))
est_val = array(0, dim = c(4,length(nvalues),length(svalues),M))
```

```
set.seed(123)

for(k in 1:length(nvalues)){
  n = nvalues[k]

  for(i in 1:length(svalues)){
    s = svalues[i]

    if(s < n){
      dim = c(s/2,s/2)
      R = 0.5^(abs(matrix(1:s-1,nrow = s, ncol = s, byrow = TRUE) - (1:s-1)))
      true_val[1,k,i] = normalize(MI_normal(R,dim)) ; true_val[2,k,i] = Hel_normal(R,dim)
      true_val[3,k,i] = BWD1(R,dim) ; true_val[4,k,i] = BWD2(R,dim)
      est = matrix(0,4,M)

      for(j in 1:M){
        sample = rmvnorm(n,rep(0,s),R, method = "chol")
        R_est = est_R(sample,1)
        est[1,j] = normalize(MI_normal(R_est,dim)) ; est[2,j] = Hel_normal(R_est,dim)
        est[3,j] = BWD1(R_est,dim) ; est[4,j] = BWD2(R_est,dim)
      }
      est_val[1,k,i,] = est[1,] ; est_val[2,k,i,] = est[2,]
      est_val[3,k,i,] = est[3,] ; est_val[4,k,i,] = est[4,]
    }
  }
}

# Compute Monte Carlo biases and standard deviations

biases = array(0, dim = c(4,length(nvalues),length(svalues)))
```

```
sds = array(0, dim = c(4,length(nvalues),length(svalues)))

for(k in 1:length(nvalues)){
  for(i in 1:length(svalues)){
    if(svalues[i] < nvalues[k]){
      biases[1,k,i] = mean(est_val[1,k,i,]) - true_val[1,k,i]
      biases[2,k,i] = mean(est_val[2,k,i,]) - true_val[2,k,i]
      biases[3,k,i] = mean(est_val[3,k,i,]) - true_val[3,k,i]
      biases[4,k,i] = mean(est_val[4,k,i,]) - true_val[4,k,i]
      sds[1,k,i] = sd(est_val[1,k,i,])
      sds[2,k,i] = sd(est_val[2,k,i,])
      sds[3,k,i] = sd(est_val[3,k,i,])
      sds[4,k,i] = sd(est_val[4,k,i,])
    }
  }
}

VTS$MI$Setting2$NonPenal$bias = biases[1,,]
VTS$Hel$Setting2$NonPenal$bias = biases[2,,]
VTS$BWD1$Setting2$NonPenal$bias = biases[3,,]
VTS$BWD2$Setting2$NonPenal$bias = biases[4,,]

VTS$MI$Setting2$NonPenal$sd = sds[1,,]
VTS$Hel$Setting2$NonPenal$sd = sds[2,,]
VTS$BWD1$Setting2$NonPenal$sd = sds[3,,]
VTS$BWD2$Setting2$NonPenal$sd = sds[4,,]

# save(VTS, file = "RidgeSimNormalized.Rdata") # Save data
```

```
# Ridge penalization

# Design 1, 5-fold cross validation for omega

svalues = seq(4,98,by = 2)
nvalues = c(50,100,500)
M = 1000

omegas = seq(0.01,0.999,len = 50) # Possible penalty parameter values
est_val = array(0, dim = c(4,length(nvalues),length(svalues),M))
est_omegas = array(0, dim = c(length(nvalues),length(svalues),M))
# Chosen penalty parameters
true_val = array(0, dim = c(4,length(nvalues),length(svalues)))
```

```
set.seed(123)

for(k in 1:length(nvalues)){
  n = nvalues[k]

  for(i in 1:length(svalues)){
    s = svalues[i]
    dim = rep(2,s/2)
    R = 0.5^(abs(matrix(1:s-1,nrow = s, ncol = s, byrow = TRUE) - (1:s-1)))
    true_val[1,k,i] = normalize(MI_normal(R,dim)) ; true_val[2,k,i] = Hel_normal(R,dim)
    true_val[3,k,i] = BWD1(R,dim) ; true_val[4,k,i] = BWD2(R,dim)
```

```
    est = matrix(0,4,M)

    for(j in 1:M){
      sample = rmvnorm(n,rep(0,s),R)
      omega = CVomega(sample,omegas,5) # 5-fold cross-validation
      R_est = est_R(sample,omega) # Estimated R, Ridge penalization
      est[1,j] = normalize(MI_normal(R_est,dim)) ; est[2,j] = Hel_normal(R_est,dim)
      est[3,j] = BWD1(R_est,dim) ; est[4,j] = BWD2(R_est,dim)
      est_omegas[k,i,j] = omega
    }
    est_val[1,k,i,] = est[1,] ; est_val[2,k,i,] = est[2,]
    est_val[3,k,i,] = est[3,] ; est_val[4,k,i,] = est[4,]
  }
}

# Compute Monte Carlo biases and standard deviations

biases = array(0, dim = c(4,length(nvalues),length(svalues)))
sds = array(0, dim = c(4,length(nvalues),length(svalues)))

for(k in 1:length(nvalues)){
  for(i in 1:length(svalues)){
      biases[1,k,i] = mean(est_val[1,k,i,]) - true_val[1,k,i]
      biases[2,k,i] = mean(est_val[2,k,i,]) - true_val[2,k,i]
      biases[3,k,i] = mean(est_val[3,k,i,]) - true_val[3,k,i]
      biases[4,k,i] = mean(est_val[4,k,i,]) - true_val[4,k,i]
      sds[1,k,i] = sd(est_val[1,k,i,])
      sds[2,k,i] = sd(est_val[2,k,i,])
      sds[3,k,i] = sd(est_val[3,k,i,])
      sds[4,k,i] = sd(est_val[4,k,i,])
  }
}

VTS$MI$Setting1$Penal$bias = biases[1,,]
VTS$Hel$Setting1$Penal$bias = biases[2,,]
VTS$BWD1$Setting1$Penal$bias = biases[3,,]
VTS$BWD2$Setting1$Penal$bias = biases[4,,]

VTS$MI$Setting1$Penal$sd = sds[1,,]
VTS$Hel$Setting1$Penal$sd = sds[2,,]
VTS$BWD1$Setting1$Penal$sd = sds[3,,]
VTS$BWD2$Setting1$Penal$sd = sds[4,,]

VTS$MI$Setting1$Penal$omegas = est_omegas

# save(VTS, file = "RidgeSimNormalized.Rdata") # Save data
```

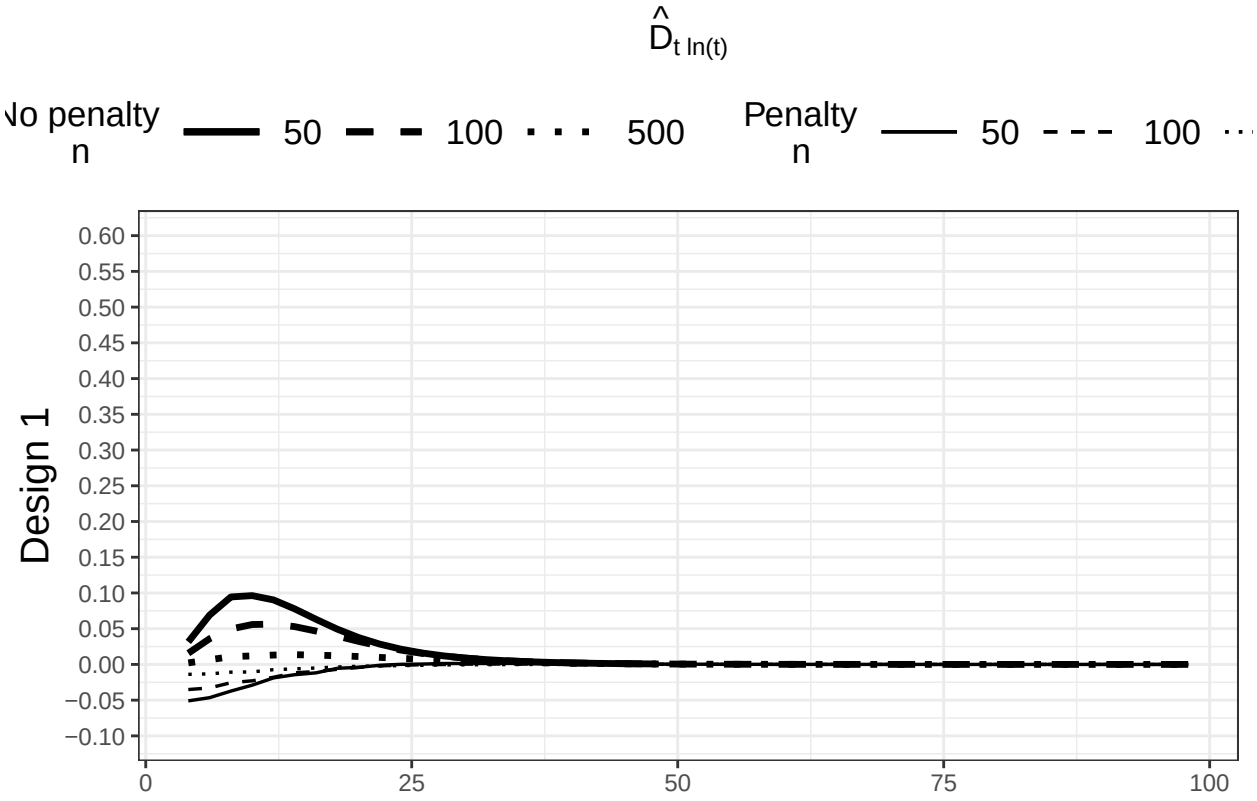
```
# Design 2, 5-fold cross validation for omega

svalues = seq(4,98,by = 2)
nvalues = c(50,100,500)
M = 1000

omegas = seq(0.01,0.999,len = 50)
```

```
est_val = array(0, dim = c(4,length(nvalues),length(svalues),M))
est_omegas = array(0, dim = c(length(nvalues),length(svalues),M))
true_val = array(0, dim = c(4,length(nvalues),length(svalues)))
```

```
set.seed(123)

for(k in 1:length(nvalues)){

  n = nvalues[k]
  for(i in 1:length(svalues)){

    s = svalues[i]
    dim = c(s/2,s/2)
    R = 0.5^(abs(matrix(1:s-1,nrow = s, ncol = s, byrow = TRUE) - (1:s-1)))
    true_val[1,k,i] = normalize(MI_normal(R,dim)) ; true_val[2,k,i] = Hel_normal(R,dim)
    true_val[3,k,i] = BWD1(R,dim) ; true_val[4,k,i] = BWD2(R,dim)
    est = matrix(0,4,M)

    for(j in 1:M){
      sample = rmvnorm(n,rep(0,s),R)
      omega = CVomega(sample,omegas,5)
      R_est = est_R(sample,omega)
      est[1,j] = normalize(MI_normal(R_est,dim)) ; est[2,j] = Hel_normal(R_est,dim)
      est[3,j] = BWD1(R_est,dim) ; est[4,j] = BWD2(R_est,dim)
      est_omegas[k,i,j] = omega
    }
    est_val[1,k,i,] = est[1,] ; est_val[2,k,i,] = est[2,]
    est_val[3,k,i,] = est[3,] ; est_val[4,k,i,] = est[4,]
  }
}

# Compute Monte Carlo biases and standard deviations

biases = array(0, dim = c(4,length(nvalues),length(svalues)))
sds = array(0, dim = c(4,length(nvalues),length(svalues)))

for(k in 1:length(nvalues)){
  for(i in 1:length(svalues)){
      biases[1,k,i] = mean(est_val[1,k,i,]) - true_val[1,k,i]
      biases[2,k,i] = mean(est_val[2,k,i,]) - true_val[2,k,i]
      biases[3,k,i] = mean(est_val[3,k,i,]) - true_val[3,k,i]
      biases[4,k,i] = mean(est_val[4,k,i,]) - true_val[4,k,i]
      sds[1,k,i] = sd(est_val[1,k,i,])
      sds[2,k,i] = sd(est_val[2,k,i,])
      sds[3,k,i] = sd(est_val[3,k,i,])
      sds[4,k,i] = sd(est_val[4,k,i,])
  }
}

VTS$MI$Setting2$Penal$bias = biases[1,,]
VTS$Hel$Setting2$Penal$bias = biases[2,,]
VTS$BWD1$Setting2$Penal$bias = biases[3,,]
VTS$BWD2$Setting2$Penal$bias = biases[4,,]
```

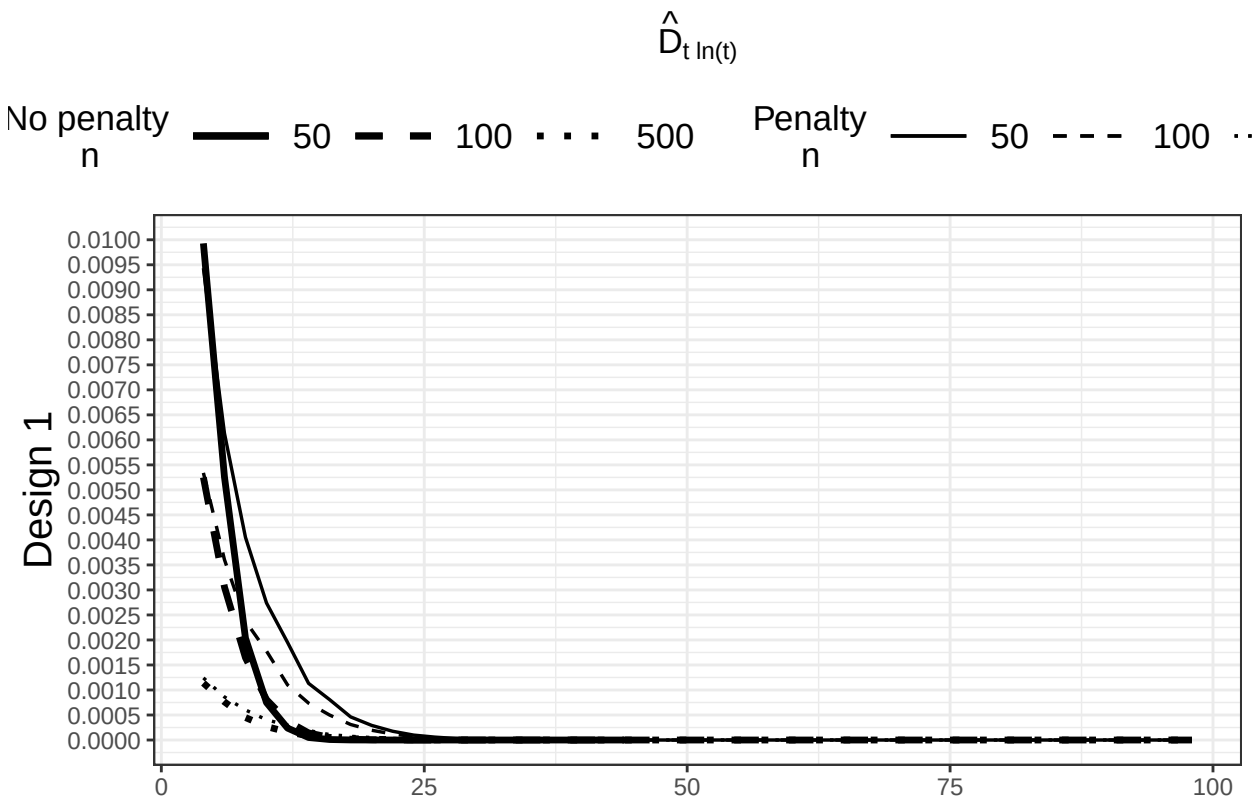
```
VTS$MI$Setting2$Penal$sd = sds[1,,]
VTS$Hel$Setting2$Penal$sd = sds[2,,]
VTS$BWD1$Setting2$Penal$sd = sds[3,,]
VTS$BWD2$Setting2$Penal$sd = sds[4,,]

