Supplementary Methods

Minta Kärkkäinen & Joonas Tuomikoski

2024

Contents

Introduction

This R-script supplementary file was used to compute computational analyses related to the research. Here we present the two main analyses of the study. The Weighted Correlation Network Analysis was used to study omics-level co-expression networks. The Cox Regression Survival Model was used to study cancer-predicting biomarkers among multi-omics datasets and to test the predictive accuracy of the found biomarkers.

Contact Information

The script has been written by Minta Kärkkäinen (Uni. of Jyväskylä) and Joonas Tuomikoski (Uni. of Jyväskylä). Correspondence to: minta.e.m.karkkainen@jyu.fi or tiina.a.jokela@jyu.fi

Weighted Correlation Network Analysis

Install packages and import data

```
## Step1: Install needed packages and import data (cmiR)

library(magrittr)    #provides the %>% operator
library(WGCNA)
library(GO.db)

#Load normalized cmiR countdata
cmiR <- read.table("normalized_miR_counts.txt")
```

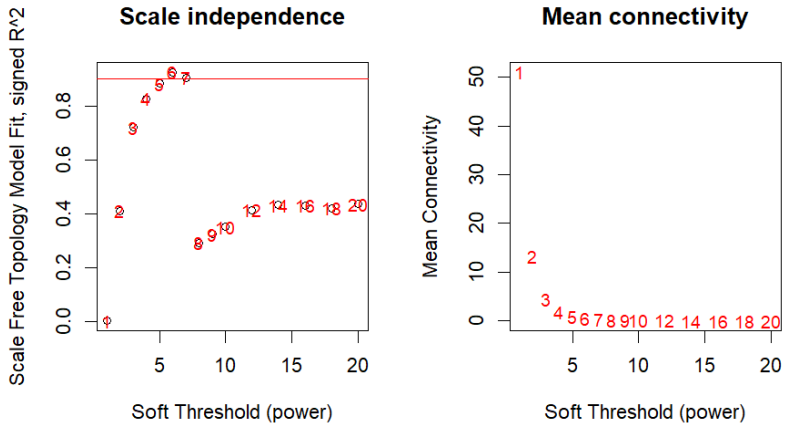
Determine Soft threshold

```
##Step 2. Pick soft threshold for WGCNA

#When you pick up a soft threshold it should only contain expression values.
 A correlation network will be a complete network (all genes are connected to all other genes).
 We will need to pick a threshold value (if the correlation is below threshold, remove the edge).
To do that, WGCNA will try a range of soft thresholds and create a diagnostic plot:

allowWGCNAThreads() #optional, allows few threads

# Choose a set of soft-thresholding powers
powers = c(c(1:10), seq(from = 12, to = 20, by = 2))

# Call the network topology analysis function
sft = pickSoftThreshold(
  cmiR,
  powerVector = powers,
  verbose = 5
)

par(mfrow = c(1,2));
cex1 = 0.9;
```

```r
plot(sft$fitIndices[, 1],
    -sign(sft$fitIndices[, 3]) * sft$fitIndices[, 2],
    xlab = "Soft Threshold (power)",
    ylab = "Scale Free Topology Model Fit, signed R^2",
    main = paste("Scale independence")
)
text(sft$fitIndices[, 1],
    -sign(sft$fitIndices[, 3]) * sft$fitIndices[, 2],
    labels = powers, cex = cex1, col = "red"
)
abline(h = 0.90, col = "red")
plot(sft$fitIndices[, 1],
    sft$fitIndices[, 5],
    xlab = "Soft Threshold (power)",
    ylab = "Mean Connectivity",
    type = "n",
    main = paste("Mean connectivity")
)
text(sft$fitIndices[, 1],
    sft$fitIndices[, 5],
    labels = powers,
    cex = cex1, col = "red")
```

