DEXA Study Rmarkdown

Adria Regue Alsina

04/01/2021

Used packages

```
if(!require("haven")) install.packages("haven")
if(!require("rstudioapi")) install.packages("rstudioapi")
if(!require("readxl")) install.packages("readxl")
if(!require("knitr")) install.packages("knitr")
if(!require("ggplot2")) install.packages("ggplot2")
if(!require("plyr")) install.packages("plyr")
if(!require("dplyr")) install.packages("dplyr")
if(!require("corrplot")) install.packages("corrplot")
if(!require("devtools")) install.packages("devtools")
if(!require("ggbiplot")) install_github("vqv/ggbiplot")
if(!require("gridExtra")) install.packages("gridExtra")
if(!require("car")) install.packages("car")
if(!require("randomForest")) install.packages("randomForest")
if(!require("caret")) install.packages("caret")
if(!require("ROCR")) install.packages("ROCR")
if(!require("randomForestExplainer")) nstall.packages("randomForestExplainer")
if(!require("factoextra")) install.packages("factoextra")
if(!require("pcalg")) install.packages("pcalg")
if(!require("qgraph")) install.packages("qgraph")
if(!require("ggm")) install.packages("ggm")
if(!require("Rgraphviz")) install.packages("Rgraphviz")
if(!require("RColorBrewer")) install.packages("RColorBrewer")
if(!require("mgm")) install.packages("mgm")
if(!require("UBL")) install.packages("UBL")
if(!require("kernlab")) install.packages("kernlab")
if(!require("class")) install.packages("class")
```

Loading data

In the following study we will be working with a dataframe from the "Lluita contra la SIDA" foundation. The data is in ".sav" format and will be read and imported using `read_sav{haven}`.

```
# Read archive
path <- file.path("E:/Arxius Adria Regue/UOC/tercer semestre/TFM/Docs Drive/Datos",
                  name = "data DEXA.sav")
dexa <- read_sav(path); rm(path) # read_delim for ".dat"
```

```r
# Changing order of the variables
dexa <- dexa[, c("ID", "gender", "gender_num", "Age", "Age_cat", "Height", "Weight",
                 "RAFp", "RAFg", "RALg", "LAFp", "LAFg", "LALg", "BothAFp", "BothAFg",
                 "BothALg", "RLFp", "RLFg", "RLLg", "LLFp", "LLFg", "LLLg", "BothLFp",
                 "BothLFg", "BothLLg", "TFp", "TFg", "TLg", "TotalFp", "TotalFg",
                 "TotalLg", "L1BMD", "L1T", "L1Z", "L2BMD", "L2T", "L2Z", "L3BMD",
                 "L3T", "L3Z", "L4BMD", "L4T", "L4Z", "L1L4BMD", "L1L4T", "L1L4Z",
                 "L2L4BMD", "L2L4T", "L2L4Z", "NeckFBMD", "NeckFT", "NeckFZ",
                 "WardsBMD", "WardsT", "WardsZ", "TrochBMD", "TrochT", "TrochZ",
                 "TotalFBMD", "TotalFT", "TotalFZ", "BMI", "BMI_cat", "FMI", "FFMI",
                 "Apendicularleanmas", "FMR", "FTrunkgFLegsg", "Indexdistributionfat",
                 "FtrunkpFlimbsp", "FtrunkgFtotalg", "FLegsgFtotalg", "FlimbsgFtotalg",
                 "LLegFgBMI", "LLegFpBMI", "Lipodistrophy", "Sarcopenia", "LipoSarcop",
                 "phenotype", "minTscore", "Tscore_3cat", "TotalBMD")]
```

Some extra information about the observations is stored in a second database. Both databases will be joined into a single R object in order to perform statistics.

```r
# Load second database:
dexa2 <- read_xlsx(
  "E:/Arxius Adria Regue/UOC/tercer semestre/TFM/Docs Drive/Datos/DEXA_nova.xlsx", sheet = 1)

# Select columns of interest:
dexa2 <- dexa2[, c("historial", "data_vihpos", "data DEXA")]

# Convert to factors
dexa2$historial <- factor(dexa2$historial)

# Delete repeated observations
dexa2 <- dexa2[order(dexa2$historial, dexa2$`data DEXA`, decreasing = T), ]
dexa2 <- dexa2[!duplicated(dexa2$historial), ]

# rename variables
colnames(dexa2) <- c("ID", "HIV_date", "dexa_date")

# New column: years infected at the moment of the DEXA
dexa2$Disease_age <- round((dexa2$dexa_date-dexa2$HIV_date)/365, 2)

# Join both databases
dexa <- merge.data.frame(dexa, dexa2, by = "ID", all.x = T); rm(dexa2)
```

Initial treatment of data

Before an univariate study is conducted, various data issues must be adressed. Those issues include management of NAs, outliers, correct definition of the nature of the variables (numeric, factorial...) and detection of typos. All decisions taken during the analysis will be captured in this Rmarkdown.