VTS$MI$Setting2$Penal$omegas = est_omegas

# save(VTS, file = "RidgeSimNormalized.Rdata") # Save data
```

```
#  Bias Plots (similar for Setting 2 and Hel_normal, BWD1, BWD2)

load("RidgeSimNormalized.Rdata") # Load data

errors1 = VTS$MI$Setting1$NonPenal$bias
errors2 = VTS$MI$Setting1$Penal$bias

frame = as.data.frame(cbind(svalues,t(errors1),t(errors2)))
frame$V2[which(frame$V2 == 0)] = rep(NA,length(which(frame$V2 == 0)))

# Make plot

g1 = ggplot(frame) + geom_line(aes(x = svalues, y = V2, lwd = "a",lty = "a")) +
  geom_line(aes(x = svalues, y = V5, lwd = "b",lty = "a")) +
  geom_line(aes(x = svalues, y = V3, lwd = "a",lty = "b")) +
  geom_line(aes(x = svalues, y = V6, lwd = "b",lty = "b")) +
  geom_line(aes(x = svalues, y = V4, lwd = "a",lty = "c")) +
  geom_line(aes(x = svalues, y = V7, lwd = "c",lty = "c")) +
  scale_y_continuous(breaks = seq(-0.1,0.6, by = 0.05),limits = c(-0.1,0.6)) +
  xlab("") + ylab("Design 1") + ggtitle(expression(hat(D)["t ln(t)"])) +
  theme_bw() + theme(plot.title = element_text(hjust=0.5),
                     axis.title = element_text(size=15)) +
  theme(legend.position = "top", legend.text=element_text(size=13),
        legend.title=element_text(size=13),
        legend.key.width = unit(1.25, 'cm'),legend.direction = "horizontal",
        legend.box = "horizontal",
        legend.key = element_rect(fill = "transparent")) +
  scale_linewidth_manual(values = c(1.2,0.6,0.6), labels = c("50","100","500"),
                         name = "No penalty\n        n") +
  scale_linetype_manual(values=c("solid", "dashed","dotted"),
                        labels = c("50","100","500"),
                        name = "Penalty\n     n") +
  guides(linetype = guide_legend(override.aes = list(linetype = c("solid","dashed",
                                                                  "dotted"),
                                                     linewidth = c(0.6,0.6,0.6),
                                                     labels = c("50","100","500"))),
         linewidth = guide_legend(override.aes = list(linewidth = c(1.2,1.2,1.2),
                                                      linetype = c("solid","dashed",
                                                                   "dotted"),
                                                      labels = c("50","100","500"))))

g1
```



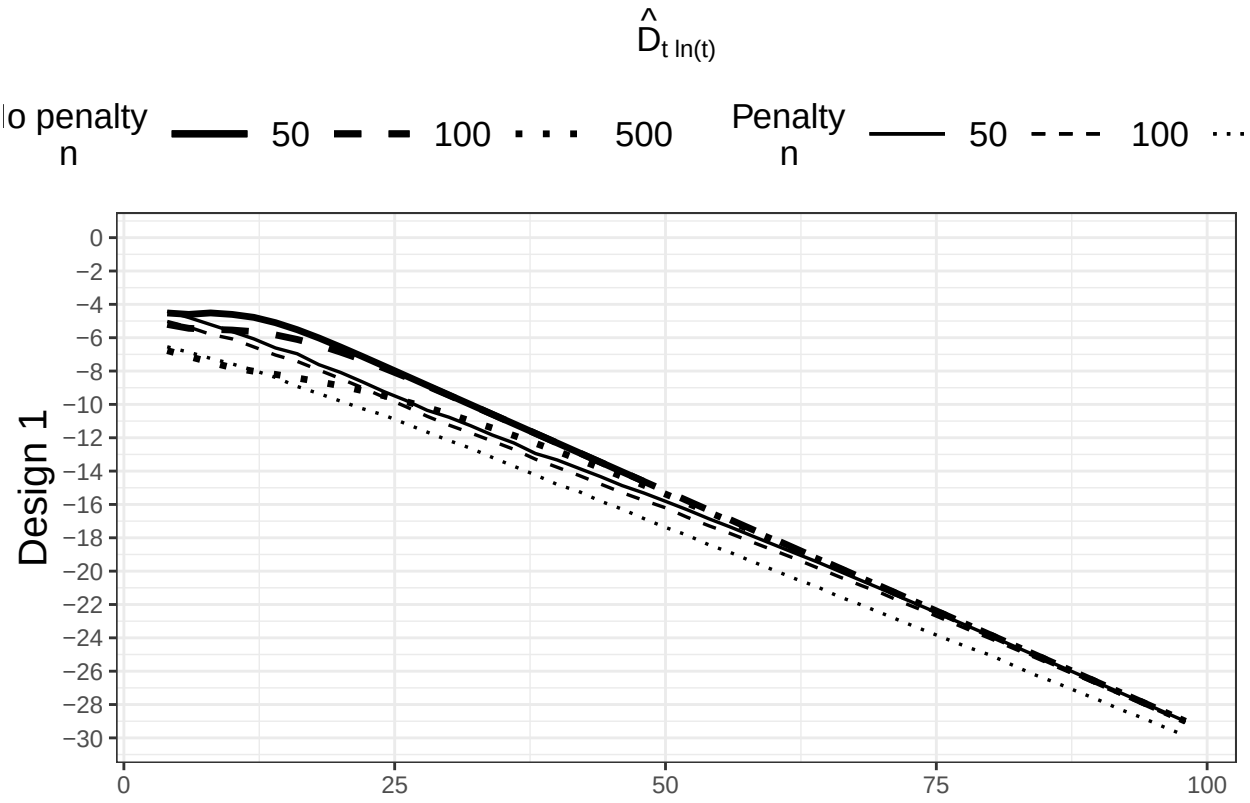

```
# cairo_pdf("BiasRidge.pdf", width = 15, height = 7) # Plot together and save

# ggarrange(g1, g3, g5, g7, g2, g4, g6, g8, ncol=4, nrow=2, common.legend = TRUE,
# legend="top")

# dev.off()

#  Variance Plots (similar for Setting 2 and Hel_normal, BWD1, BWD2)

errors1 = VTS$MI$Setting1$NonPenal$sd^2
errors2 = VTS$MI$Setting1$Penal$sd^2

frame = as.data.frame(cbind(svalues,t(errors1),t(errors2)))
frame$V2[which(frame$V2 == 0)] = rep(NA,length(which(frame$V2 == 0)))

# Make plot

g1 = ggplot(frame) + geom_line(aes(x = svalues, y = V2, lwd = "a",lty = "a")) +
  geom_line(aes(x = svalues, y = V5, lwd = "b",lty = "a")) +
  geom_line(aes(x = svalues, y = V3, lwd = "a",lty = "b")) +
  geom_line(aes(x = svalues, y = V6, lwd = "b",lty = "b")) +
  geom_line(aes(x = svalues, y = V4, lwd = "a",lty = "c")) +
  geom_line(aes(x = svalues, y = V7, lwd = "c",lty = "c")) +
  scale_y_continuous(breaks = seq(0,0.01, by = 0.0005),limits = c(0,0.01)) +
  xlab("") + ylab("Design 1") + ggtitle(expression(hat(D)["t ln(t)"])) +
  theme_bw() + theme(plot.title = element_text(hjust=0.5),
```

```
                    axis.title = element_text(size=15)) +
  theme(legend.position = "top",
        legend.text=element_text(size=13),legend.title=element_text(size=13),
        legend.key.width = unit(1.25, 'cm'),legend.direction = "horizontal",
        legend.box = "horizontal",
        legend.key = element_rect(fill = "transparent")) +
  scale_linewidth_manual(values = c(1.2,0.6,0.6), labels = c("50","100","500"),
                         name = "No penalty\n        n") +
  scale_linetype_manual(values=c("solid", "dashed","dotted"),
                        labels = c("50","100","500"),
                        name = "Penalty\n     n") +
  guides(linetype = guide_legend(override.aes = list(linetype = c("solid","dashed",
                                                                  "dotted"),
                                                     linewidth = c(0.6,0.6,0.6),
                                                     labels = c("50","100","500"))),
         linewidth = guide_legend(override.aes = list(linewidth = c(1.2,1.2,1.2),
                                                      linetype = c("solid","dashed",
                                                                   "dotted"),
                                                      labels = c("50","100","500"))))

g1
```



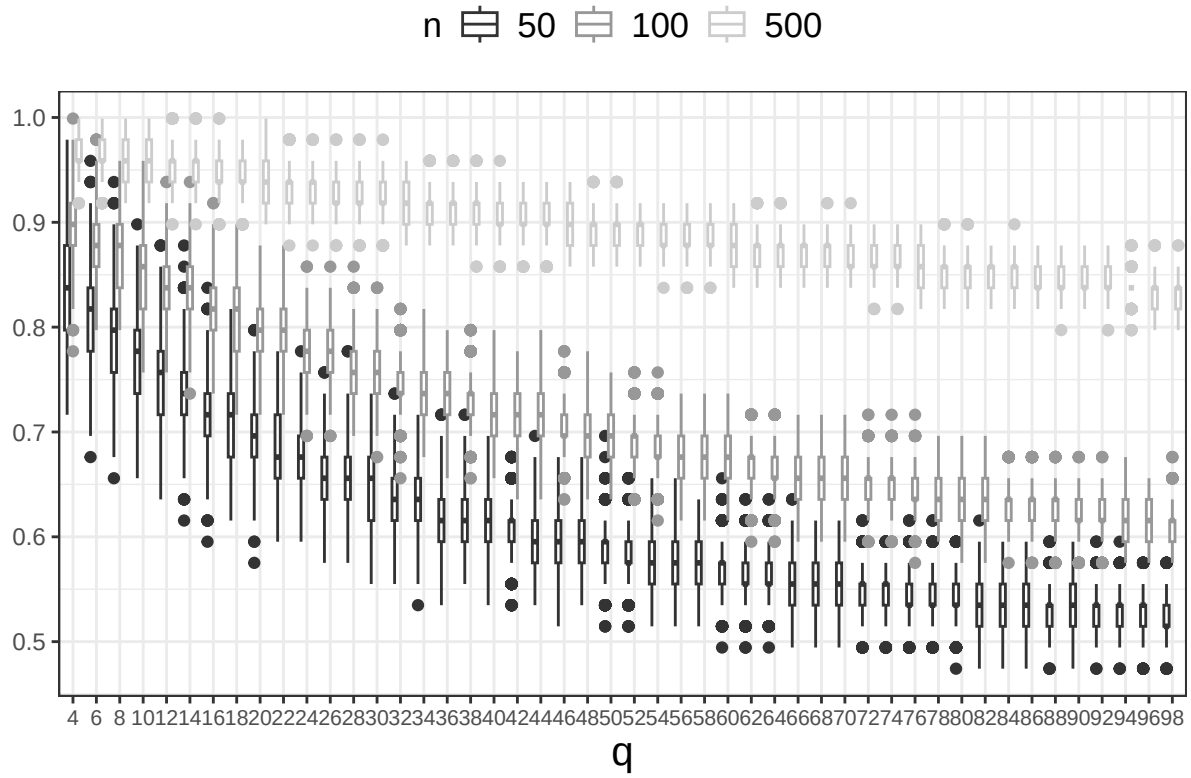

```
# cairo_pdf("VarianceRidge.pdf", width = 15, height = 7) # Plot together and save

# ggarrange(g1, g3, g5, g7, g2, g4, g6, g8, ncol=4, nrow=2, common.legend = TRUE,
# legend="top")
```

```
# dev.off()

# Logarithm of MSE as function of dimension for non-penalized vs Ridge penalized

errors1 = log(VTS$MI$Setting1$NonPenal$bias^2 + VTS$MI$Setting1$NonPenal$sd^2)
errors2 = log(VTS$MI$Setting1$Penal$bias^2 + VTS$MI$Setting1$Penal$sd^2)

frame = as.data.frame(cbind(svalues,t(errors1),t(errors2)))
frame$V2[which(frame$V2 == -Inf)] = rep(NA,length(which(frame$V2 == -Inf)))

# Make plot

g1 = ggplot(frame) + geom_line(aes(x = svalues, y = V2, lwd = "a",lty = "a")) +
  geom_line(aes(x = svalues, y = V5, lwd = "b",lty = "a")) +
  geom_line(aes(x = svalues, y = V3, lwd = "a",lty = "b")) +
  geom_line(aes(x = svalues, y = V6, lwd = "b",lty = "b")) +
  geom_line(aes(x = svalues, y = V4, lwd = "a",lty = "c")) +
  geom_line(aes(x = svalues, y = V7, lwd = "c",lty = "c")) +
  scale_y_continuous(breaks = seq(-30,0, by = 2),limits = c(-30,0)) + xlab("") +
  ylab("Design 1") + ggtitle(expression(hat(D)["t ln(t)"])) + theme_bw() +
  theme(plot.title = element_text(hjust=0.5),
        axis.title = element_text(size=15)) +
  theme(legend.position = "top",
        legend.text=element_text(size=13),legend.title=element_text(size=13),
        legend.key.width = unit(1.25, 'cm'),legend.direction = "horizontal",
        legend.box = "horizontal",
        legend.key = element_rect(fill = "transparent")) +
  scale_linewidth_manual(values = c(1.2,0.6,0.6), labels = c("50","100","500"),
                         name = "No penalty\n        n") +
  scale_linetype_manual(values=c("solid","dashed","dotted"),
                        labels = c("50","100","500"),
                        name = "Penalty\n     n") +
  guides(linetype = guide_legend(override.aes = list(linetype = c("solid","dashed",
                                                                  "dotted"),
                                                     linewidth = c(0.6,0.6,0.6),
                                                     labels = c("50","100","500"))),
         linewidth = guide_legend(override.aes = list(linewidth = c(1.2,1.2,1.2),
                                                      linetype = c("solid","dashed",
                                                                   "dotted"),
                                                      labels = c("50","100","500"))))

g1
```



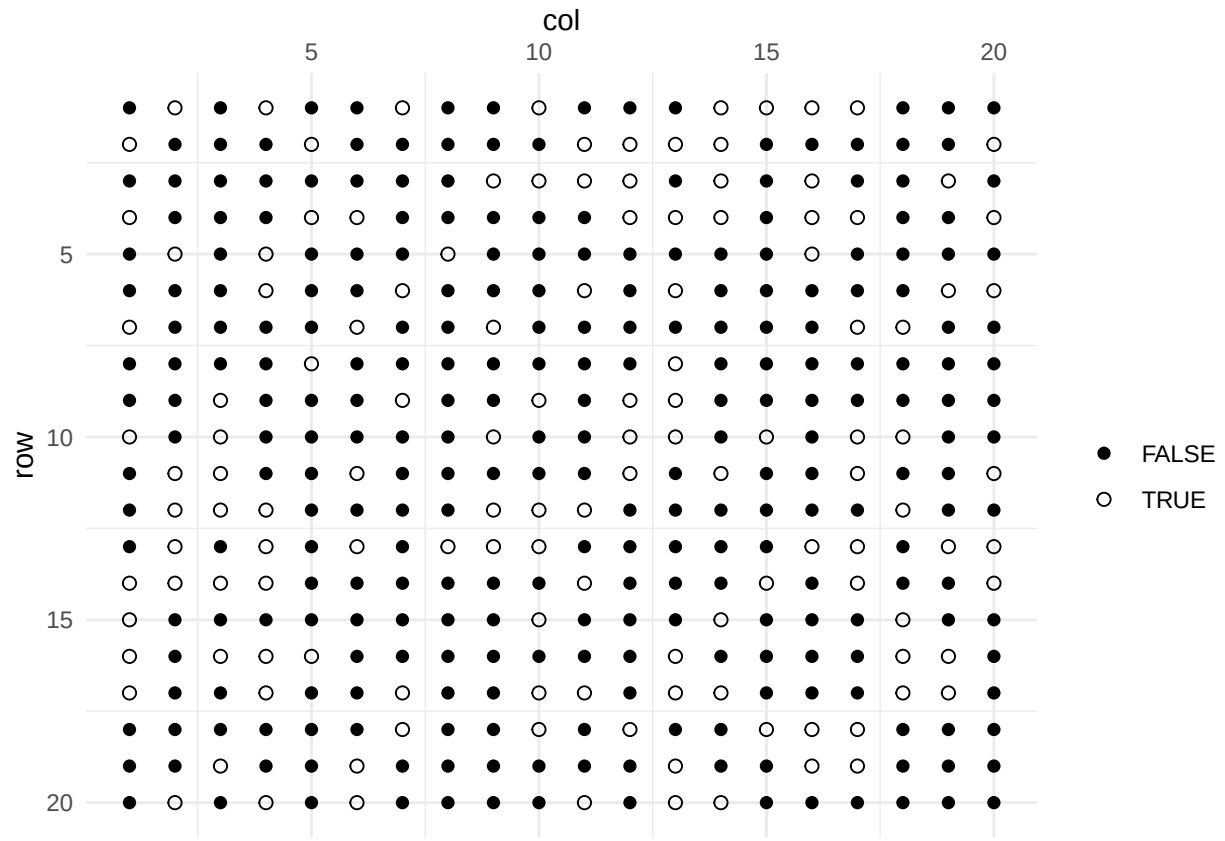

```
# cairo_pdf("Plot7.pdf", width = 15, height = 7) # Plot together and save

# ggarrange(g1, g3, g5, g7, g2, g4, g6, g8, ncol=4, nrow=2, common.legend = TRUE,
# legend="top")

# dev.off()

# Penalization parameters omega

est_omegas = VTS$MI$Setting1$Penal$omegas

omegas_all = c()
for(i in 1:length(nvalues)){
  for(j in 1:length(svalues)){
    omegas_all = rbind(omegas_all,cbind(est_omegas[i,j,],nvalues[i],svalues[j]))
  }
}

frame = as.data.frame(omegas_all)

colnames(frame) = c("Omegas","n","s")
frame$n = as.factor(frame$n)
frame$s = as.factor(frame$s)