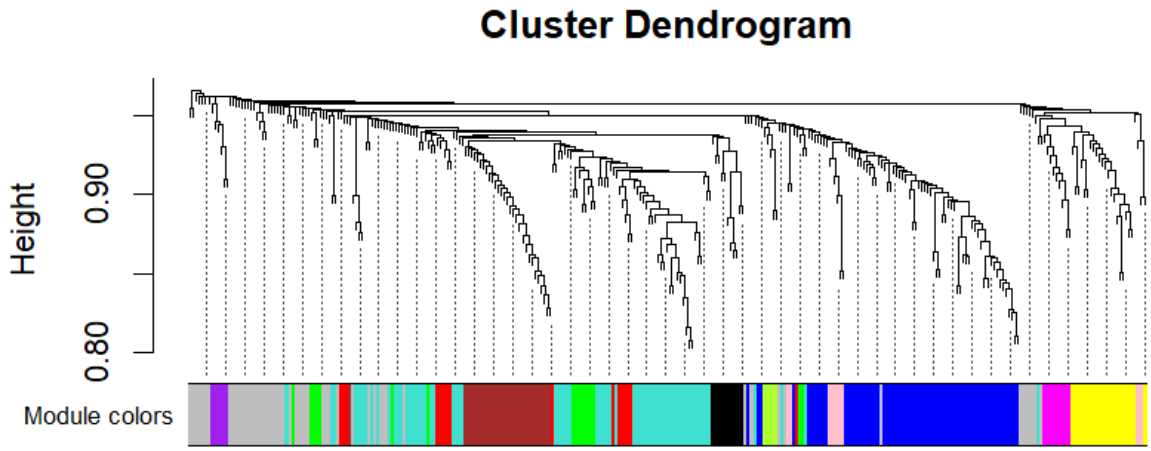
#The Scale-Free Topology Model Fit (signed R^2) plot generated by pickSoftThreshold in WGCNA is used to assess the goodness of fit of the network to a scale-free topology under different soft-thresholding powers.
Ideally, you want to choose a soft-thresholding power at which the model fit (R^2) is high and the curve reaches a plateau, indicating that the network follows a scale-free topology.
If your Scale-Free Topology Model Fit plot shows a scattered or irregular pattern with data points it may indicate that the data does not exhibit a clear scale-free network structure across the tested soft-thresholding powers.

#Here, we chose soft thresholding power of 6:



WGCNA

```
##Step 3. Compute Weighted Correlation Networks


picked_power = 6     # <= selected soft threshold here
temp_cor <- cor
cor <- function(...) WGCNA::cor(method = "spearman", ...) # force it to use WGCNA cor function
                                                           (fix a namespace conflict issue)
netwk <- blockwiseModules(cmiR,              # <= input here

                          # == Adjacency Function ==
                          power = picked_power, # <= power here
                          networkType = "signed", #allows positive and negative correlations

                          # == Tree and Block Options ==
                          deepSplit = 2,
                          pamRespectsDendro = F,
                          # detectCutHeight = 0.75,
                          minModuleSize = 5,    #determine minimum module size
                          maxBlockSize = 317,   #max module size was set as the number of cmiRs in the data

                          # == Module Adjustments ==
                          reassignThreshold = 0,
                          mergeCutHeight = 0.25,

                          # == TOM == Archive the run results in TOM file
                          saveTOMs = T,
                          saveTOMFileBase = "ER",

                          # == Output Options
                          numericLabels = T,
                          verbose = 3)

cor <- temp_cor     # return cor function to original namespace

# Convert labels to colors for plotting
mergedColors = labels2colors(netwk$colors)
# Plot the dendrogram and the module colors underneath
plotDendroAndColors(
  netwk$dendrograms[[1]],
  mergedColors[netwk$blockGenes[[1]]],
  "Module colors",
  dendroLabels = FALSE,
  hang = 0.03,
  addGuide = TRUE,
  guideHang = 0.05 )
```




```
#We have written a tab-delimited file listing the genes and their modules.
 WGCNA will calculate an Eigengene (hypothetical central gene) for each module, so it is easier
 to determine if modules are associated with a trait of interest.


module_df <- data.frame(
  gene_id = names(netwk$colors),
  colors = labels2colors(netwk$colors)
)

# Get Module Eigengenes per cluster
MEs0 <- moduleEigengenes(cmiR, mergedColors)$eigengenes

# Reorder modules so similar modules are next to each other
MEs0 <- orderMEs(MEs0)
module_order = names(MEs0) %>% gsub("ME","", .)


#We have now calculated module eigengenes (MEs) for each study subject which can now be used in
module-trait association analyses.
```