Defining categorial variables

There are 9 variables that R imported as numeric or character while they represent factors. We are talking about *ID, gender, gender_num, Age_cat, BMI_cat, Lipodistrophy, Sarcopenia, LipoSarcop* and *phenotype*.

`factor{base}` and `as.factor{base}` will be used to reconfigure said variables. In adition, gender_num is coded as a two level factor (1, 2) when a (0,1) level is preferred. The new coding for gender_num is defined as: **0 = Female**, **1 = Male**.

Lipodistrophy has been renamed to Lipodystrophy.

Disease_age has been converted to numeric.

```
dexa$ID <- as.factor(dexa$ID)

dexa$gender <- factor(dexa$gender, levels = c("F","M"))

dexa$gender_num[which(dexa$gender_num==2)] <- 0
dexa$gender_num <- factor(dexa$gender_num)

dexa$Age_cat <- factor(dexa$Age_cat, levels = c("0","1"), labels = c("<50",">50"))

dexa$BMI_cat <- factor(dexa$BMI_cat, levels = c(0,1,2,3),
                       labels = c("Underweight", "Normal", "Overweight", "Obesity"))

dexa$Lipodistrophy <- factor(dexa$Lipodistrophy, levels = c(0,1), labels = c("Healthy", "Diseased"))
names(dexa)[names(dexa)=="Lipodistrophy"] <- c("Lipodystrophy")

dexa$Sarcopenia <- factor(dexa$Sarcopenia, levels = c(0,1), labels = c("Healthy", "Diseased"))

dexa$LipoSarcop <- factor(dexa$LipoSarcop, levels = c(0,1), labels = c("Healthy", "Diseased"))

dexa$phenotype <- factor(dexa$phenotype, levels = 1:16, labels = c(
  "Underweight, lipo, sarco",  "Underweight, lipo",
  "Underweight, sarco",        "Normal, lipo, sarco",
  "Normal, lipo",              "Normal, sarco",
  "Obesity, lipo, sarco",      "Obesity, lipo",
  "Obesity, sarco",            "Underweight",
  "Normal",                    "Obesity",
  "Overweight",                "Overweight, lipo, sarco",
  "Overweight, lipo",          "Overweight, sarco"
))

dexa$Disease_age <- as.numeric(dexa$Disease_age)
```

Typos

Some unexpected values have been detected in several variables. Those values do not correspond to a biologically possible scenario and therefore are attributed to typos. Following are the typos documented and how they have been addressed:

- **Issue 1:** ID 420494 has a weight value of 0 kg and BMI = FMI = FFMI = 0. All values replaced with NAs.
- **Issue 2:** IDs 349374 and 10006435 have LAFp values over 100. This variable represents percentiles and therefore its values must be in range [0, 100]. I understand that this is a typo in the decimal separation, therefore I divide the unexpected values by 100.
- **Issue 3:** Sample 378 has some variables beyond what is mathematically possible: FtrunkgFtotalg, FLegsgFtotalg and FlimbsgFtotalg are percentiles, and must be in range [0,1]. This sample has higher

values, which I attribute as a wrongly placed decimal coma. Dividing the sample values by 10 places their new value in range.

- **Issue 4:** ID 420494 has an apendicularleanmas of 0. All apendicularleanmas values are recalculated, based on the following formula: $Apendicularleanmas = \frac{BothALg+BothLLg}{Height^2 ù 1000}$
- **Issue 5:** Columns FTrunkgFLegsg and FtrunkgFtotalg are identical. I recalculate the former, based on the formula: $FTrunkgFLegsg = \frac{TFg}{BothLFg}$
- **Issue 6:** sarcopenia cutoff points used to classify observations.
- **Issue 7:** FMR variable calculation.
- **Issue 8:** lipodystrophy curoff points used to classify observations.
- **Issue 9:** redefinition of liposarcopenia.

```
# Issue 1
dexa[dexa$ID=="420494",c("Weight","BMI","FMI","FFMI")] <- NA


# Issue 2
for(i in dexa$LAFp){
  if(i>100){
    dexa$LAFp[which(dexa$LAFp==i)] <- i/100
  }
}; rm(i)

# Issue 3

for(i in dexa$FtrunkgFtotalg){
  if(i>1){
    dexa$FtrunkgFtotalg[which(dexa$FtrunkgFtotalg==i)] <- i/10
  }
}; rm(i)

for(i in dexa$FLegsgFtotalg){
  if(i>1){
    dexa$FLegsgFtotalg[which(dexa$FLegsgFtotalg==i)] <- i/10
  }
}; rm(i)

for(i in dexa$FlimbsgFtotalg){
  if(i>1){
    dexa$FlimbsgFtotalg[which(dexa$FlimbsgFtotalg==i)] <- i/10
  }
}; rm(i)

# Issue 4
dexa$Apendicularleanmas <- (dexa$BothALg + dexa$BothLLg) /
  (dexa$Height^2 * 1000)

# Issue 5
dexa$FTrunkgFLegsg <- dexa$TFg/dexa$BothLFg

# Issue 6 (LMM)
```

```r
for (i in 1:dim(dexa)[1]){
  if (dexa$gender[i]=="M"){
    dexa$Sarcopenia[i] <- ifelse(dexa$Apendicularleanmas[i] < 7, "Diseased", "Healthy")
  } else {
    dexa$Sarcopenia[i] <- ifelse(dexa$Apendicularleanmas[i] < 6, "Diseased", "Healthy")
  }
}; rm(i)

# Issue 7 (FMR)
dexa$FMR <- dexa$TFp/dexa$BothLFp

# Issue 8 (lipodystrohpy)
for (i in 1:dim(dexa)[1]){
  if(dexa$gender[i]=="M"){
    dexa$Lipodystrophy[i] <- ifelse(dexa$FMR[i] >= 1.961, "Diseased", "Healthy")
  } else {
    dexa$Lipodystrophy[i] <- ifelse(dexa$FMR[i] >= 1.329, "Diseased", "Healthy")
  }
}

# Issue 9 (liposarcop)
for (i in 1:dim(dexa)[1]){
  dexa$LipoSarcop[i] <- ifelse(dexa$Sarcopenia[i]=="Healthy" &
                                 dexa$Lipodystrophy[i]=="Healthy", "Healthy", "Diseased")
}
```