# cairo_pdf("Plot9.pdf", width = 8, height = 5) # Plot and save
```

```
ggplot(frame, aes(x=s, y=Omegas, color=n)) + geom_boxplot() +  xlab("q") + ylab("") +
  theme_bw() + ggtitle("") + theme(plot.title = element_text(hjust=0.5),
                                   axis.title = element_text(size=15)) +
theme(legend.position = "top",legend.text=element_text(size=13),
      legend.title=element_text(size=13),axis.text.x= element_text(size = 7.5)) +
  scale_color_grey()
```




```
# dev.off()
```

```
# Simulation for Lasso-type estimation

# VTS = list("Setting1" = list("Lasso" = 0, "ALasso" = 0, "Scad" = 0, "GLasso" = 0),
#            "Setting2" = list("Lasso" = 0, "ALasso" = 0, "Scad" = 0, "GLasso" = 0),
#            "Setting3" = list("Lasso" = 0, "ALasso" = 0, "Scad" = 0, "GLasso" = 0))
#            "Setting4" = list("Lasso" = 0, "ALasso" = 0, "Scad" = 0, "GLasso" = 0,
#                              "SGLasso" = 0))

## Scenario 1 (lower dimension, quite low sparsity, no entire blocks of zero)

set.seed(123)
R = gen_cor(20,0.6) # True sparse correlation matrix
q = 20 # Total dimension
dim = c(3,3,3,3,3,3,2) # Dimensions of the random vectors
create_image(R == 0)
```



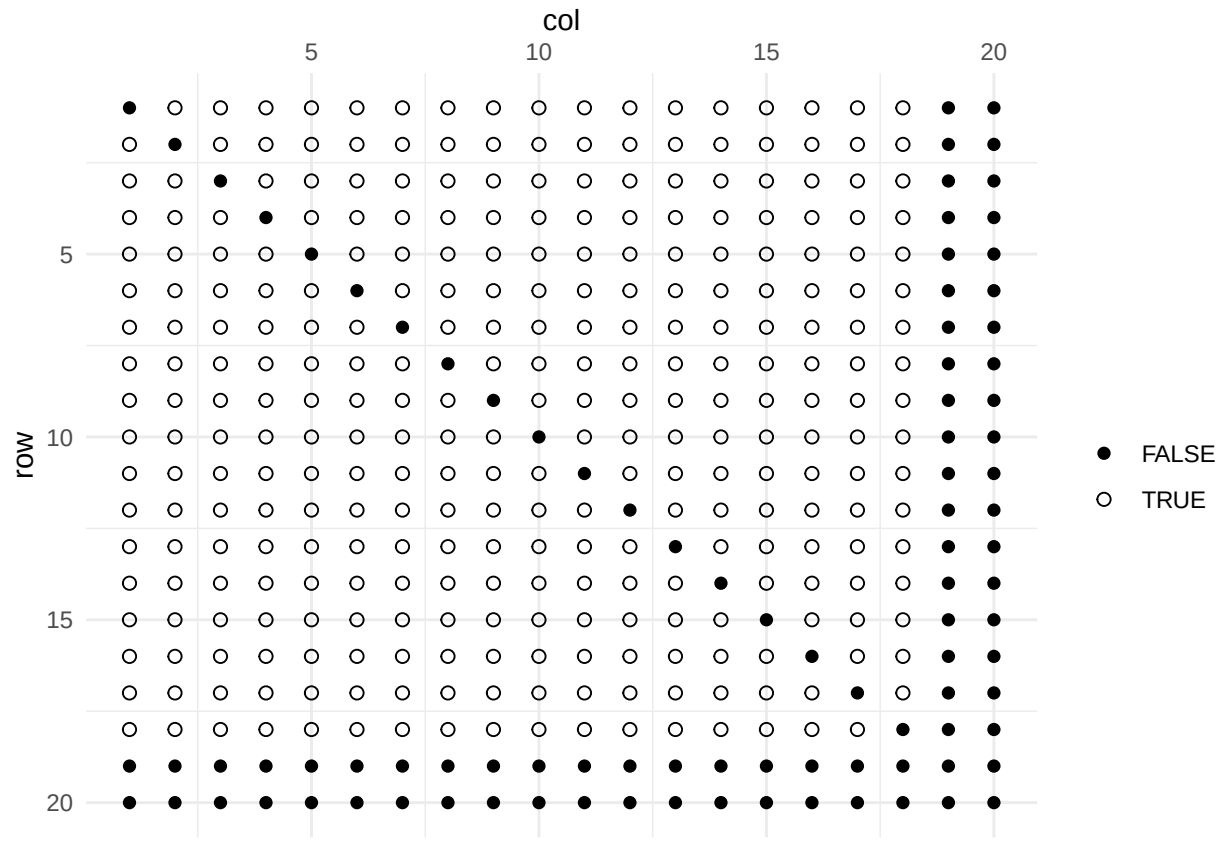

```
length(which(R == 0)) / (q^2) # Sparseness density
```

```
## [1] 0.325
```

```
n = c(50,100,500)
M = 1000
rho = seq(0.01, 0.6, length = 50)
omega = seq(0.01, 0.6, length = 50)

## Scenario 2 (lower dimension, quite high sparsity, entire blocks of zero)

q = 20 # Must equal m*p + k
m = 3
p = 6 # Construct p^2 blocks of size m^2
k = 2 # Add a p+1'th block of dimension k that has no zeroes

set.seed(123)
R = sim_cor(m,p,k) # True sparse correlation matrix
dim = c(rep(m,p),k) # Dimensions of the random vectors
length(which(R == 0)) / (q^2) # Sparseness density
```

```
## [1] 0.765
```

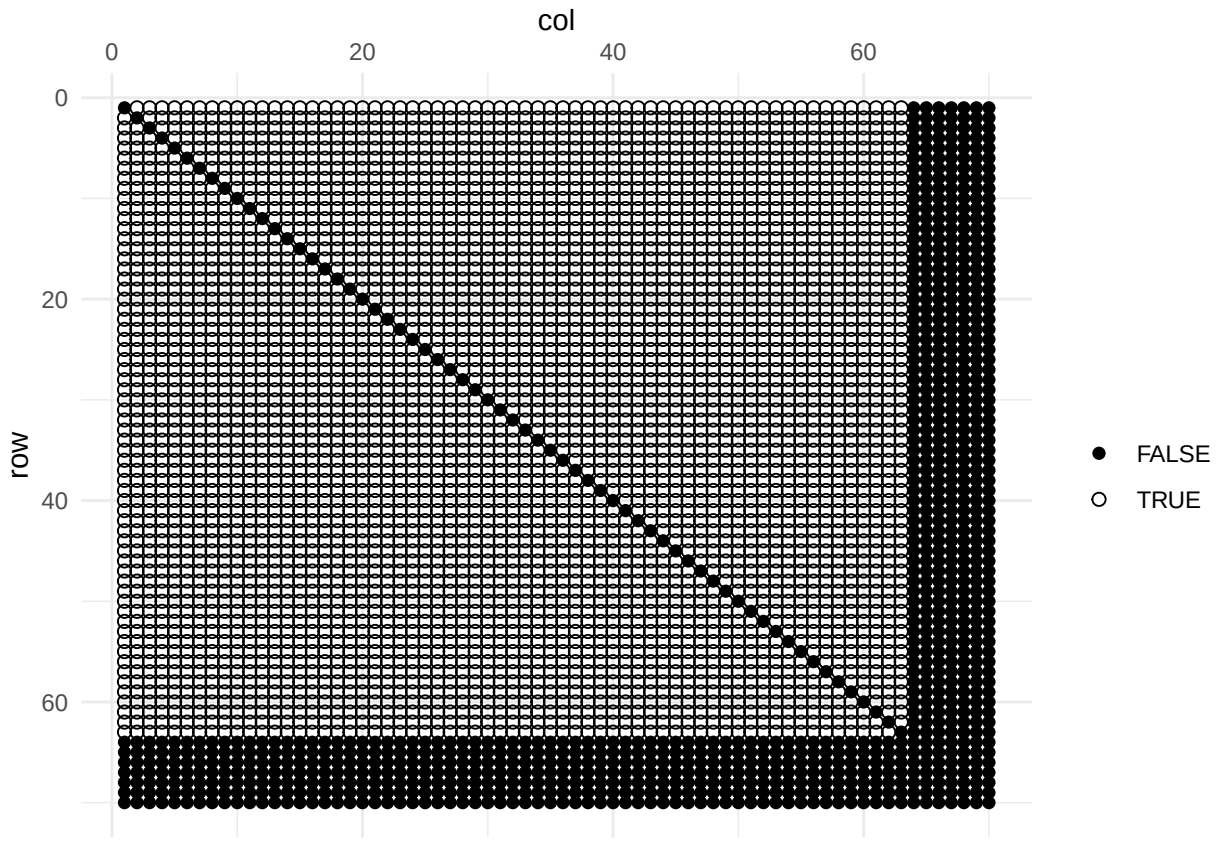
```
create_image(R == 0)
```



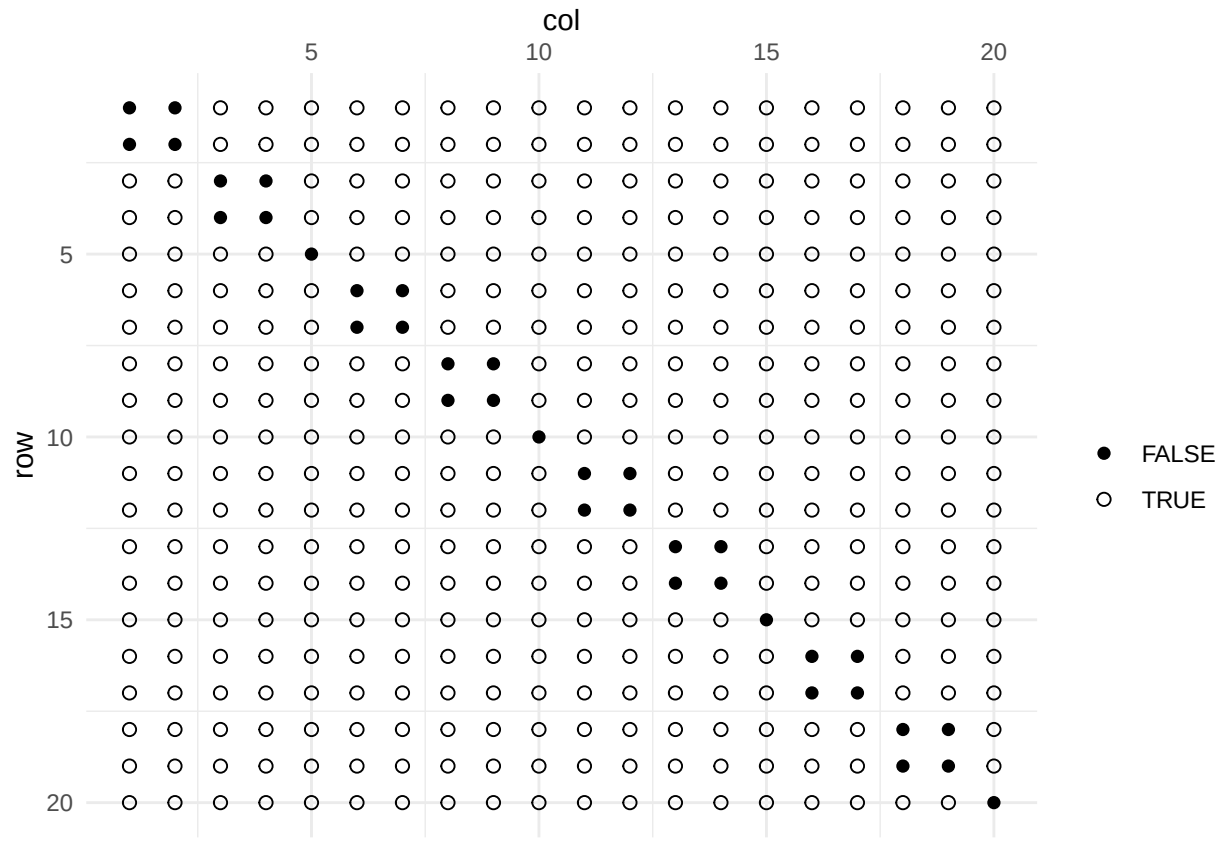

```
n = c(50,100,500)
M = 1000
rho = seq(0.01, 0.6, length = 50)
omega = seq(0.01, 0.6, length = 50)

## Scenario 3 (higher dimension, quite high sparsity, entire blocks of zero)

q = 70 # Must equal m*p + k
m = 21
p = 3 # Construct p^2 blocks of size m^2
k = 7 # Add a p+1'th block of dimension k that has no zeroes

set.seed(345)
R = sim_cor(m,p,k) # True sparse correlation matrix
dim = c(rep(m,p),k) # Dimensions of the random vectors
length(which(R == 0)) / (q^2) # Sparseness density
```

```
## [1] 0.7971429
```

```
create_image(R == 0)
```




```
n = c(100,500)
M = 1000
rho = seq(0.01, 0.6,length = 50)
omega = seq(0.01, 0.6,length = 50)

## Scenario 4 (lower dimension, high sparsity, group sparsity + sparsity within blocks)

set.seed(123)
R11 = gen_cor(5,0.2)
R = adiag(R11, R11, R11, R11)
q = 20
dim = c(5,5,5,5)
create_image(R == 0)
```