Lasso Cox regression survival model

Feature selection using Lasso Cox regression

```
#Step 1. Download packages and import data

#Install libraries
library(glmnet)
library(survival)
library(mixOmics) # to impute missing values
library(SurvMetrics) # to get all the metrics
library(pec) # to make predictions based on the Cox model
#Import data
totalx <- read.delim("multi_omics_Filtered2.txt", header = TRUE)

#Impute missing values, NIPALS is used to decompose the dataset.
totalx <- impute.nipals(X = totalx, ncomp = 10)


#Step 2. Feature selection for cmiRs

# Extract predictors (cmiR expression levels)
x0 <- data.matrix(totalx[ ,c(7:21)]) %>%
  scale(center = T) %>%
  na.omit()

# Extract response variable (time and status)
y0 <- totalx[,c(5,6)] %>%
  na.omit() %>%
  as.matrix()

# Fit the LASSO model (Lasso: Alpha = 1)

# Inspect Lasso  fit lambdas
fit0 <- glmnet(x0,y0, family = "cox", alpha = 1, maxit=1000000)
plot(fit0, xvar="lambda")
print(fit0)
lambda0 <- coef(fit0, s = 0.040680)  #select Lambda where the number of features is thresholded to 5

#The selected features and their coefficients can be obtained:
nonZeroIdx0<-which(lambda0[,1]!= 0)
```

```
features0<-rownames(lambda0)[nonZeroIdx0]
features0

#Step 3. Feature selection for cMets

# Extract predictors
x1 <- data.matrix(totalx[ ,c(22:85)]) %>%
  scale(center = T) %>%
  na.omit()

# Inspect Lasso fit lambdas
fit1 <- glmnet(x1,y0, family = "cox", alpha = 1, maxit=1000000)
plot(fit1, xvar="lambda")
print(fit1)
lambda1 <- coef(fit1, s = 0.051400)

nonZeroIdx1<-which(lambda1[,1]!= 0)
features1<-rownames(lambda1)[nonZeroIdx1]
features1
```