Duplicate IDs

Of the databases provided, we are working with the one that represents only the latest of the DEXAs of every patient and therefore we should not have any repeated IDs. Duplicated ID values are faced and managed as follows: for each ID that shows various entries, only the latest one will be kept, based on patients age.

`duplicated{base}` detects repeated values inside a column, so in order to delete all duplicates and keep only the newest observation the database is firstly ordered by age (`order{base}`, higher to lower) and then filtered.

```r
dexa <- dexa[order(dexa$ID, -dexa$Age), ]
dexa <- dexa[!duplicated(dexa$ID), ]
```

0.34% of the observations have been excluded.

Calculation of minTscore and Tscore_3cat

MinTscore/Tscore_3cat calculated according to the WHO criteria. Details in paper.

```r
# minTscore
for(i in 1:dim(dexa)[1]){
  dexa$minTscore[i] <-min(c(
    dexa$L1L4T[i],   dexa$NeckFT[i],
    dexa$TrochT[i],  dexa$TotalFT[i] #,
    #dexa$L1T[i],    #dexa$L2T[i],
    #dexa$L3T[i],    #dexa$L4T[i],
```

```
    #dexa$L2L4T[i], #dexa$WardsT[i]
    ), na.rm = T)
}; rm(i)

# Tscore_3cat
for(i in 1:dim(dexa)[1]){
  if(dexa$minTscore[i] <= (-2.5)){
    dexa$Tscore_3cat[i] <- "Osteoporosis"
  }
  else(
    ifelse(dexa$minTscore[i]<= -1,
           dexa$Tscore_3cat[i] <- "Osteopenia",
           dexa$Tscore_3cat[i] <- "Healthy"))
}; rm(i)

dexa$Tscore_3cat <- factor(dexa$Tscore_3cat) # convert to factor
```

Management of NAs

In order to avoid (when possible) data deletion, two groups of samples have been defined:

- **Group 1:** The first ones contain missing values on variables of crucial importance for the study. Those observations are eliminated in all cases.
- **Group 2:** The other group are observations with missing values in variables that may not be part of the final model because they might not be as robust as other variables in the database (i.e. single vertebrae observations vs L1L4 or L2L4 values). A second database is defined, keeping those observations.

```
# Create a List with "complete.cases"
x <- (lapply(dexa, complete.cases))

# Get a second list with only the variables that are NA:
NAvals <- list()

for (i in names(x)){
  NAvals [[i]] <- (which(x[[i]]==F))
}; rm(i); rm(x)
```

```
# Create a copy of the database to study deleted samples:
dexa_del <- dexa
# Variables that are not as robust as others:
weak_vars <- c("L1BMD","L1T","L1Z","L2BMD","L2T","L2Z","L3BMD","L3T","L3Z","L4BMD","L4T","L4Z")

# Group 2
drop <- dexa[, !names(dexa) %in% c(weak_vars, "minTscore", "Tscore_3cat")]
dexa_weak <- dexa[complete.cases(drop),]

# Group 1
dexa <- dexa[complete.cases(dexa[, !names(dexa) %in% c("minTscore", "Tscore_3cat")]),]
```

Therfore, we end up with 2 data.frames: “dexa” which contains no NAs, and “dexa_weak” that contain NAs only in “weak” variables. In the former database we deleted 3.32% of the observations; in the later, 3.05%. In relation with the original sample size, the reduction is of a 3.65% and a 3.38% respectively.