```
length(which(R == 0)) / (q^2) # Sparseness density
```

```
## [1] 0.91
```

```
# Simulation

# The function Do_sim performs the simulation;
# For lasso, set omega = omega
# For scad, set omega = omega, scad = TRUE
# For adaptive lasso, set rho = rho
# For group lasso, set omega = omega, group = TRUE

out = Do_sim(R,dim,n,M,omega = omega)

# VTS$Setting1$Lasso = out
# save(VTS, file = "LassoSimNew.Rdata") # Save data
```

```
load("LassoSimNew.Rdata") # Load data
out = VTS$Setting1$Lasso

# Average TPR and FPR

# TPR
for(i in 1:length(n)){
  print(paste("TPR for n =", n[i], ":", mean(out$TPR[[paste("n =", toString(n[i]))]])))
}
```

```
## [1] "TPR for n = 50 : 0.831184615384615"
```

```
## [1] "TPR for n = 100 : 0.724230769230769"
## [1] "TPR for n = 500 : 0.554307692307692"
```

```
# FPR
for(i in 1:length(n)){
  print(paste("FPR for n =" ,n[i] , ":", mean(out$FPR[[paste("n =", toString(n[i]))]])))
}
```

```
## [1] "FPR for n = 50 : 0.592251851851852"
## [1] "FPR for n = 100 : 0.354785185185185"
## [1] "FPR for n = 500 : 0.121237037037037"
```

```
# Chosen rhos (or omegas)

# In case rhos

# for(i in 1:length(n)){
#  print(paste("Rhos for n =",n[i], ":"))
#  print(out$est_rho[[paste("n =", toString(n[i]))]])
#  print(paste("Mean = ", mean(out$est_rho[[paste("n =", toString(n[i]))]])))
#}

# In case omegas

for(i in 1:length(n)){
 print(paste("Omegas for n =",n[i], ":"))
 print(paste("Mean = ", mean(out$est_omega[[paste("n =", toString(n[i]))]])))
}
```

```
## [1] "Omegas for n = 50 :"
## [1] "Mean =  0.520446326530612"
## [1] "Omegas for n = 100 :"
## [1] "Mean =  0.259220816326531"
## [1] "Omegas for n = 500 :"
## [1] "Mean =  0.0553818367346939"
```

```
# Comparing penalized estimator to ML estimator

# Frobenius error

# RMSE (sqrt(Mean^2 + Var))
for(i in 1:length(n)){
  print(paste("RMSE for n = ", n[i], ":"))
  print(paste("MLE :", sqrt(mean(out$FerrorsR_est[[paste("n =", toString(n[i]))]])^2 +
                              var(out$FerrorsR_est[[paste("n =", toString(n[i]))]]))))
  print(paste("covglasso :",
              sqrt(mean(out$FerrorsR_est_covglasso[[paste("n =", toString(n[i]))]])^2 +

                var(out$FerrorsR_est_covglasso[[paste("n =", toString(n[i]))]]))))
}
```

```
## [1] "RMSE for n =  50 :"
## [1] "MLE : 0.137210767063556"
## [1] "covglasso : 0.10488452810613"
## [1] "RMSE for n =  100 :"
## [1] "MLE : 0.0961646577597088"
## [1] "covglasso : 0.0767418993671932"
```

```
## [1] "RMSE for n =  500 :"
## [1] "MLE : 0.0428178890951772"
## [1] "covglasso : 0.0392541413916955"
```

```
# Mutual information

# RMSE
for(i in 1:length(n)){
    print(paste("RMSEs for n = ", n[i], ":"))
    print(paste("MLE :",
                sqrt((mean(normalize(out$MIests[[paste("n =", toString(n[i]))]])) -
                       normalize(MI_normal(R,dim)))^2 +
                      var(normalize(out$MIests[[paste("n =", toString(n[i]))]])))))
    print(paste("covglasso :",
          sqrt((mean(normalize(out$MIests_covglasso[[paste("n =", toString(n[i]))]])) -
                  normalize(MI_normal(R,dim)))^2 +
            var(normalize(out$MIests_covglasso[[paste("n =", toString(n[i]))]])))))
}
```

```
## [1] "RMSEs for n =  50 :"
## [1] "MLE : 0.000399298152579561"
## [1] "covglasso : 0.208910419385468"
## [1] "RMSEs for n =  100 :"
## [1] "MLE : 0.000289794056033824"
## [1] "covglasso : 0.00539448490827801"
## [1] "RMSEs for n =  500 :"
## [1] "MLE : 8.33097865321631e-05"
## [1] "covglasso : 0.000768156217122551"
```

```
# Hellinger distance

# RMSE
for(i in 1:length(n)){
  print(paste("RMSEs for n = ", n[i], ":"))
  print(paste("MLE :",
              sqrt((mean(out$Helests[[paste("n =", toString(n[i]))]]) -
                      Hel_normal(R,dim))^2 +
                      var(out$Helests[[paste("n =", toString(n[i]))]])))))
  print(paste("covglasso :",
      sqrt((mean(out$Helests_covglasso[[paste("n =", toString(n[i]))]]) -
              Hel_normal(R,dim))^2 +
            var(out$Helests_covglasso[[paste("n =", toString(n[i]))]])))))
}
```

```
## [1] "RMSEs for n =  50 :"
## [1] "MLE : 0.108881750034094"
## [1] "covglasso : 0.327381227839941"
## [1] "RMSEs for n =  100 :"
## [1] "MLE : 0.0436002928369955"
## [1] "covglasso : 0.152671838995316"
## [1] "RMSEs for n =  500 :"
## [1] "MLE : 0.00896342369675275"
## [1] "covglasso : 0.0540732765025117"
```

```
# BWD1
```

```
# RMSE
for(i in 1:length(n)){
  print(paste("RMSEs for n = ", n[i], ":"))
  print(paste("MLE :",
              sqrt((mean(out$BWD1ests[[paste("n =", toString(n[i]))]]) - BWD1(R,dim))^2 +

                                  var(out$BWD1ests[[paste("n =", toString(n[i]))]])))))
  print(paste("covglasso :",
              sqrt((mean(out$BWD1ests_covglasso[[paste("n =", toString(n[i]))]]) -
                      BWD1(R,dim))^2 +

             var(out$BWD1ests_covglasso[[paste("n =", toString(n[i]))]])))))
}
```

```
## [1] "RMSEs for n =  50 :"
## [1] "MLE : 0.0576476694448822"
## [1] "covglasso : 0.0542261942844824"
## [1] "RMSEs for n =  100 :"
## [1] "MLE : 0.0262528659679739"
## [1] "covglasso : 0.0324297381878021"
## [1] "RMSEs for n =  500 :"
## [1] "MLE : 0.00519937970644673"
## [1] "covglasso : 0.014307160836326"
```

```
# BWD2

# RMSE
for(i in 1:length(n)){
  print(paste("RMSEs for n = ", n[i], ":"))
  print(paste("MLE :",
              sqrt((mean(out$BWD2ests[[paste("n =", toString(n[i]))]]) - BWD2(R,dim))^2 +

                                  var(out$BWD2ests[[paste("n =", toString(n[i]))]])))))
  print(paste("covglasso :",
              sqrt((mean(out$BWD2ests_covglasso[[paste("n =", toString(n[i]))]]) -
                      BWD2(R,dim))^2 +

                var(out$BWD2ests_covglasso[[paste("n =", toString(n[i]))]])))))
}
```

```
## [1] "RMSEs for n =  50 :"
## [1] "MLE : 0.0568818580409094"
## [1] "covglasso : 0.0524953248677141"
## [1] "RMSEs for n =  100 :"
## [1] "MLE : 0.0261584945820606"
## [1] "covglasso : 0.0312790756406848"
## [1] "RMSEs for n =  500 :"
## [1] "MLE : 0.00524873627796485"
## [1] "covglasso : 0.0138454484443038"
```

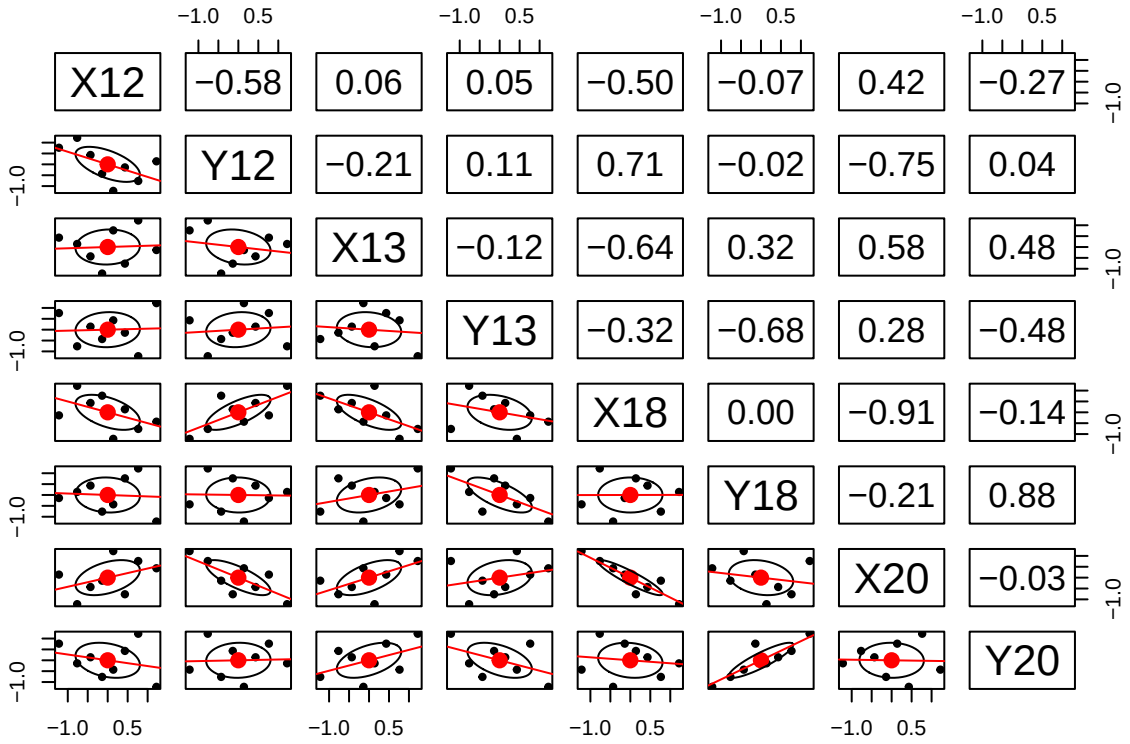
Next, we show the R code that was used for the two real data applications.

Real data applications

Application to sensory analysis

We start with Section 6.1.

```
# Real data example 1: smoothies data

data(smoothies)

# Remove the Ci variables

smoothies = smoothies[,-seq(3,72,by = 3)]

# Pairwise scatterplots

n = nrow(smoothies)
d = ncol(smoothies)

pseudos = matrix(0,n,d) # Pseudo observations
for(j in 1:d){pseudos[,j] = (n/(n+1)) * ecdf(smoothies[,j])(smoothies[,j])}
scores = qnorm(pseudos) # Normal scores
colnames(scores) = colnames(smoothies)

# cairo_pdf("ScatterSmoothies.pdf", width = 100, height = 100) # For saving plot
# pairs.panels2(scores,method="pearson",ellipses = TRUE,lm = TRUE,density = FALSE,
#                     rug = FALSE, breaks = 0, cex.cor = 1.7,cex.labels = 8)
# dev.off()

# Pairwise scatterplots of candidates P12,P13,P18 and P20

# cairo_pdf("ScatterSmoothies2.pdf", width = 20, height = 20) # For saving plot
pairs.panels2(scores[,c(23,24,25,26,35,36,39,40)],method="pearson",ellipses = TRUE,
              lm = TRUE,density = FALSE,
              rug = FALSE, breaks = 0, cex.cor = 1.3,cex.labels = 2)
```




```
# dev.off()

# Strongest 2D similarities (similarly for D2)

twodsims = integer(276)
comb = combn(seq(1,47,by = 2), 2)

for(i in 1:276){
  pos1 = comb[1,i] ; pos2 = comb[2,i]
  data = cbind(smoothies[1:8,pos1:(pos1+1)],smoothies[1:8,pos2:(pos2+1)])
  twodsims[i] = BWD1(est_R(data,1),rep(2,2))
}

sorted = sort(twodsims, decreasing = TRUE)
comb[1:2,which(sorted[1] == twodsims)] # P18 and P20 win
```

```
## [1] 35 39
```

```
comb[1:2,which(sorted[2] == twodsims)] # P9 and P23 second place
```

```
## [1] 17 45
```

```
comb[1:2,which(sorted[275] == twodsims)] # P12 and P19 second last place
```

```
## [1] 23 37
```

```
comb[1:2,which(sorted[276] == twodsims)] # P12 and P13 second last place
```

```
## [1] 23 25
```

```r
# Strongest 3D similarities (similarly for D2)

threedsims = integer(2024)
comb = combn(seq(1,47,by = 2), 3)

for(i in 1:2024){
  pos1 = comb[1,i] ; pos2 = comb[2,i] ; pos3 = comb[3,i]
  data = cbind(smoothies[1:8,pos1:(pos1+1)],smoothies[1:8,pos2:(pos2+1)],
               smoothies[1:8,pos3:(pos3+1)])
  threedsims[i] = BWD1(est_R(data,1),rep(2,3))
}

sorted = sort(threedsims, decreasing = TRUE)
comb[1:3,which(sorted[1] == threedsims)] # P15, P18 and P20 win
```

```
## [1] 29 35 39
```

```r
comb[1:3,which(sorted[2] == threedsims)] # P9, P10 and P23 second place
```

```
## [1] 17 19 45
```

```r
comb[1:3,which(sorted[2023] == threedsims)] # P2, P3 and P14 second last place
```

```
## [1]  3  5 27
```

```r
comb[1:3,which(sorted[2024] == threedsims)] # P12, P13 and P21 last place
```

```
## [1] 23 25 41
```

```r
# Now looking at the similarity between the smoothies

new_smoothies = matrix(0,nrow = 24,ncol = 16)
# People as rows, coordinates of smoothies as columns
for(i in seq(1,47,by = 2)){
  for(j in 1:8){
    row = (i+1)/2 ; col = (2*j-1):(2*j)
    new_smoothies[((i+1)/2),((2*j-1):(2*j))] = as.numeric(smoothies[j,i:(i+1)])
  }
}

# Strongest 2D similarities (similarly for D2)

twodsims = integer(28)
comb = combn(seq(1,15,by = 2), 2)

for(i in 1:28){
  pos1 = comb[1,i] ; pos2 = comb[2,i]
  data = cbind(new_smoothies[1:24,pos1:(pos1+1)],new_smoothies[1:24,pos2:(pos2+1)])
  twodsims[i] = BWD1(est_R(data,1),rep(2,2))
}

sorted = sort(twodsims, decreasing = TRUE)
comb[1:2,which(sorted[1] == twodsims)] # S5 and S6 win
```

```
## [1]  9 11
```

```
comb[1:2,which(sorted[2] == twodsims)] # S1 and S2 second place
```

```
## [1] 1 3
```

```
comb[1:2,which(sorted[3] == twodsims)] # S4 and S6 third place
```

```
## [1]  7 11
```

```
comb[1:2,which(sorted[28] == twodsims)] # S6 and S7 lose
```

```
## [1] 11 13
```

```
# Strongest 3D similarities (similarly for D2)

threedsims = integer(56)
comb = combn(seq(1,15,by = 2), 3)

for(i in 1:56){
  pos1 = comb[1,i] ; pos2 = comb[2,i] ; pos3 = comb[3,i]
  data = cbind(new_smoothies[1:24,pos1:(pos1+1)],new_smoothies[1:24,pos2:(pos2+1)],
               new_smoothies[1:24,pos3:(pos3+1)])
  threedsims[i] = BWD1(est_R(data,1),rep(2,3))
}

sorted = sort(threedsims, decreasing = TRUE)
comb[1:3,which(sorted[1] == threedsims)] # S4, S5 and S6 win
```