Fit Cox regression model

```
#Step 1. Fit the cmiRs and cMets to Cox regression model using predictive features from the feature selection

#Scale cmiRs and cMets
total1[, -c(1, 2)] <- scale(total1[, -c(1, 2)],
                            center = TRUE,
                            scale = TRUE)
#Fit the model
full.cox <- coxph(Surv(time,status) ~ hsa.miR.101.3p + hsa.miR.182.5p + hsa.miR.183.5p + hsa.miR.4732.3p + hsa.miR.148b.3p + HDL_TG + Tyr + Glucose + Acetate + GlycA, data = total1, x=TRUE)

summary(full.cox)
## Call:
## coxph(formula = Surv(time, status) ~ hsa.miR.101.3p + hsa.miR.182.5p +
##                          hsa.miR.183.5p + hsa.miR.4732.3p + hsa.miR.148b.3p + HDL_TG +
##                          Tyr + Glucose + Acetate + GlycA, data = total1, x = TRUE)
##
## n= 116, number of events= 17
##
##                          coef      exp(coef) se(coef)    z      Pr(>|z|)
## hsa.miR.101.3p        0.54142   1.71844  0.34455  1.571    0.116
## hsa.miR.182.5p        0.23665   1.26699  0.40600  0.583    0.560
## hsa.miR.183.5p        0.22955   1.25803  0.45463  0.505    0.614
```

```
## hsa.miR.4732.3p      -0.01475   0.98536  0.21581 -0.068    0.946
## hsa.miR.148b.3p      -0.47138   0.62414  0.33372 -1.413    0.158
## HDL_TG               0.46815   1.59703  0.32727  1.430    0.153
## Tyr                  0.10789   1.11392  0.28337  0.381    0.703
## Glucose              0.44125   1.55465  0.29130  1.515    0.130
## Acetate              0.44192   1.55569  0.28270  1.563    0.118
## GlycA                0.53485   1.70719  0.35023  1.527    0.127
##
##                      exp(coef) exp(-coef) lower .95 upper .95
## hsa.miR.101.3p       1.7184     0.5819    0.8747     3.376
## hsa.miR.182.5p       1.2670     0.7893    0.5717     2.808
## hsa.miR.183.5p       1.2580     0.7949    0.5161     3.067
## hsa.miR.4732.3p      0.9854     1.0149    0.6455     1.504
## hsa.miR.148b.3p      0.6241     1.6022    0.3245     1.200
## HDL_TG               1.5970     0.6262    0.8409     3.033
## Tyr                  1.1139     0.8977    0.6392     1.941
## Glucose              1.5546     0.6432    0.8784     2.752
## Acetate              1.5557     0.6428    0.8939     2.707
## GlycA                1.7072     0.5858    0.8593     3.392
##
## Concordance= 0.815  (se = 0.045 )
## Likelihood ratio test= 26.78  on 10 df,   p=0.003
## Wald test = 22.08  on 10 df,   p=0.01
## Score (logrank) test = 24.72  on 10 df,   p=0.006

#Test the model assumptions using Schoenfeld residuals
cox.zph(full.cox)

#Remove the least significant covariates
anova(full.cox)
## Analysis of Deviance Table
## Cox model: response is Surv(time, status)
## Terms added sequentially (first to last)
##
##                      loglik  Chisq Df Pr(>|Chi|)
## NULL                 -78.946
## hsa.miR.101.3p       -76.029 5.8342  1   0.015718 *
## hsa.miR.182.5p       -74.995 2.0677  1   0.150444
## hsa.miR.183.5p       -73.605 2.7811  1   0.095382 .
## hsa.miR.4732.3p      -72.922 1.3662  1   0.242474
## hsa.miR.148b.3p      -72.768 0.3070  1   0.579549
## HDL_TG               -69.110 7.3166  1   0.006832 **
## Tyr                  -68.673 0.8744  1   0.349729
## Glucose              -67.655 2.0343  1   0.153787
## Acetate              -66.725 1.8610  1   0.172515
## GlycA                -65.555 2.3400  1   0.126091
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

#Plot the results
ggforest(full.cox, data = total1)
```

```
#Step 2. Fit the model again using covariates based on Anova results (highest Chisq-values)
 full.cox.1 <- coxph(Surv(time,status) ~ hsa.miR.101.3p + hsa.miR.183.5p + HDL_TG, data = total1, x= TRUE)

summary(full.cox.1)
## Call:
## coxph(formula = Surv(time, status) ~ hsa.miR.101.3p + hsa.miR.183.5p +
##                                      HDL_TG, data = total1, x = TRUE)
##
##  n= 116, number of events= 17
##
##                          coef      exp(coef) se(coef)   z     Pr(>|z|)
## hsa.miR.101.3p        0.7000    2.0137   0.3038 2.304   0.0212 *
## hsa.miR.183.5p        0.5811    1.7881   0.2809 2.069   0.0385 *
## HDL_TG                0.6876    1.9889   0.2758 2.493   0.0127 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                          exp(coef) exp(-coef) lower .95 upper .95
## hsa.miR.101.3p        2.014     0.4966     1.110     3.652
## hsa.miR.183.5p        1.788     0.5593     1.031     3.101
## HDL_TG                1.989     0.5028     1.158     3.415
##
## Concordance= 0.764  (se = 0.04 )
## Likelihood ratio test= 17.15  on 3 df,   p=7e-04
## Wald test = 13.81  on 3 df,   p=0.003
## Score (logrank) test = 15.34  on 3 df,   p=0.002

anova(full.cox.1)
## Analysis of Deviance Table
##  Cox model: response is Surv(time, status)
## Terms added sequentially (first to last)
##
##                          loglik    Chisq    Df Pr(>|Chi|)
## NULL                  -78.946
## hsa.miR.101.3p        -76.029 5.8342  1    0.01572 *
## hsa.miR.183.5p        -73.607 4.8440  1    0.02774 *
## HDL_TG                -70.372 6.4697  1    0.01097 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

#Test the model assumptions using Schoenfeld residuals
cox.zph(full.cox.1)
```

```
#Step 3. Train the full and reduced model for CRC

#Make data frame containing only CRC
total2 <- total1[!(rownames(total1) %in% c("LSME0160", "LSME0152", "LSME0105", "LSME0095",
"LSME0078", "LSME0045", "LSME0039", "LSME0004", "LSJ126")), ]

#Fit the model for CRC with all 10 predictors
full.cox.CRC <- coxph(Surv(time,status) ~ hsa.miR.101.3p + hsa.miR.182.5p + hsa.miR.183.5p +
                                          hsa.miR.4732.3p + hsa.miR.148b.3p + HDL_TG + Tyr +
                                          Glucose + Acetate + GlycA, data = total2, x=TRUE)
summary(full.cox.CRC)
## Call:
## coxph(formula = Surv(time, status) ~ hsa.miR.101.3p + hsa.miR.182.5p +
##                                       hsa.miR.183.5p + hsa.miR.4732.3p + hsa.miR.148b.3p +
##                                      HDL_TG + Tyr + Glucose + Acetate + GlycA,
##                                      data = total2, x = TRUE)
##   n= 107, number of events= 8
##
##                       coef      exp(coef) se(coef)  z      Pr(>|z|)
## hsa.miR.101.3p       1.9688    7.1619   0.8532  2.307   0.0210 *
##hsa.miR.182.5p        -1.3121   0.2693   1.0609 -1.237  0.2162
## hsa.miR.183.5p       0.2898    3.6319   1.3106  0.984   0.3251
## hsa.miR.4732.3p      0.5705    1.7691   0.5246  1.087   0.2769
## hsa.miR.148b.3p      -0.9391   0.3910   0.5242 -1.863   0.0625 .
## HDL_TG               0.2666    1.3055   0.5270  0.506   0.6130
## Tyr                  0.3678    1.4446   0.4687  0.785   0.4326
## Glucose              1.2748    3.5779   0.6353  2.006   0.0448 *
## Acetate              0.9691    2.6357   0.5331  1.818   0.0691 .
## GlycA                0.8590    2.3608   0.5233  1.642   0.1007
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                      exp(coef) exp(-coef) lower .95 upper .95
## hsa.miR.101.3p       7.1619    0.1396    1.34508   38.133
##hsa.miR.182.5p        0.2693    3.7140    0.03366   2.154
## hsa.miR.183.5p       3.6319    0.2753    0.27831   47.395
## hsa.miR.4732.3p      1.7691    0.5653    0.63270   4.946
## hsa.miR.148b.3p      0.3910    2.5576    0.14555   1.050
## HDL_TG               1.3055    0.7660    0.46468   3.668
## Tyr                  1.4446    0.6922    0.57648   3.620
## Glucose              3.5779    0.2795   1.02997    12.429
## Acetate              2.6357    0.3794    0.92715   7.493
## GlycA                2.3608    0.4236    0.84654   6.584
##
## Concordance= 0.898  (se = 0.041 )
## Likelihood ratio test= 21.01  on 10 df,   p=0.02
## Wald test = 9.77  on 10 df,   p=0.5
## Score (logrank) test = 14.56  on 10 df,   p=0.1
```

```
anova(full.cox.CRC)
ggforest(full.cox.CRC, data = total2)
cox.zph(full.cox.CRC)

#Fit the model for CRC with 3 predictors
full.cox.CRC2 <- coxph(Surv(time,status) ~ hsa.miR.101.3p + hsa.miR.183.5p + HDL_TG, data = total2
, x=TRUE)

summary(full.cox.CRC2)
## Call:
## coxph(formula = Surv(time, status) ~ hsa.miR.101.3p + hsa.miR.183.5p +
##                                          HDL_TG, data = total2, x = TRUE)
##
##  n= 107, number of events= 8
##
##                       coef     exp(coef)   se(coef)  z      Pr(>|z|)
## hsa.miR.101.3p       0.9491    2.5834   0.4441 2.137   0.0326 *
## hsa.miR.183.5p       0.4801    1.6162   0.3980 1.206   0.2278
## HDL_TG               0.4067    1.5018   0.4292 0.948   0.3434
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                       exp(coef) exp(-coef) lower .95 upper .95
## hsa.miR.101.3p       2.583       0.3871    1.0819     6.169
## hsa.miR.183.5p       1.616       0.6187    0.7408     3.526
## HDL_TG               1.502       0.6659    0.6476     3.483
##
## Concordance= 0.796  (se = 0.045 )
## Likelihood ratio test= 8.32  on 3 df,   p=0.04
## Wald test = 6.88  on 3 df,   p=0.08
## Score (logrank) test = 7.45  on 3 df,   p=0.06

anova(full.cox.CRC2)
cox.zph(full.cox.CRC2)
ggforest(full.cox.CRC2, data = total2)
```