The less restrictive data-frame is 4 observations larger than the other one. Observations with NAs are studied to detect any patterns, and then removed (no patterns detected).

```
# Deleted samples IDs:
del <- dexa_del[which(!dexa_del$ID %in% dexa$ID),]
del$ID

# Numeric table
means <- round(del %>% summarise_if(is.numeric, mean, na.rm = T), 2)
stdevs <- round(del %>% summarise_if(is.numeric, sd, na.rm = T), 2)
minvals <- round(del %>% summarise_if(is.numeric, min, na.rm = T), 2)
maxvals <- round(del %>% summarise_if(is.numeric, max, na.rm = T), 2)
q25 <- round(sapply(del[,sapply(del, is.numeric)],
                    quantile, probs = 0.25, na.rm = T), 2)
q75 <- round(sapply(del[,sapply(del, is.numeric)],
                    quantile, probs = 0.75, na.rm = T), 2)


mat <- matrix(c(minvals, q25, means, q75, maxvals, stdevs), ncol = 6)
rownames(mat) <- names(del[,sapply(del, is.numeric)])

deleted_summary <- kable(mat, col.names = c("Min", "25%", "Mean", "75%", "Max", "SD"))
```

```
# Cleanup
rm(NAvals, drop, weak_vars, dexa_weak, del, dexa_del,
   means, stdevs, minvals, maxvals, q25, q75, mat)
```

```
# Renumber dexa rows
rownames(dexa) <- c(1:dim(dexa)[1])
```

Outlier detection

```
# Whole population
subset <- dexa[,which(sapply(dexa, is.numeric))]
m_dist <- mahalanobis(subset, colMeans(subset), cov(subset))
names(m_dist) <- dexa[, "ID"]

png(file = "Graphs/1. Outliers/Whole_set.png")
plot(m_dist); text(m_dist, labels = names(m_dist)); abline(
  h = m_dist[order(m_dist, decreasing = T)][5], col = "blue")
dev.off()

# Females
subsetF <- dexa[which(dexa$gender=="F"),which(sapply(dexa, is.numeric))]
m_distF <- mahalanobis(subsetF, colMeans(subsetF), cov(subsetF))
names(m_distF) <- dexa[which(dexa$gender=="F"), "ID"]

png(file = "Graphs/1. Outliers/Female.png")
plot(m_distF); text(m_distF, labels = names(m_distF)); abline(
  h = m_distF[order(m_distF, decreasing = T)][5], col = "blue")
dev.off()
```

```r
# Males
subsetM <- dexa[which(dexa$gender=="M"),which(sapply(dexa, is.numeric))]
m_distM <- mahalanobis(subsetM, colMeans(subsetM), cov(subsetM))
names(m_distM) <- dexa[which(dexa$gender=="M"), "ID"]

png(file = "Graphs/1. Outliers/Male.png")
plot(m_distM); text(m_distM, labels = names(m_distM)); abline(
  h = m_distM[order(m_distM, decreasing = T)][5], col = "blue")
dev.off()
```

```r
# Cleanup
rm(subset, subsetM, subsetF, m_dist, m_distM, m_distF)
```

IDs of the 5 uppermost samples:

- **Whole sample:** "15958419", "63394", "608293", "180433" and "12120517".
- **Females:** "63394", "15958419", "528280", "397968" and "353494".
- **Males:** "608293", "445579", "12120517", "180433" and "162626".

Looking at the phenotypes and the number of years living with the disease, those observations don't show any clear patterns. Sample ID "63394" has a FMR value about 2 points over the next sample. I don't have enough background to decide if we are talking about a human error or it is a biologically possible scenario. No variables are dropped, because we lack the background information to further corroborate the nature of said observations.

Exploratory data analysis

Factorial variable barplots

Using `ggplot2`, graphics have been generated for every factorial variable. The graphics represent the proportions inside the whole group of observations and inside every gender.

```r
# Calculate the proportions of every group:
facts <- dexa %>% select(names(dexa[sapply(dexa, is.factor)])) %>%
  select(-c("ID", "gender", "gender_num")) %>%
  sapply(table) %>%sapply(prop.table)

fem.facts <- dexa[which(dexa$gender=="F"),] %>% select(names(dexa[sapply(dexa, is.factor)])) %>%
  select(-c("ID", "gender", "gender_num")) %>% sapply(table) %>% sapply(prop.table)

male.facts <- dexa[which(dexa$gender=="M"),] %>% select(names(dexa[sapply(dexa, is.factor)])) %>%
  select(-c("ID", "gender", "gender_num")) %>% sapply(table) %>% sapply(prop.table)


# Generate a function that arranges data to be ploted:
temp.df <- function(x) {
  df <- data.frame(
    # Column 1: factor levels
```

```
    rep(names(facts[[x]]),3),
    # Column 2: Proportion values
    as.numeric(c(male.facts[[x]], fem.facts[[x]], facts[[x]])),
    # Column 3: gender
    rep(c("M", "F", "All"), each = length(facts[[x]])))

  colnames(df) <- c(paste(x), "Proportions", "Gender")
  return(df)
}

# Generate the graphics:
for (name in names(facts)){
  ggplot(temp.df(name), aes_string(y="Proportions", x = name, fill = "Gender")) +
    geom_bar(position="dodge", stat="identity", alpha = 0.7, show.legend = F) +
    scale_fill_manual(values = c("#999999","#f8766d", "#00bfc4")) +
    geom_text(aes(label=paste(round(Proportions,2)*100, "%", sep="")),
              position=position_dodge(width=0.9), size = 10) + coord_flip() +
    theme(text = element_text(size = 20), axis.title.x = element_blank(),
          axis.ticks.x = element_blank(), axis.text.x = element_blank())

  ggsave(filename = paste("Graphs/2. Barplots/", name, ".png", sep = ""),
         plot = last_plot(), device = "png")
}
```

```
# Cleanup
rm(facts, fem.facts, male.facts, temp.df, name)
```