```
## [1]  7  9 11
```

```
comb[1:3,which(sorted[2] == threedsims)] # S2, S5 and S6 second place
```

```
## [1]  3  9 11
```

```
comb[1:3,which(sorted[3] == threedsims)] # S3, S5 and S6 second place
```

```
## [1]  5  9 11
```

```
comb[1:3,which(sorted[56] == threedsims)] # S6, S7 and S8 lose
```

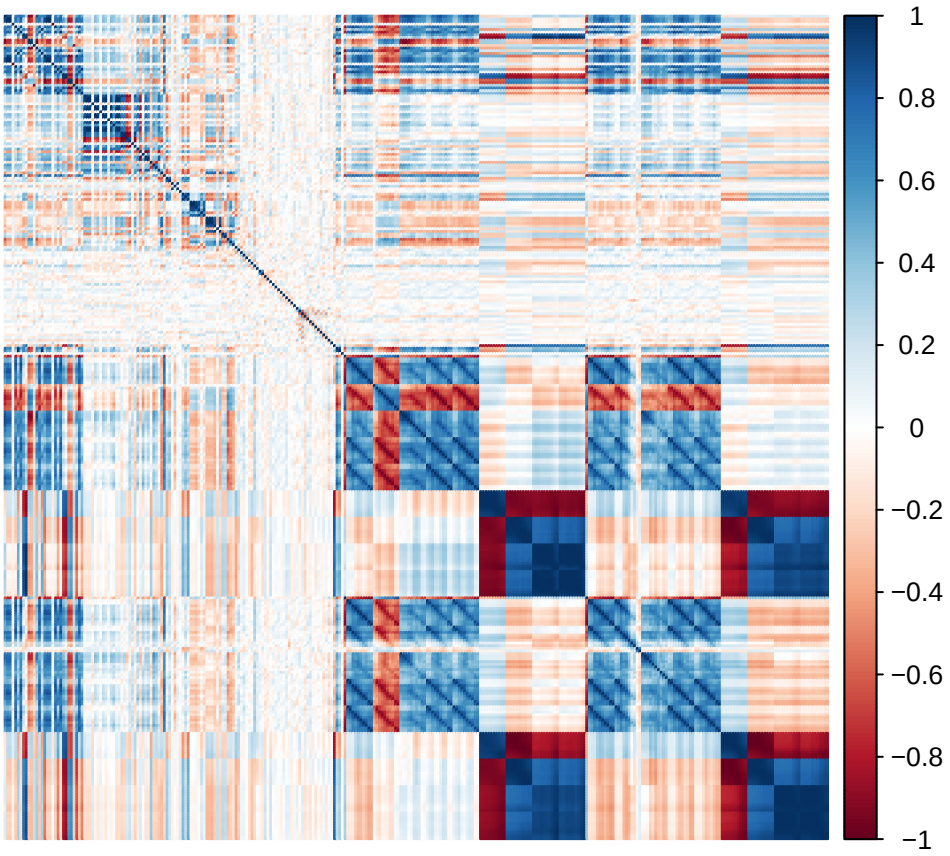
```
## [1] 11 13 15
```

Clustering dysphonia measures

Next, we look at the application considered in Section 6.2. Note that the R data files RealData.Rdata and Pvalues.Rdata are loaded.

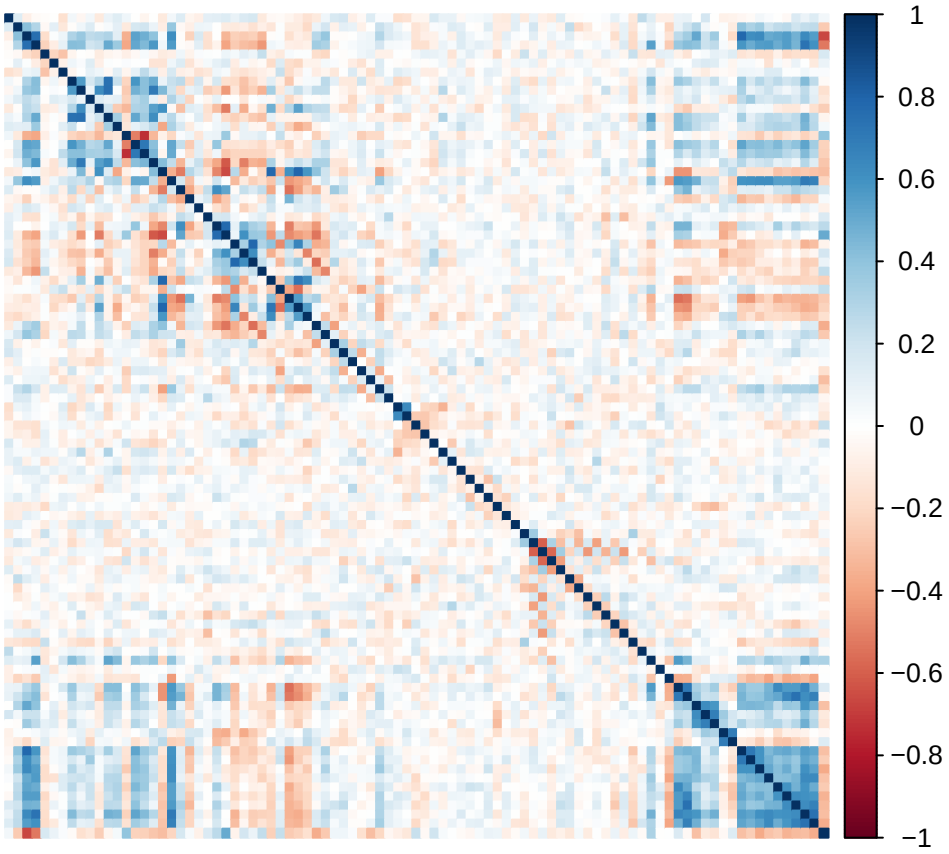
```
data = read_excel("./Voice.xlsx")
data = as.matrix(data[1:126,1:310])

comb = combs(1:310,2)

filter = function(data){

  # Pairwise removal of variables that have normal scrores rank correlation
  # larger than 0.8 in absolute value

  comb = combs(1:ncol(data),2)
  for(i in 1:nrow(comb)){
    if(abs(est_R(cbind(data[,comb[i,1]],data[,comb[i,2]]),1)[1,2]) > 0.8){
```

```
      new_data = data[,-comb[i,1]]
      print(ncol(new_data))
      return(filter(new_data))
    }
  }
  return(data)
}

colnames(data) = NULL
corrplot(est_R(data,1), method = 'color',tl.pos='n')
```




```
new_data = filter(data) # Remove detrimental correlations (91 variables remain)

# VTS = list()
# VTS$new_data = new_data
# save(VTS, file = "RealData.Rdata") # Save data
```

```
load("RealData.Rdata") # Load data
new_data = VTS$new_data

corrplot(est_R(new_data,1), method = 'color',tl.pos='n')
```




```r
# Pairwise GOF tests for testing whether Gaussian copula is appropriate

n = nrow(new_data)
d = ncol(new_data)

pvalues = matrix(0,d,d)
```

```r
set.seed(123)
for(i in 1:(d-1)){
  for(j in (i+1):d){
    print(paste("i = ", i, "and j = ", j))
    pvalues[i,j] = gofCopula(normalCopula(dim = 2),new_data[,c(i,j)])$p.value
  }
}

# save(pvalues,file = "Pvalues.Rdata") # Save data
```

```r
load("Pvalues.Rdata") # Load data

pvalues = pvalues[which(c(pvalues)!=0)]
frame = as.data.frame(pvalues)

# cairo_pdf("p-values.pdf", width = 8, height = 6.7) # For saving plot

ggplot(frame, aes(y = pvalues, x = NA, col = "a")) +
  geom_boxplot(fill='#A4A4A4', color="black") + theme_bw() +
  xlab("") + theme(axis.text.x = element_blank(),
```

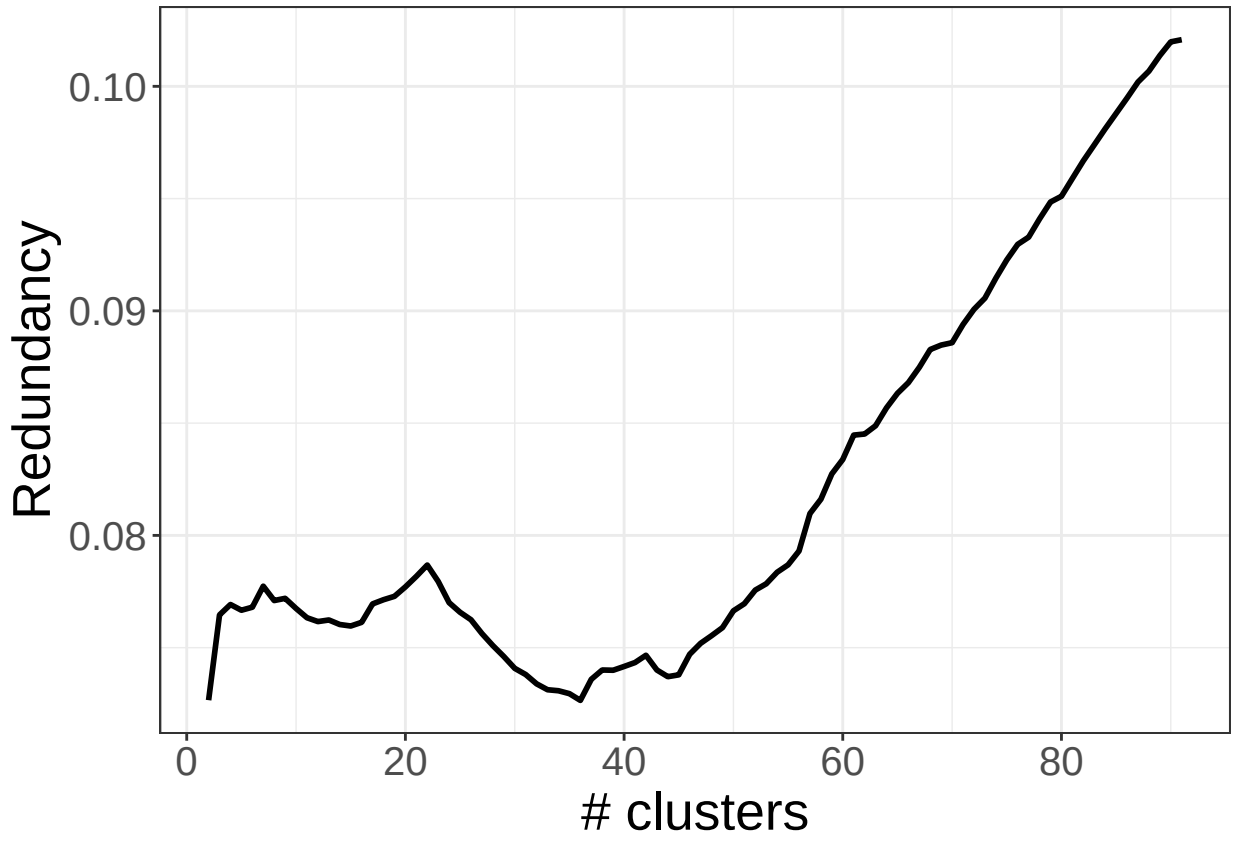
```
                  axis.ticks.x = element_blank()) + ylab("p-value") +
  stat_summary(fun.y=mean, geom="point", shape=23, size=3) +
  scale_colour_manual(name = element_blank(), values =c('black'),
                      labels = c("mean")) +
  theme(legend.position = c(0.85,0.91),legend.text = element_text(size=18)) +
  theme(legend.background=element_blank()) +
  theme(axis.text.y = element_text(size=14),
        axis.title.y = element_text(size=14))
```


```
# dev.off()
```

```
# Clustering

omegas = seq(0.01,0.999,len = 50)

set.seed(123) # Do the clustering; takes approximately 14 minutes to run
Iclust = Icluster(data = new_data, est_method = list("OT", coef = 1, "omegas" = omegas))

# VTS$Iclust = Iclust
# save(VTS, file = "RealData.Rdata") # Save data
```

```
load("RealData.Rdata") # Load data

set.seed(123)
red1 = Redundancy(new_data,VTS$Iclust,coef = "BWD1",omegas = omegas) # Redundancy
```

```
ggplot(red1) + geom_line(aes(x = V1, y = rev(red)), size = 1, col = "black") +
  xlab("# clusters") + ylab("Redundancy") +
  scale_x_continuous(breaks= pretty_breaks()) + theme_bw() +
  theme(axis.text = element_text(size=15), axis.title = element_text(size=20),
        plot.title = element_text(hjust=0.5, size = 25))
```




```
seq(2,89)[which.min(rev(red1$red))] # Partition with minimal redundancy
```

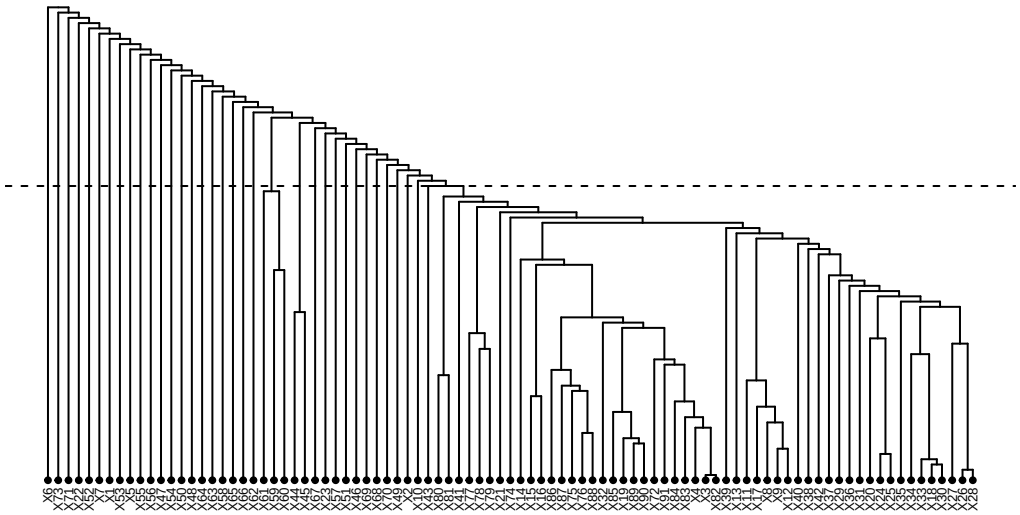
```
## [1] 36
```