Model validation

```
#Step 1. Import 5 data splits

train0 <- read.csv("train_set_0.csv")
train1 <- read.csv("train_set_1.csv")
train2 <- read.csv("train_set_2.csv")
train3 <- read.csv("train_set_3.csv")
train4 <- read.csv("train_set_4.csv")


#Step 2. Fit the Cox Proportional Hazards Model on train data and make predictions
with the test data

Iteration 0:

# Standardize the data (excluding the response variables)
train_data_scaled0 <- scale(train_data0[, -c(1, 2)],
                            center = TRUE,
                            scale = TRUE)
train_means0 <- attr(train_data_scaled0, "scaled:center") #save train means for scaling the test data
train_sds0 <- attr(train_data_scaled0, "scaled:scale") #save train sds for scaling the test data

#Convert the scaled matrix to a data frame
train_data_scaled0 <- as.data.frame(train_data_scaled0)

#Add the unscaled 'time' and 'status' variables back to the scaled data frame
train_data_scaled0 <- cbind(train_data0[, c("time", "status")], train_data_scaled0)

#Train the model with predictive features
full.cox0 <- coxph(Surv(time,status) ~ hsa.miR.101.3p + hsa.miR.183.5p + HDL_TG, data = train_data_
scaled0, x=TRUE)

summary(full.cox0)
cox.zph(full.cox0)
ggforest(full.cox0, data = train_data_scaled0)

#Make model predictions with test data
test_data0 <- read.csv("test_set_0.csv")

## Scale the test data using the mean and sd from the training data
test_data0[, -c(1, 2)] <- scale(test_data0[, -c(1, 2)],
                                center = train_means0,
                                scale = train_sds0)


#Use the SurvMetrics package to validate the predictions
event_times <- full.cox0$y[, "time"]
```

```
event_indicator <- full.cox0$y[, "status"]  # this indicates if the event occurred (1) or was censored (0)
distime <- event_times[event_indicator == 1]
distime <- sort(unique(distime))
mat_cox <- predictSurvProb(full.cox0, test_data0, distime) #get the survival probability matrix
med_index <- median(1:length(distime)) #the index of median survival time of events
vec_cox <- mat_cox[ ,med_index]
vec_cox  # print probabilities
```

```
times <- test_data0$time  #extract survival time of events
status <- test_data0$status #extract survival status of events

#CI BS IBS IAE ISE based on Cox model: Non-standard model input methods
Cindex_cox <- Cindex(Surv(times, status), vec_cox)
BS_cox <- Brier(Surv(times, status), vec_cox, distime[med_index])
IBS_cox <- IBS(Surv(times, status), mat_cox, distime)
IAE_cox <- IAEISE(Surv(times, status), mat_cox, distime)[1]
ISE_cox <- IAEISE(Surv(times, status), mat_cox, distime)[2]
```