Summary overview of numeric variables

```
# Table construction
# Generic values
means <- round(dexa %>% summarise_if(is.numeric, mean, na.rm = T), 2)
stdevs <- round(dexa %>% summarise_if(is.numeric, sd, na.rm = T), 2)
minvals <- round(dexa %>% summarise_if(is.numeric, min, na.rm = T), 2)
maxvals <- round(dexa %>% summarise_if(is.numeric, max, na.rm = T), 2)
q25 <- round(sapply(dexa[,sapply(dexa, is.numeric)], quantile, probs = 0.25), 2)
q75 <- round(sapply(dexa[,sapply(dexa, is.numeric)], quantile, probs = 0.75), 2)


mat <- matrix(c(minvals, q25, means, q75, maxvals, stdevs), ncol = 6)
rownames(mat) <- names(dexa[,sapply(dexa, is.numeric)])

generic_summary <- kable(mat, col.names = c("Min", "25%", "Mean", "75%", "Max", "SD"))


# By gender

group.means <- dexa %>% group_by(gender) %>% summarise_if(is.numeric, mean, na.rm = T)
group.stdevs <- dexa %>% group_by(gender) %>% summarise_if(is.numeric, sd, na.rm = T)
group.minvals <- dexa %>% group_by(gender) %>% summarise_if(is.numeric, min, na.rm = T)
group.q25 <- dexa %>% group_by(gender) %>% summarise_if(is.numeric,
```

```
                                                    quantile, probs=0.25, na.rm = T)
group.q75 <- dexa %>% group_by(gender) %>% summarise_if(is.numeric,
                                                    quantile, probs=0.75, na.rm = T)
group.maxvals <- dexa %>% group_by(gender) %>% summarise_if(is.numeric, max, na.rm = T)


mat <- matrix(c(round(group.q25[1,-1], 2), round(group.means[1,-1], 2),
                round(group.q75[1,-1], 2), round(group.stdevs[1,-1], 2),
                round(group.q25[2,-1], 2), round(group.means[2,-1], 2),
                round(group.q75[2,-1], 2), round(group.stdevs[2,-1], 2)),
              ncol = 8)
rownames(mat) <- names(dexa[,sapply(dexa, is.numeric)])

bysex_summary <- kable(mat, col.names = c("25% F", "Mean F", "75% F", "SD F",
                                          "25% M", "Mean M", "75% M", "SD M"))
```

```
# Cleanup
rm(means, stdevs, minvals, maxvals, q25, q75, group.means,
   group.stdevs, group.minvals, group.maxvals, group.q25, group.q75, mat)
```

Density plots of numeric variables

```
# Calculate the means of every variable, by gender
group.means <- dexa %>% group_by(gender) %>% summarise_if(
  is.numeric, mean, na.rm = T)

dens.graphs <- list()

for (varname in names(dexa[sapply(dexa, is.numeric)])){
  densplot <- ggplot(dexa, aes_string(x=varname, fill = "gender")) +
    geom_density(alpha=0.4) +
    geom_vline(data = group.means, aes_string(xintercept=varname, color= "gender"),
               linetype = "dashed", size = 1) +
    labs(title=paste(varname, "density plot"), y = "Density", x = element_blank())

  # Save each graphic into the harddrive
  ggsave(filename = paste("Graphs/3. Densplots/", varname, ".png", sep = ""),
         plot = last_plot())

  # Save each graphic into an R object (list)
  dens.graphs[[varname]] <- densplot
}; rm(densplot, varname, group.means) # Cleanup
```

Gender effect

T-test

The null hypothesis states that "there is no difference on the mean of the variable because of the gender of the population", and its viability is checked using a T-test (`t.test{stats}`). Significance level has been stablished at 0.05, and variances have been treated as unequal (Welch approximation).

```r
gender.tscores <- lapply(dexa[,names(dexa[,sapply(dexa,is.numeric)])],
                         function(x) t.test(x ~ dexa$gender, conf.level = 0.95)$p.value)

names(which(gender.tscores >= 0.05))

# Table with p-values (values <0.0001 excluded)
gender.tscores <- gender.tscores[gender.tscores %>% unlist %>% order(decreasing = T)]
gender.tscores <- gender.tscores[which(gender.tscores>0.0001)]
varnames <- names(gender.tscores)
values <- round(as.numeric(gender.tscores), 4)
ttest_pvals <- kable(matrix(c(varnames, values), ncol=2), col.names = c("Variable", "P-Value"))
```

```r
# Cleanup
rm(gender.tscores, values, varnames)
```