```
min(red1$red) # Minimal redundancy
```

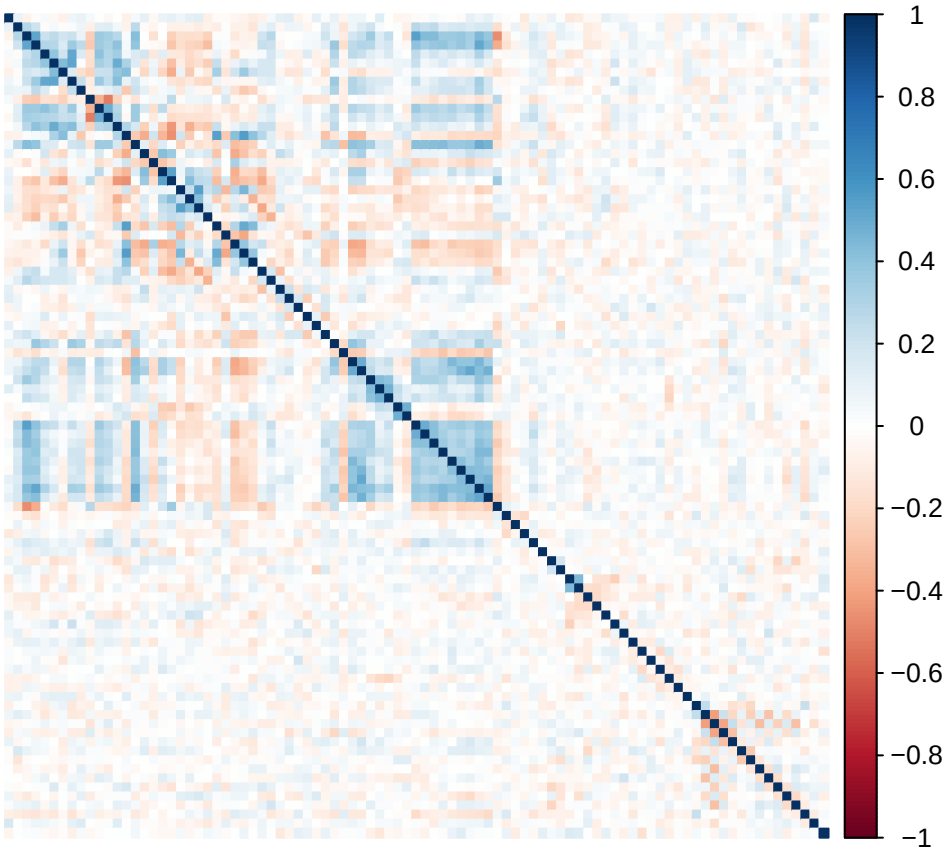
```
## [1] 0.07265135
```

```
# Plot dendrogram

l = length(VTS$Iclust$split) + 1
diams = VTS$Iclust$diam[2:l]
splits = VTS$Iclust$split
d = as.dendrogram(convert_to_hclust(VTS$Iclust))
d %>% dendextend::set("leaves_pch", 19) %>%
    dendextend::set("leaves_cex", 0.4) %>%
    dendextend::set("leaves_col", "black") %>%
    dendextend::set("branches_k_color","black") %>% # k = k for colors
    dendextend::set("labels_cex", 0.4) %>%
    plot(main = "Dendrogram", cex.main = 2.5, axes = F)
    abline(h = 56, lty = 2)
```

Dendrogram



```
# Pick the 36 cluster partition

# VTS$Iclust$hierarchy$Aleph_56

# Arrange data accordingly
sample = new_data[,c(1,2,3,4,8,9,11,12,13,14,15,16,17,18,19,20,21,24,25,26,27,28,29,
                     30,31,32,33,34,35,36,37,38,39,40,41,42,72,74,75,76,77,78,79,80,
                     81,82,83,84,85,86,87,88,89,90,91,5,6,7,10,22,23,43,44,45,46,47,48,49,
                     50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,
                     73)]

dim = c(1,1,53,1,1,1,1,1,1,1,2,1,1,1,1,1,1,1,1,1,1,1,1,1,3,1,1,1,1,1,1,1,1,1,1,1)
# Cluster dimensions

set.seed(123)
om_all = CVomega(sample,omegas,5) # 0.6962449, chosen Ridge penalty parameter
corrplot(est_R(sample,om_all), method = 'color',tl.pos='n')
```




```
BWD1(est_R(sample,om_all),dim) # Redundancy using the Ridge penalty
```

```
## [1] 0.07265135
```

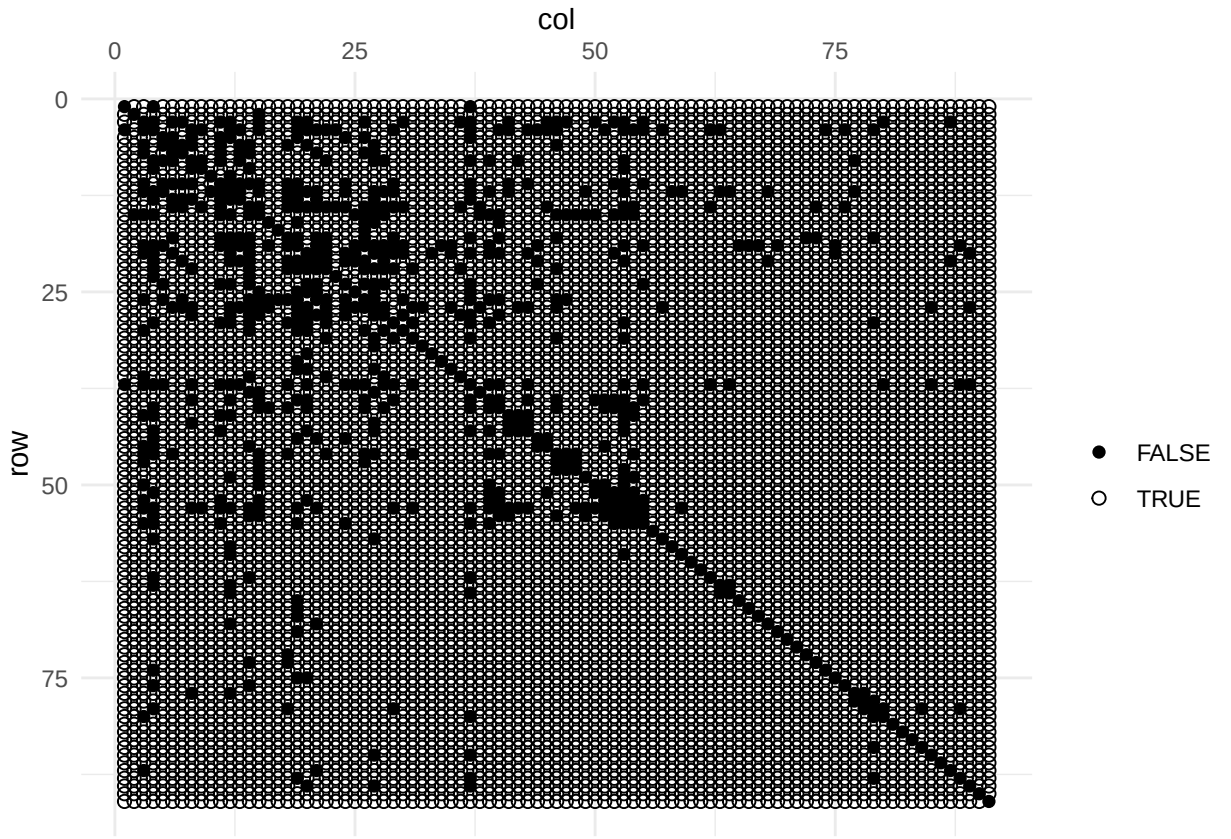
```
BWD1(est_R(sample,1),dim) # Redundancy using no penalty
```

```
## [1] 0.1901318
```

```
n = 126 ; q = 91
scores = matrix(0,n,q) # Normal scores
for(j in 1:q){
  scores[,j] = qnorm((n/(n + 1)) * ecdf(sample[,j])(sample[,j]))
}

Sigma_est = cov(scores) * ((n-1)/n)
```

```
# Lasso, BIC chooses omega = 0.6226531

omega = seq(0.01, 0.8, length = 50)
P = matrix(1,q,q)
diag(P) = 0
array = array(0,dim = c(q,q,length(omega)))
for(o in 1:length(omega)){array[,,o] = omega[o] * P}
covglasso = covglasso(S = Sigma_est, n = n, lambda = array, start = Sigma_est)

# VTS$Lasso = covglasso
# save(VTS, file = "RealData.Rdata") # Save data
```

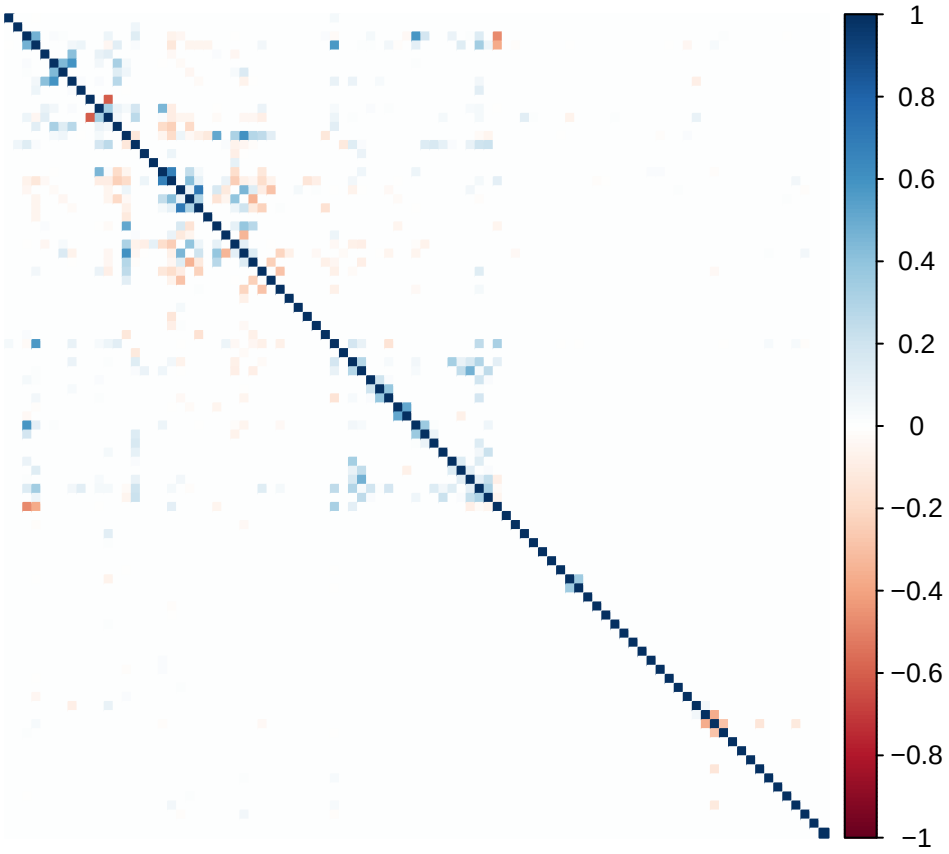
```
load("RealData.Rdata") # Load data

Sigma_est_covglasso = VTS$Lasso$sigma
D = sqrt(solve(diag(diag(Sigma_est_covglasso))))
R_est_covglasso = D %*% Sigma_est_covglasso %*% D

create_image(R_est_covglasso == 0)
```




```
BWD1(R_est_covglasso,dim) # Redundancy using lasso penalty
```

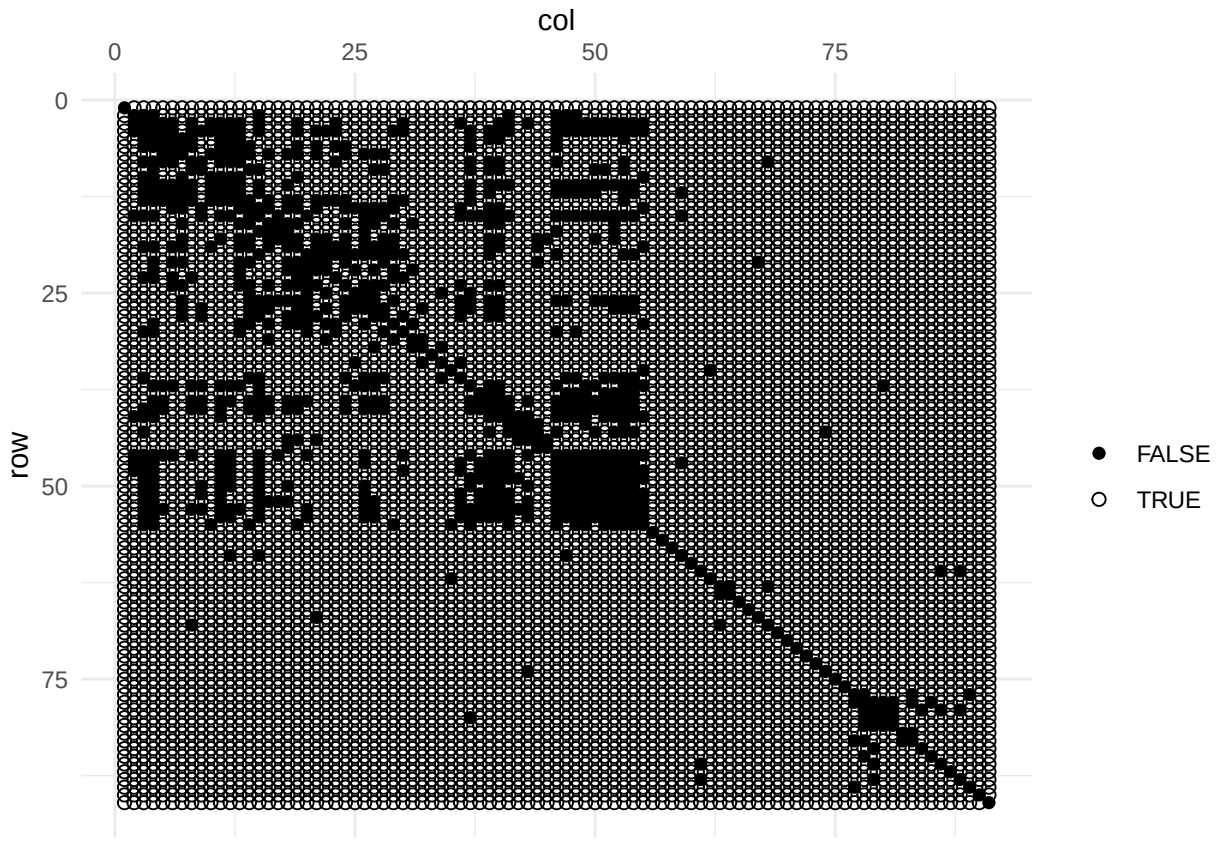
```
## [1] 0.001417654
```

```
corrplot(R_est_covglasso, method = 'color',tl.pos='n')
```




```
# Adaptive lasso, BIC chooses rho = 0.2869388

rho = seq(0.01, 0.6, length = 50)
covglasso = covglasso(S = Sigma_est, n = n,  rho = rho, start = Sigma_est)

# VTS$ALasso = covglasso
# save(VTS, file = "RealData.Rdata") # Save data
```

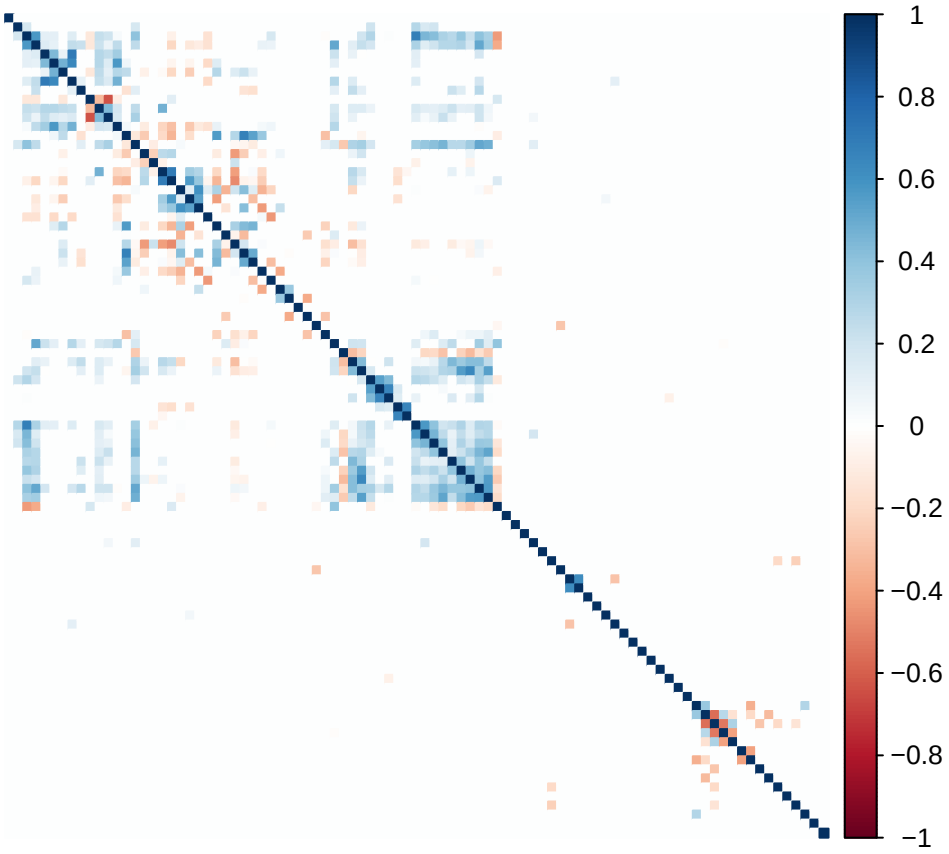
```
load("RealData.Rdata") # Load data