Iteration 1:

```
train_data_scaled1 <- scale(train_data1[, -c(1, 2)],
                            center = TRUE,
                            scale = TRUE)
train_means1 <- attr(train_data_scaled1, "scaled:center")
train_sds1 <- attr(train_data_scaled1, "scaled:scale")
train_data_scaled1 <- as.data.frame(train_data_scaled1)
train_data_scaled1 <- cbind(train_data1[, c("time", "status")], train_data_scaled1)

#Train the model
full.cox1 <- coxph(Surv(time,status) ~ hsa.miR.101.3p + hsa.miR.183.5p + HDL_TG, data = train_data_
scaled1, x=TRUE)

summary(full.cox1)
cox.zph(full.cox1)

#Plot predictive features' impact on event hazard
ggforest(full.cox1, data = train_data_scaled1)
```

```
#Make model predictions with test data
test_data1 <- read.csv("test_set_1.csv")

#Scale the test data using the mean and sd from the training data
test_data1[, -c(1, 2)] <- scale(test_data1[, -c(1, 2)],
                                center = train_means1,
                                scale = train_sds1)

# Extract event times and event indicators
```

```
event_times1 <- full.cox1$y[, "time"]
event_indicator1 <- full.cox1$y[, "status"]
distime1 <- event_times1[event_indicator1 == 1]
distime1 <- sort(unique(distime1))
mat_cox1 <- predictSurvProb(full.cox1, test_data1, distime1)
med_index1 <- median(1:length(distime1))
vec_cox1 <- mat_cox1[ ,med_index1]

times1 <- test_data1$time
status1 <- test_data1$status

#CI BS IBS IAE ISE based on Cox model: Non-standard model input methods
Cindex_cox1 <- Cindex(Surv(times1, status1), vec_cox1)
BS_cox1 <- Brier(Surv(times1, status1), vec_cox1, distime1[med_index1])
IBS_cox1 <- IBS(Surv(times1, status1), mat_cox1, distime1)
IAE_cox1 <- IAEISE(Surv(times1, status1), mat_cox1, distime1)[1]
ISE_cox1 <- IAEISE(Surv(times1, status1), mat_cox1, distime1)[2]
```