Linear correlation between variables

```r
corr_simple <- function(data, sig = 0.5, met = "circle"){
  # Convert data to numeric in order to run correlations
  df_cor <- data %>% mutate_if(is.factor, as.numeric)
  # Calculate the correlation matrix
  corr <- cor(df_cor)
  # Prepare to drop duplicates
  corr[lower.tri(corr,diag=TRUE)] <- NA
  # Turn into a 3-column table
  corr <- as.data.frame(as.table(corr))
  # Remove the NA values from above
  corr <- na.omit(corr)
  # Select significant values
  corr <- subset(corr, abs(Freq) > sig)
  # Turn corr back into matrix in order to plot with corrplot
  mtx_corr <- reshape2::acast(corr, Var1~Var2, value.var="Freq")

  # Plot correlations visually
  corrplot(mtx_corr, is.corr=FALSE, type = "upper", tl.col="black",
           na.label=" ", tl.cex=0.8, tl.srt = 45, method = met)
}

# Create and save the graphic
png(file = "Graphs/4. Corplots/Whole_set.png"); corr_simple(dexa[, -c(83, 84)])
dev.off()
```

```r
BMD <- grep("BMD", names(dexa), value=TRUE)
mass <- grep("g", names(dexa), value=TRUE)[7:22]
fat <- grep("Fp|Fg", names(dexa), value = TRUE)[1:16]


png(file = "Graphs/4. Corplots/BMDs.png")
corr_simple(dexa[, BMD], met = "circle", sig = 0.5)
dev.off()
```

```r
png(file = "Graphs/4. Corplots/lean_fat_mass.png")
corr_simple(dexa[, mass], met = "circle", sig = 0.5)
dev.off()

png(file = "Graphs/4. Corplots/fat.png")
corr_simple(dexa[, fat], met = "circle", sig = 0.5)
dev.off()

png(file = "Graphs/4. Corplots/calc_vars.png")
corr_simple(dexa[, 64:74], met = "circle", sig = 0.5)
dev.off()
```

```r
# Cleanup
rm(corr_simple, BMD, mass, fat)
```

Normality

```r
pvals <- c()
for (i in which(sapply(dexa, is.numeric))){
  pvals <- append(pvals, shapiro.test(dexa[,i])$p.value)
}; rm(i)

pvals_f <- c()
for (i in which(sapply(dexa, is.numeric))){
  pvals_f <- append(pvals_f, shapiro.test(dexa[which(dexa$gender=="F"),i])$p.value)
}; rm(i)

pvals_m <- c()
for (i in which(sapply(dexa, is.numeric))){
  pvals_m <- append(pvals_m, shapiro.test(dexa[which(dexa$gender=="M"),i])$p.value)
}; rm(i)

names(which(sapply(dexa, is.numeric)))[which(pvals>0.1)]
names(which(sapply(dexa, is.numeric)))[which(pvals_f>0.1)]
names(which(sapply(dexa, is.numeric)))[which(pvals_m>0.1)]
```

```r
# Cleanup
rm(pvals, pvals_f, pvals_m)
```

```r
for (i in which(sapply(dexa, is.numeric))){
  png(filename = paste("Graphs/5. QQplots/", names(dexa)[i], ".png", sep=""))
  qqPlot(dexa[,i], col = "#999999", col.lines = "black", id = F,
         ylab = "Sample quantiles", xlab = "Theoretical quantiles", main = names(dexa)[i])
  dev.off()
}; rm(i)

for (i in which(sapply(dexa, is.numeric))){
  png(filename = paste("Graphs/5. QQplots/", names(dexa)[i], "_f.png", sep=""))
  qqPlot(dexa[which(dexa$gender=="F"),i], col = "#f8766d", col.lines = "black", id = F,
         ylab = "Sample quantiles", xlab = "Theoretical quantiles", main = names(dexa)[i])
```

```
  dev.off()
}; rm(i)

for (i in which(sapply(dexa, is.numeric))){
  png(filename = paste("Graphs/5. QQplots/", names(dexa)[i], "_m.png", sep=""))
  qqPlot(dexa[which(dexa$gender=="M"),i], col = "#00bfc4", col.lines = "black", id = F,
         ylab = "Sample quantiles", xlab = "Theoretical quantiles", main = names(dexa)[i])
  dev.off()
}; rm(i)
```

```
# Save untransformed database
dexa_untrans <- dexa
# Logarithmic transformations

dexa$BMI <- log(dexa$BMI)
dexa$BothAFg <- log(dexa$BothAFg)
dexa$BothALg <- log(dexa$BothALg)
dexa$BothLLg <- log(dexa$BothLLg)
dexa$BothLFg <- log(dexa$BothLFg)
dexa$FMI <- log(dexa$FMI)
dexa$FMR <- log(dexa$FMR)
dexa$FTrunkgFLegsg <- log(dexa$FTrunkgFLegsg)
dexa$Indexdistributionfat <- log(dexa$Indexdistributionfat)
dexa$LAFg <- log(dexa$LAFg)
dexa$RAFg <- log(dexa$RAFg)
dexa$LALg <- log(dexa$LALg)
dexa$RALg <- log(dexa$RALg)
dexa$LLFg <- log(dexa$LLFg)
dexa$RLFg <- log(dexa$RLFg)
dexa$LLLg <- log(dexa$LLLg)
dexa$RLLg <- log(dexa$RLLg)
dexa$TotalLg <- log(dexa$TotalLg)

# Square root transformations

dexa$TFg <- sqrt(dexa$TFg)
dexa$TLg <- sqrt(dexa$TLg)
dexa$TotalFg <- sqrt(dexa$TotalFg)

# Inverse transformations

dexa$FFMI <- 1/(dexa$FFMI)
dexa$FtrunkpFlimbsp <- 1/(dexa$FtrunkpFlimbsp)
```