Sigma_est_covglasso = VTS$ALasso$sigma
D = sqrt(solve(diag(diag(Sigma_est_covglasso))))
R_est_covglasso = D %*% Sigma_est_covglasso %*% D

create_image(R_est_covglasso == 0)
```




```
BWD1(R_est_covglasso,dim) # Redundancy using adaptive lasso penalty
```

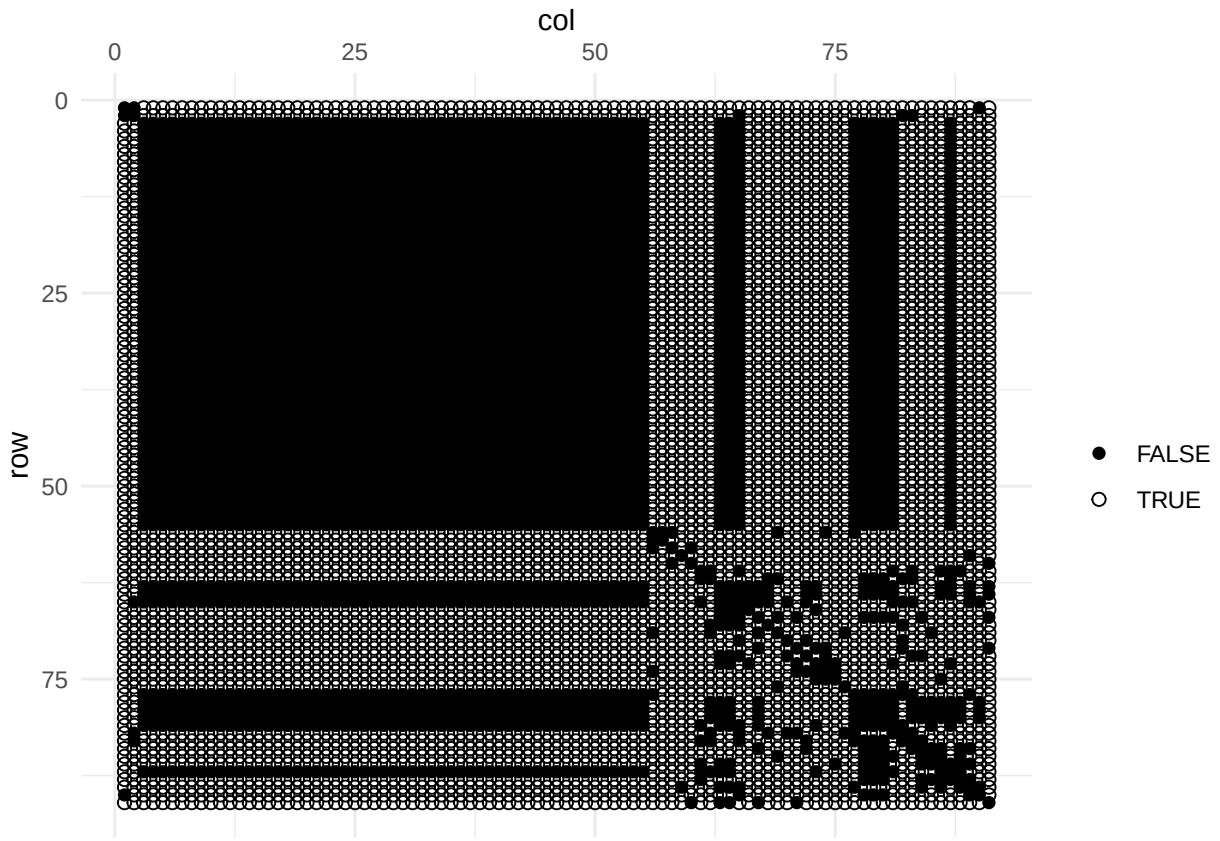
```
## [1] 0.01409284
```

```
corrplot(R_est_covglasso, method = 'color',tl.pos='n')
```




```
# Group lasso, BIC chooses 0.445

omega = seq(0.01, 0.8, length = 50)
groupl = GroupLasso(Sigma_est, Sigma_est, n, omega, dim, step.size = 100)

# VTS$GLasso = groupl
# save(VTS, file = "RealData.Rdata") # Save data
```

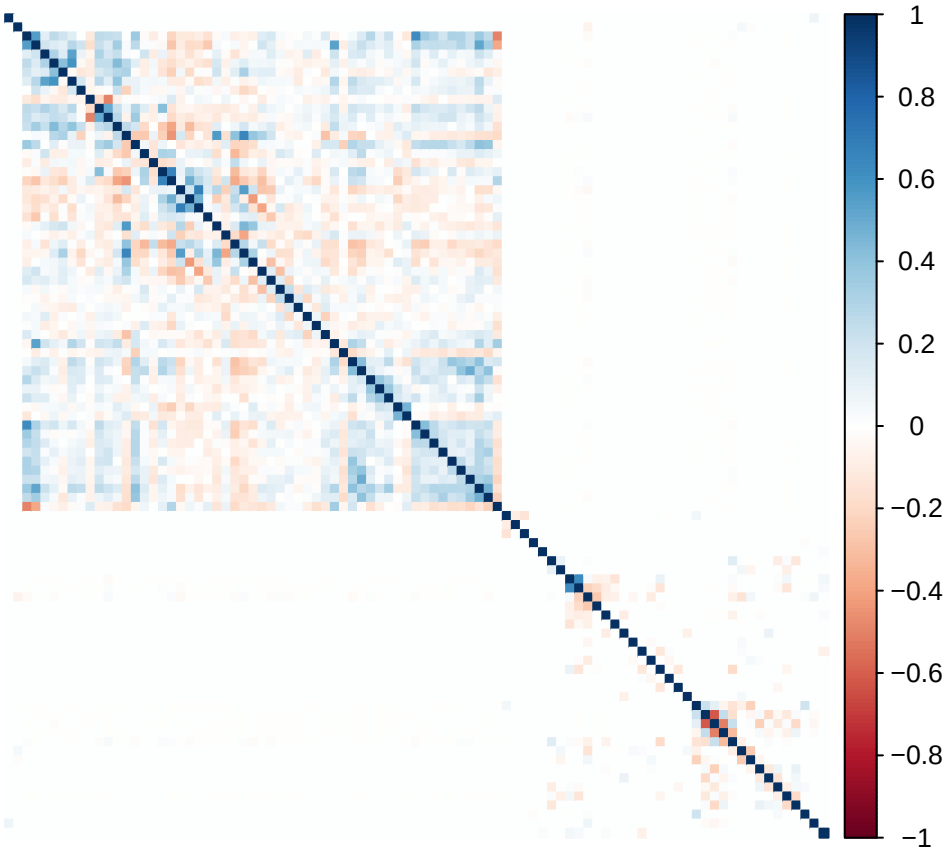
```
load("RealData.Rdata") # Load data

covglasso = VTS$GLasso
Sigma_est_covglasso = covglasso$sigma
D = sqrt(solve(diag(diag(Sigma_est_covglasso))))
R_est_covglasso = D %*% Sigma_est_covglasso %*% D

create_image(R_est_covglasso == 0)
```




```
BWD1(R_est_covglasso,dim) # Redundancy using group lasso penalty
```

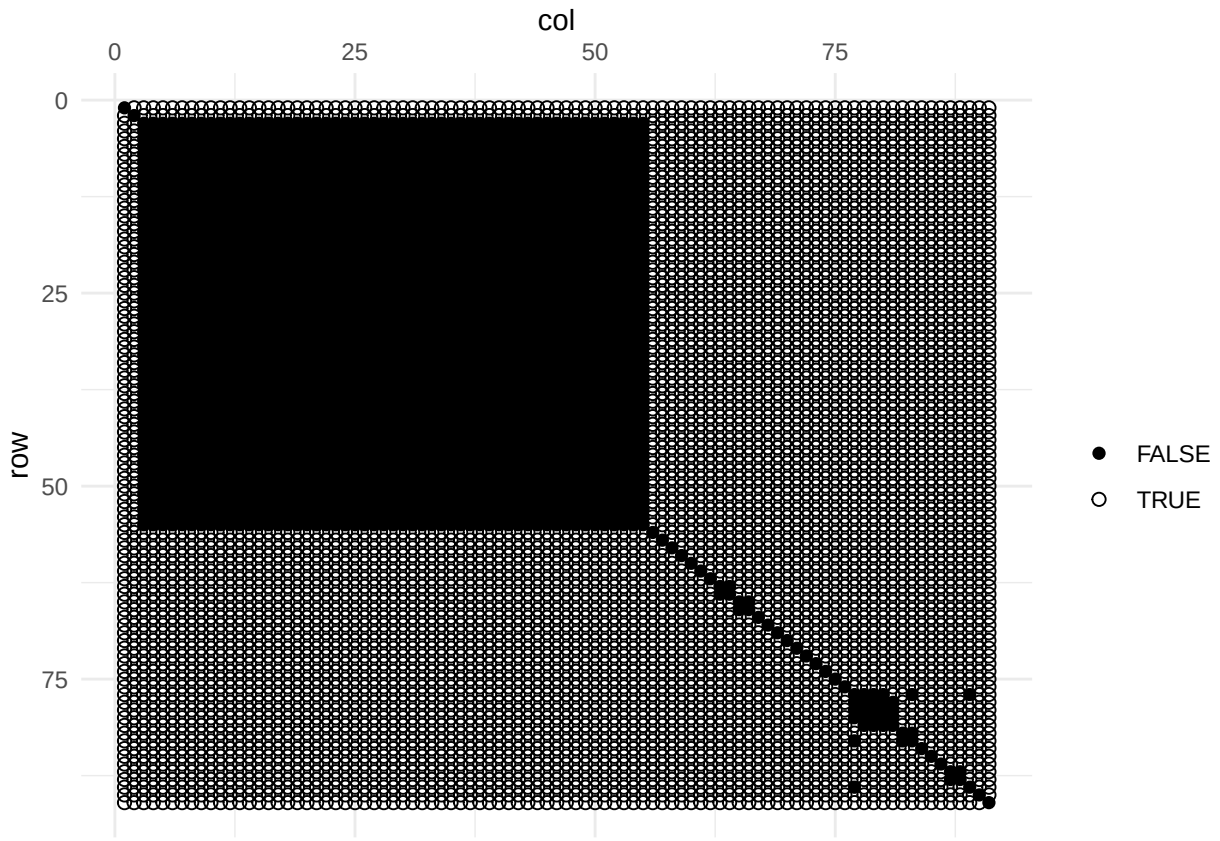
```
## [1] 0.01299779
```

```
corrplot(R_est_covglasso, method = 'color',tl.pos='n')
```




```
# Group lasso with omega = 0.67 and n.outer.steps = 100000

groupl = GroupLasso(Sigma_est, Sigma_est, n, 0.67, dim, step.size = 100)

# VTS$Glasso2 = groupl
# save(VTS, file = "RealData.Rdata") # Save data
```

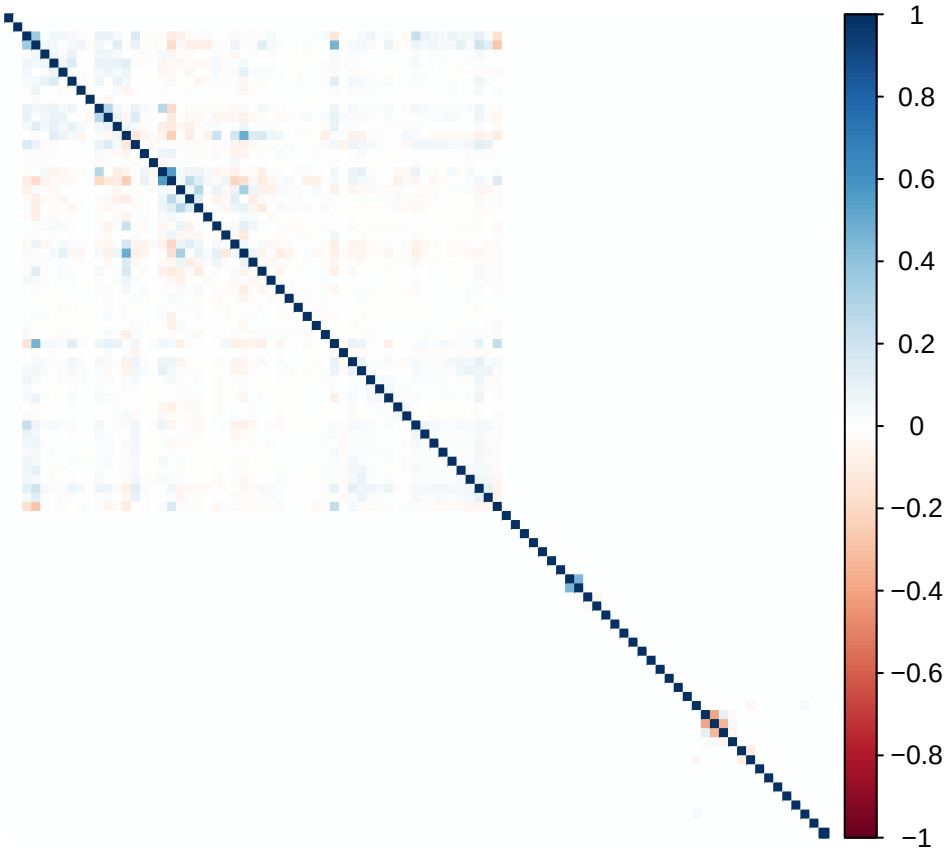
```
load("RealData.Rdata") # Load data

covglasso = VTS$Glasso2
Sigma_est_covglasso = covglasso$sigma
D = sqrt(solve(diag(diag(Sigma_est_covglasso))))
R_est_covglasso = D %*% Sigma_est_covglasso %*% D

create_image(R_est_covglasso == 0)
```




```
BWD1(R_est_covglasso,dim) # Redundancy using group lasso penalty with
```

```
## [1] 0.0001495374
```

```
# larger penalty parameter
corrplot(R_est_covglasso, method = 'color',tl.pos='n')
```



Finally, we show the code that was used for the extra illustrations in the Supplementary Material. Note that the R data files LassoSimSetting2CV.Rdata and Misspecification.Rdata are loaded.

Supplementary Material

Scenario 2 with cross-validation

```
# Scenario 2

q = 20 # Must equal m*p + k
m = 3
p = 6 # Construct p^2 blocks of size m^2
k = 2 # Add a p+1'th block of dimension k that has no zeroes

set.seed(123)
R = sim_cor(m,p,k) # True sparse correlation matrix
dim = c(rep(m,p),k) # Dimensions of the random vectors

n = c(50,100,500)
M = 1000
rho = seq(0.01, 0.6, length = 50)
omega = seq(0.01, 0.8, length = 50)
```

```
# VTS = list("Setting2CV" = list("Lasso" = 0, "ALasso" = 0, "Scad" = 0))

# Simulation
```

```r
out = Do_simCV(R,dim,n,M,omega = omega)