Iteration 2:

```
# Standardize the data (excluding the response variable)
train_data_scaled2 <- scale(train_data2[, -c(1, 2)],
                            center = TRUE,
                            scale = TRUE)
train_means2 <- attr(train_data_scaled2, "scaled:center")
train_sds2 <- attr(train_data_scaled2, "scaled:scale")
train_data_scaled2 <- as.data.frame(train_data_scaled2)
train_data_scaled2 <- cbind(train_data2[, c("time", "status")], train_data_scaled2)

#Train the model with predictive features
full.cox2 <- coxph(Surv(time,status) ~ hsa.miR.101.3p + hsa.miR.183.5p + HDL_TG,
                                       data = train_data_ scaled2, x=TRUE)

summary(full.cox2)
cox.zph(full.cox2)
ggforest(full.cox2, data = train_data_scaled2)

#Make model predictions with test data
test_data2 <- read.csv("test_set_2.csv")

#Scale the test data using the mean and sd from the training data
test_data2[, -c(1, 2)] <- scale(test_data2[, -c(1, 2)],
                                center = train_means2,
                                scale = train_sds2)

# Extract event times and event indicators
event_times2 <- full.cox2$y[, "time"]
event_indicator2 <- full.cox2$y[, "status"]
distime2 <- event_times2[event_indicator2 == 1]
```

```
distime2 <- sort(unique(distime2))
mat_cox2 <- predictSurvProb(full.cox2, test_data2, distime2)
med_index2 <- median(1:length(distime2))
vec_cox2 <- mat_cox2[ ,med_index2]

times2 <- test_data2$time
status2 <- test_data2$status

#CI BS IBS IAE ISE based on Cox model: Non-standard model input methods
Cindex_cox2 <- Cindex(Surv(times2, status2), vec_cox2)
BS_cox2 <- Brier(Surv(times2, status2), vec_cox2, distime2[med_index2])
IBS_cox2 <- IBS(Surv(times2, status2), mat_cox2, distime2)
IAE_cox2 <- IAEISE(Surv(times2, status2), mat_cox2, distime2)[1]
ISE_cox2 <- IAEISE(Surv(times2, status2), mat_cox2, distime2)[2]
```