PCA analysis

```
# Excluding calculated and response variables
subset <- dexa[, -c(62:81)]

# PCA over the numeric ones:
PCA <- prcomp(subset[sapply(subset, is.numeric)], scale = T, center = T)
```

```r
# Look at summary of results

summary(PCA)$importance[,1:10]

fviz_eig(PCA, barfill = "#999999", barcolor = "#999999")
```

```r
ggsave("Graphs/6. PCA/variance_explained.png")
```

```r
# 1st and 2nd components plot:
fviz_pca_var(PCA, col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)
```

```r
ggsave("Graphs/6. PCA/comp12.png")

# 1st component:
PCA.rotations <- PCA$rotation[which(abs(PCA$rotation[,1])>0.1),1]

# Those variables are the ones that appear on the cluster of variables at the right:
ggbiplot2(PCA) + ylim(-0.5,0.5) + xlim(0,1.75)
```

```r
ggsave("Graphs/6. PCA/comp1_vars.png")

# Low Muscle Mass
ggbiplot2(PCA, groups = dexa$Sarcopenia, ellipse = T, ell.size = 1) + ggtitle("LMM")
```

```r
ggsave("Graphs/6. PCA/comp12Sarco.png")

# Tscore
ggbiplot2(PCA, groups = dexa$Tscore_3cat, ellipse = T, ell.size = 1) + ggtitle("Bone health")
```

```r
ggsave("Graphs/6. PCA/comp12Bones.png")

# 2nd component:
PCA.rotations2 <- PCA$rotation[which(abs(PCA$rotation[,2])>0.1),2]

ggbiplot2(PCA) + xlim(0,1) + ylim(0,1)
```

```r
ggsave("Graphs/6. PCA/comp2_vars1.png")
ggbiplot2(PCA) + xlim(-1,1) + ylim(-2,0)
```

```r
ggsave("Graphs/6. PCA/comp2_vars2.png")

# Those variables are partially related to gender:
ggbiplot2(PCA, groups = dexa$gender, ellipse = T, ell.size = 1) + ggtitle("Gender")
```

```r
ggsave("Graphs/6. PCA/comp2_gender.png")
```

```r
dexa$FMI_cat <- NA
for (i in 1:length(dexa$FMI)){
  dexa$FMI_cat[i] <- ifelse(dexa$FMI[i] >= mean(dexa$FMI), 1, 0)
}; dexa$FMI_cat <- factor(dexa$FMI_cat, levels = c(0,1), labels = c("Above mean", "Under mean"))


ggbiplot2(PCA, groups = dexa$FMI_cat, ellipse = T, ell.size = 1) + ggtitle("FMI")
```

```r
ggsave("Graphs/6. PCA/comp2_FMI.png")

dexa <- dexa[,-86]
```

```r
# 3rd and 4th components:
fviz_pca_var(PCA, axes = c(3,4), col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)
```

```r
ggsave("Graphs/6. PCA/comp34.png")

PCA.rotations3 <- PCA$rotation[which(abs(PCA$rotation[,3])>0.1),3]

## 3rd component is related to BMI_cat
ggbiplot2(PCA, choices = c(3,4), groups = dexa$BMI_cat, ellipse = T, ell.size = 1) + ggtitle("BMI")
```

```r
ggsave("Graphs/6. PCA/comp3_BMI.png")
```