# VTS$Setting1$Lasso = out
# save(VTS, file = "LassoSimSetting2CV.Rdata") # Save data
```

```r
load("LassoSimSetting2CV.Rdata") # Load data
out = VTS$Setting2CV$Lasso

# Average TPR and FPR

# TPR
for(i in 1:length(n)){
  print(paste("TPR for n =", n[i], ":", mean(out$TPR[[paste("n =", toString(n[i]))]])))
}
```

```
## [1] "TPR for n = 50 : 0.990699346405229"
## [1] "TPR for n = 100 : 0.845013071895425"
## [1] "TPR for n = 500 : 0.772503267973856"
```

```r
# FPR
for(i in 1:length(n)){
  print(paste("FPR for n =" ,n[i] , ":", mean(out$FPR[[paste("n =", toString(n[i]))]])))
}
```

```
## [1] "FPR for n = 50 : 0.664255319148936"
## [1] "FPR for n = 100 : 0.106404255319149"
## [1] "FPR for n = 500 : 0.0408936170212766"
```

```r
# Chosen rhos (or omegas)

# In case rhos

# for(i in 1:length(n)){
#  print(paste("Rhos for n =",n[i], ":"))
#  print(out$est_rho[[paste("n =", toString(n[i]))]])
#  print(paste("Mean = ", mean(out$est_rho[[paste("n =", toString(n[i]))]])))
#}

# In case omegas

for(i in 1:length(n)){
 print(paste("Omegas for n =",n[i], ":"))
 print(paste("Mean = ", mean(out$est_omega[[paste("n =", toString(n[i]))]])))
}
```

```
## [1] "Omegas for n = 50 :"
## [1] "Mean =  0.735768163265306"
## [1] "Omegas for n = 100 :"
## [1] "Mean =  0.281840612244898"
## [1] "Omegas for n = 500 :"
## [1] "Mean =  0.0813257142857143"
```

```r
# Comparing penalized estimator to ML estimator

# Frobenius error
```

```r
# RMSE (sqrt(Mean^2 + Var))
for(i in 1:length(n)){
  print(paste("RMSE for n = ", n[i], ":"))
  print(paste("MLE :", sqrt(mean(out$FerrorsR_est[[paste("n =", toString(n[i]))]])^2 +
                              var(out$FerrorsR_est[[paste("n =", toString(n[i]))]]))))
  print(paste("covglasso :",
              sqrt(mean(out$FerrorsR_est_covglasso[[paste("n =", toString(n[i]))]])^2 +

              var(out$FerrorsR_est_covglasso[[paste("n =", toString(n[i]))]]))))
}
```

```
## [1] "RMSE for n =  50 :"
## [1] "MLE : 0.138388992856243"
## [1] "covglasso : 0.0839290349510492"
## [1] "RMSE for n =  100 :"
## [1] "MLE : 0.0973223803003858"
## [1] "covglasso : 0.0346085816027589"
## [1] "RMSE for n =  500 :"
## [1] "MLE : 0.0432697143918703"
## [1] "covglasso : 0.0172024686983185"
```

```r
# Mutual information

# RMSE
for(i in 1:length(n)){
    print(paste("RMSEs for n = ", n[i], ":"))
    print(paste("MLE :",
                sqrt((mean(normalize(out$MIests[[paste("n =", toString(n[i]))]])) -
                        normalize(MI_normal(R,dim)))^2 +
                    var(normalize(out$MIests[[paste("n =", toString(n[i]))]])))))
    print(paste("covglasso :",
          sqrt((mean(normalize(out$MIests_covglasso[[paste("n =", toString(n[i]))]]))
                  - normalize(MI_normal(R,dim)))^2 +
      var(normalize(out$MIests_covglasso[[paste("n =", toString(n[i]))]])))))
}
```

```
## [1] "RMSEs for n =  50 :"
## [1] "MLE : 0.0055403092544657"
## [1] "covglasso : 0.749115606472465"
## [1] "RMSEs for n =  100 :"
## [1] "MLE : 0.00438051867926876"
## [1] "covglasso : 0.0114768410684877"
## [1] "RMSEs for n =  500 :"
## [1] "MLE : 0.00137267952778447"
## [1] "covglasso : 0.00248161800536913"
```

```r
# Hellinger distance

# RMSE
for(i in 1:length(n)){
  print(paste("RMSEs for n = ", n[i], ":"))
  print(paste("MLE :",
              sqrt((mean(out$Helests[[paste("n =", toString(n[i]))]]) -
                      Hel_normal(R,dim))^2 +
                  var(out$Helests[[paste("n =", toString(n[i]))]])))))
```

```
  print(paste("covglasso :",
              sqrt((mean(out$Helests_covglasso[[paste("n =", toString(n[i]))]]) -
                      Hel_normal(R,dim))^2 +
                var(out$Helests_covglasso[[paste("n =", toString(n[i]))]])))))
}
```

```
## [1] "RMSEs for n =  50 :"
## [1] "MLE : 0.17778674716814"
## [1] "covglasso : 0.490793911897467"
## [1] "RMSEs for n =  100 :"
## [1] "MLE : 0.0773260244508615"
## [1] "covglasso : 0.103312503517874"
## [1] "RMSEs for n =  500 :"
## [1] "MLE : 0.0155327699137318"
## [1] "covglasso : 0.0337791026862663"
```

```
# BWD1

# RMSE
for(i in 1:length(n)){
  print(paste("RMSEs for n = ", n[i], ":"))
  print(paste("MLE :",
              sqrt((mean(out$BWD1ests[[paste("n =", toString(n[i]))]]) - BWD1(R,dim))^2 +

                             var(out$BWD1ests[[paste("n =", toString(n[i]))]])))))
  print(paste("covglasso :",
              sqrt((mean(out$BWD1ests_covglasso[[paste("n =", toString(n[i]))]]) -
                      BWD1(R,dim))^2 +

                var(out$BWD1ests_covglasso[[paste("n =", toString(n[i]))]])))))
}
```

```
## [1] "RMSEs for n =  50 :"
## [1] "MLE : 0.0651113556366089"
## [1] "covglasso : 0.05140388171475"
## [1] "RMSEs for n =  100 :"
## [1] "MLE : 0.0298558101761796"
## [1] "covglasso : 0.00958876216808385"
## [1] "RMSEs for n =  500 :"
## [1] "MLE : 0.00579300684576867"
## [1] "covglasso : 0.00353844230513182"
```

```
# BWD2

# RMSE
for(i in 1:length(n)){
  print(paste("RMSEs for n = ", n[i], ":"))
  print(paste("MLE :",
              sqrt((mean(out$BWD2ests[[paste("n =", toString(n[i]))]]) - BWD2(R,dim))^2 +

                             var(out$BWD2ests[[paste("n =", toString(n[i]))]])))))
  print(paste("covglasso :",
              sqrt((mean(out$BWD2ests_covglasso[[paste("n =", toString(n[i]))]]) -
                      BWD2(R,dim))^2 +
```

```
            var(out$BWD2ests_covglasso[[paste("n =", toString(n[i]))]])))))
}
```

```
## [1] "RMSEs for n =  50 :"
## [1] "MLE : 0.0639457631353639"
## [1] "covglasso : 0.051425993835766"
## [1] "RMSEs for n =  100 :"
## [1] "MLE : 0.0295282425593941"
## [1] "covglasso : 0.00974207622425762"
## [1] "RMSEs for n =  500 :"
## [1] "MLE : 0.00576089196379137"
## [1] "covglasso : 0.00364143805425726"
```

Misspecification

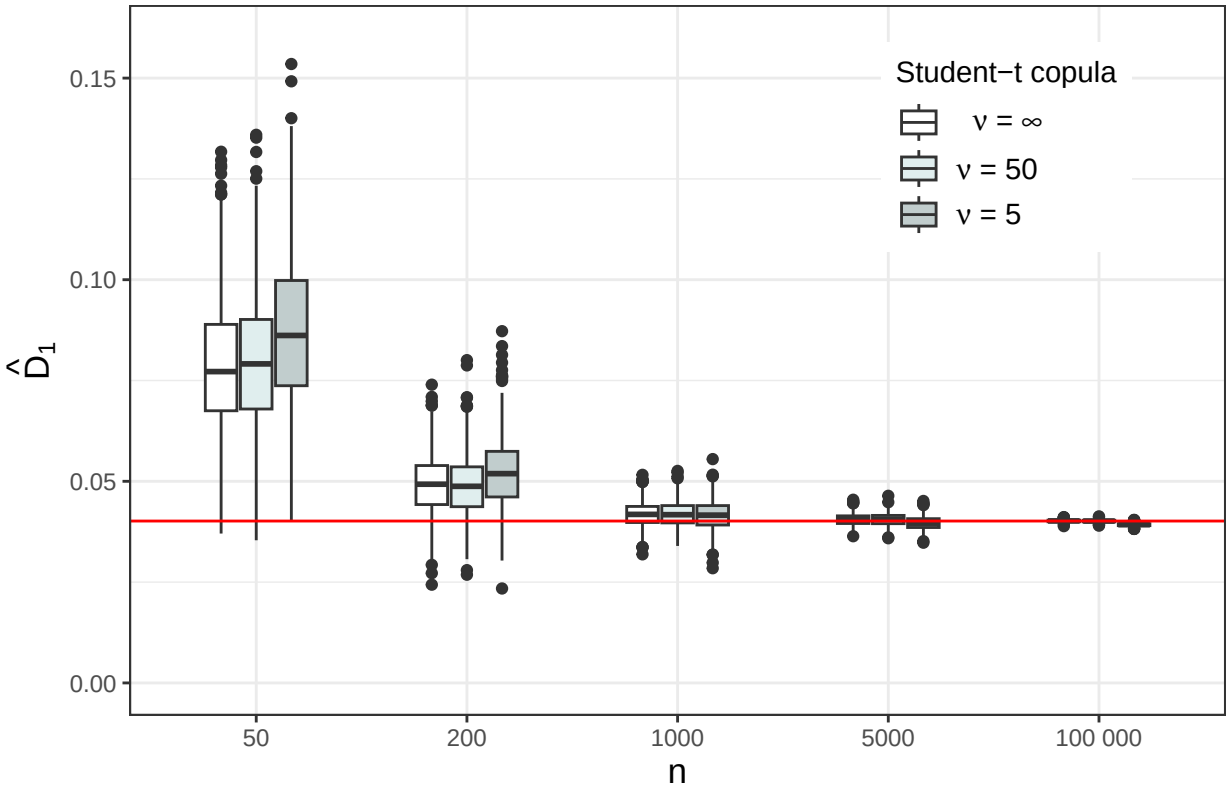
```
q = 10
dim = c(2,4,4)
R = 0.5^(abs(matrix(1:q-1, nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))
true = BWD1(R,dim)
Avar_true = BWD1_Avar(R,dim)^2

M = 1000
n = c(50,200,1000,5000,100000)
nu = 5
Q = function(t){qlst(t,nu,0,sqrt((nu-2)/nu))}

# VTS = list("True" = 0,"nu50Miss" = 0,"nu5Miss" = 0)

estimates = matrix(0,M,length(n))
```

```
# Simulation

set.seed(123)

for(i in 1:length(n)){
  for(j in 1:M){
    print(paste("i = ", i, "j = ", j))
    sample = mvtnorm::rmvt(n = n[i], sigma = ((nu-2)/nu) * R, df = nu)
    scores = matrix(0,n[i],q)
    for(l in 1:q){
      scores[,l] = qnorm((n[i]/(n[i]+1)) * ecdf(sample[,l])(sample[,l]))
    }
    R_est = cor(scores)
    estimates[j,i] = BWD1(R_est,dim)
  }
}

VTS$nu5Miss = estimates
# save(VTS, file = "Misspecification.Rdata") # Save data
```

```
load("Misspecification.Rdata")

frame = data.frame(matrix(nrow = 15000, ncol = 3))
colnames(frame) = c("est_dep","n","group")
```

```
frame$est_dep[1:1000] = VTS$True[,1]
frame$n[1:1000] = rep(50,1000)
frame$est_dep[1001:2000] = VTS$True[,2]
frame$n[1001:2000] = rep(200,1000)
frame$est_dep[2001:3000] = VTS$True[,3]
frame$n[2001:3000] = rep(1000,1000)
frame$est_dep[3001:4000] = VTS$True[,4]
frame$n[3001:4000] = rep(5000,1000)
frame$est_dep[4001:5000] = VTS$True[,5]
frame$n[4001:5000] = rep(100000,1000)

frame$est_dep[5001:6000] = VTS$nu50Miss[,1]
frame$n[5001:6000] = rep(50,1000)
frame$est_dep[6001:7000] = VTS$nu50Miss[,2]
frame$n[6001:7000] = rep(200,1000)
frame$est_dep[7001:8000] = VTS$nu50Miss[,3]
frame$n[7001:8000] = rep(1000,1000)
frame$est_dep[8001:9000] = VTS$nu50Miss[,4]
frame$n[8001:9000] = rep(5000,1000)
frame$est_dep[9001:10000] = VTS$nu50Miss[,5]
frame$n[9001:10000] = rep(100000,1000)

frame$est_dep[10001:11000] = VTS$nu5Miss[,1]
frame$n[10001:11000] = rep(50,1000)
frame$est_dep[11001:12000] = VTS$nu5Miss[,2]
frame$n[11001:12000] = rep(200,1000)
frame$est_dep[12001:13000] = VTS$nu5Miss[,3]
frame$n[12001:13000] = rep(1000,1000)
frame$est_dep[13001:14000] = VTS$nu5Miss[,4]
frame$n[13001:14000] = rep(5000,1000)
frame$est_dep[14001:15000] = VTS$nu5Miss[,5]
frame$n[14001:15000] = rep(100000,1000)


frame$group[1:5000] = rep(1,5000)
frame$group[5001:10000] = rep(2,5000)
frame$group[10001:15000] = rep(3,5000)
frame$n = as.factor(frame$n)
frame$group = as.factor(frame$group)


ggplot(frame, aes(x=n, y=est_dep, fill = group)) +
  scale_y_continuous(labels = label_number(accuracy = 0.01),limits = c(0,0.16)) +
  geom_boxplot(width=0.5) + labs(title="", y = expression(hat(D)[1])) +
  theme_bw() + theme(plot.title = element_text(hjust=0.5, size = 14),
                     axis.title = element_text(size=13)) +
  scale_fill_manual(name = "Student-t copula", values = c("white",
                    labels = c(expression("  " * nu * " = " * infinity * "   "),
                               expression(nu * " = 50" * "  "),
                               expression(nu * " = 5" * "   "))) +
  guides(colour = guide_legend(title.position = "top", keywidth = unit(1, "cm"),
                               keyheight = unit(1, "cm"),
                               override.aes = list(shape = 22,size = 10))) +
```

```
  theme(legend.position = c(0.8, 0.8),legend.text = element_text(size = 11)) +
  scale_x_discrete(labels=c("50","200","1000","5000","100 000")) +
  geom_hline(yintercept = 0.04015, col = "red")
```




```
sqrt(n) * (apply(VTS$True,2,mean) - true)
```

```
## [1] 0.274308095 0.128371473 0.052040573 0.023769830 0.005888278
```

```
sqrt(n) * (apply(VTS$nu50Miss,2,mean) - true)
```

```
## [1]  0.279695351  0.125947942  0.054042976  0.027260043 -0.002025523
```

```
sqrt(n) * (apply(VTS$nu5Miss,2,mean) - true)
```

```
## [1]  0.33323998  0.17148796  0.04694466 -0.03528609 -0.28453195
```

```
n * apply(VTS$True,2,var)
```

```
## [1] 0.012907036 0.010451839 0.009137754 0.009875638 0.009440390
```

```
n * apply(VTS$nu50Miss,2,var)
```

```
## [1] 0.013387974 0.011249215 0.009824957 0.010631644 0.009210887
```

```
n * apply(VTS$nu5Miss,2,var)
```

```
## [1] 0.01845093 0.01416199 0.01302047 0.01351048 0.01342563
```