Iteration 3:

```
# Standardize the data (excluding the response variable)
train_data_scaled3 <- scale(train_data3[, -c(1, 2)],
                            center = TRUE,
                            scale = TRUE)
train_means3 <- attr(train_data_scaled3, "scaled:center")
train_sds3 <- attr(train_data_scaled3, "scaled:scale")
train_data_scaled3 <- as.data.frame(train_data_scaled3)
train_data_scaled3 <- cbind(train_data3[, c("time", "status")], train_data_scaled3)

#Train the model with predictive features
full.cox3 <- coxph(Surv(time,status) ~ hsa.miR.101.3p + hsa.miR.183.5p + HDL_TG,
                                       data = train_data_scaled3, x=TRUE)

summary(full.cox3)
cox.zph(full.cox3)
ggforest(full.cox3, data = train_data_scaled3)
```

```
#Make model predictions with test data
test_data3 <- read.csv("test_set_3.csv")

#Scale the test data using the mean and sd from the training data
test_data3[, -c(1, 2)] <- scale(test_data3[, -c(1, 2)],
                                center = train_means3,
                                scale = train_sds3)

# Extract event times and event indicators
event_times3 <- full.cox3$y[, "time"]
event_indicator3 <- full.cox3$y[, "status"]
distime3 <- event_times3[event_indicator3 == 1]
distime3 <- sort(unique(distime3))
```

```
mat_cox3 <- predictSurvProb(full.cox3, test_data3, distime3)
med_index3 <- median(1:length(distime3))
vec_cox3 <- mat_cox3[ ,med_index3]

times3 <- test_data3$time
status3 <- test_data3$status

#CI BS IBS IAE ISE based on Cox model: Non-standard model input methods
Cindex_cox3 <- Cindex(Surv(times3, status3), vec_cox3)
BS_cox3 <- Brier(Surv(times3, status3), vec_cox3, distime3[med_index3])
IBS_cox3 <- IBS(Surv(times3, status3), mat_cox3, distime3)
IAE_cox3 <- IAEISE(Surv(times3, status3), mat_cox3, distime3)[1]
ISE_cox3 <- IAEISE(Surv(times3, status3), mat_cox3, distime3)[2]
```

Iteration 4:

```
# Standardize the data (excluding the response variable)
train_data_scaled4 <- scale(train_data4[, -c(1, 2)],
                            center = TRUE,
                            scale = TRUE)
train_means4 <- attr(train_data_scaled4, "scaled:center")
train_sds4 <- attr(train_data_scaled4, "scaled:scale")
train_data_scaled4 <- as.data.frame(train_data_scaled4)
train_data_scaled4 <- cbind(train_data4[, c("time", "status")], train_data_scaled4)

#Train the model with predictive features
full.cox4 <- coxph(Surv(time,status) ~ hsa.miR.101.3p + hsa.miR.183.5p + HDL_TG,
                                       data = train_data_scaled4, x=TRUE)

summary(full.cox4)
cox.zph(full.cox4)
ggforest(full.cox4, data = train_data_scaled4)
```

```
#Make model predictions with test data
test_data4 <- read.csv("test_set_4.csv")

#Scale the test data using the mean and sd from the training data
test_data4[, -c(1, 2)] <- scale(test_data4[, -c(1, 2)],
                                center = train_means4,
                                scale = train_sds4)

# Extract event times and event indicators
event_times4 <- full.cox4$y[, "time"]
event_indicator4 <- full.cox4$y[, "status"]
distime4 <- event_times4[event_indicator4 == 1]
distime4 <- sort(unique(distime4))
```

```
mat_cox4 <- predictSurvProb(full.cox4, test_data4, distime4)
med_index4 <- median(1:length(distime4))
vec_cox4 <- mat_cox4[ ,med_index4]

times4 <- test_data4$time
status4 <- test_data4$status

#CI BS IBS IAE ISE based on Cox model: Non-standard model input methods
Cindex_cox4 <- Cindex(Surv(times4, status4), vec_cox4)
BS_cox4 <- Brier(Surv(times4, status4), vec_cox4, distime4[med_index4])
IBS_cox4 <- IBS(Surv(times4, status4), mat_cox4, distime4)
IAE_cox4 <- IAEISE(Surv(times4, status4), mat_cox4, distime4)[1]
ISE_cox4 <- IAEISE(Surv(times4, status4), mat_cox4, distime4)[2]
```