Females

```r
# Excluding calculated and response variables
subset <- dexa[dexa$gender=="F", -c(62:81)]

# PCA over the numeric ones:
PCAF <- prcomp(subset[sapply(subset, is.numeric)], scale = T, center = T)

# Look at summary of results
summary(PCAF)$importance[,1:10]

fviz_eig(PCAF, barfill = "#999999", barcolor = "#999999")
```

```r
ggsave("Graphs/6. PCA/F/variance_explained.png")
```

```r
# 1st and 2nd components plot:
fviz_pca_var(PCAF, col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)
```

```r
ggsave("Graphs/6. PCA/F/comp12.png")

# 1st component:
```

```r
PCAF.rotations <- PCAF$rotation[which(abs(PCAF$rotation[,1])>0.11),1]

# Those variables are the ones that appear on the cluster of variables at the right:
ggbiplot2(PCAF) + ylim(-0.5,0.5) + xlim(0,1.75)
```

```r
ggsave("Graphs/6. PCA/F/comp1_vars.png")

# Low Muscle Mass
ggbiplot2(PCAF, groups = dexa$Sarcopenia[which(dexa$gender=="F")],
          ellipse = T, ell.size = 1) + ggtitle("LMM")
```

```r
ggsave("Graphs/6. PCA/F/comp12Sarco.png")
# Tscore
ggbiplot2(PCAF, groups = dexa$Tscore_3cat[which(dexa$gender=="F")],
          ellipse = T, ell.size = 1) + ggtitle("Bone health")
```

```r
ggsave("Graphs/6. PCA/F/comp12Bones.png")
```

```r
# 2nd component:
PCAF.rotations2 <- PCAF$rotation[which(abs(PCAF$rotation[,2])>0.1),2]

ggbiplot2(PCAF) + xlim(-0.5,1) + ylim(0,1.6)
```

```r
ggsave("Graphs/6. PCA/F/comp2_vars1.png")

# Those variables are partially related to BMI:
ggbiplot2(PCAF, groups = dexa$BMI_cat[which(dexa$gender=="F")],
          ellipse = T, ell.size = 1) + ggtitle("BMI")
```

```r
ggsave("Graphs/6. PCA/F/comp2_BMI.png")

dexa$FMI_cat <- NA
for (i in 1:length(dexa$FMI)){
  dexa$FMI_cat[i] <- ifelse(dexa$FMI[i] >= mean(dexa$FMI), 1, 0)
}; dexa$FMI_cat <- factor(dexa$FMI_cat, levels = c(0,1), labels = c("Above mean", "Under mean"))

ggbiplot2(PCAF, groups = dexa$FMI_cat[which(dexa$gender=="F")],
          ellipse = T, ell.size = 1) + ggtitle("FMI")
```

```r
ggsave("Graphs/6. PCA/F/comp2_FMI.png")

dexa <- dexa[,-86]
```

```r
# 3rd and 4th components:
fviz_pca_var(PCAF, axes = c(3,4), col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)
```

```r
ggsave("Graphs/6. PCA/F/comp34.png")

PCAF.rotations3 <- PCAF$rotation[which(abs(PCAF$rotation[,3])>0.1),3]
```

Males

```
# Excluding calculated and response variables
subset <- dexa[dexa$gender=="M", -c(62:81)]

# PCA over the numeric ones:
PCAM <- prcomp(subset[sapply(subset, is.numeric)], scale = T, center = T)

# Look at summary of results
summary(PCAM)$importance[,1:11]

fviz_eig(PCAM, barfill = "#999999", barcolor = "#999999")
```

```
ggsave("Graphs/6. PCA/M/variance_explained.png")
```

```
# 1st and 2nd components plot:
fviz_pca_var(PCAM, col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)
```

```
ggsave("Graphs/6. PCA/M/comp12.png")

# 1st component:
PCAM.rotations <- PCAM$rotation[which(abs(PCAM$rotation[,1])>0.10),1]

# Those variables are the ones that appear on the cluster of variables at the right:
ggbiplot2(PCAM) + ylim(-0.5,0.5) + xlim(0,1.75)
```

```
ggsave("Graphs/6. PCA/M/comp1_vars.png")

# Low Muscle Mass
ggbiplot2(PCAM, groups = dexa$Sarcopenia[which(dexa$gender=="M")],
          ellipse = T, ell.size = 1) + ggtitle("LMM")
```

```
ggsave("Graphs/6. PCA/M/comp12Sarco.png")
# Tscore
ggbiplot2(PCAM, groups = dexa$Tscore_3cat[which(dexa$gender=="M")],
          ellipse = T, ell.size = 1) + ggtitle("Bone health")
```

```
ggsave("Graphs/6. PCA/M/comp12Bones.png")
```

```
# 2nd component:
PCAM.rotations2 <- PCAM$rotation[which(abs(PCAM$rotation[,2])>0.1),2]

ggbiplot2(PCAM) + xlim(-0.5,1) + ylim(-2,0)
```

```
ggsave("Graphs/6. PCA/M/comp2_vars1.png")

# Those variables are partially related to BMI:
ggbiplot2(PCAM, groups = dexa$BMI_cat[which(dexa$gender=="M")],
          ellipse = T, ell.size = 1) + ggtitle("BMI")
```

```r
ggsave("Graphs/6. PCA/M/comp2_BMI.png")


dexa$FMI_cat <- NA
for (i in 1:length(dexa$FMI)){
  dexa$FMI_cat[i] <- ifelse(dexa$FMI[i] >= mean(dexa$FMI), 1, 0)
}; dexa$FMI_cat <- factor(dexa$FMI_cat, levels = c(0,1), labels = c("Above mean", "Under mean"))

ggbiplot2(PCAM, groups = dexa$FMI_cat[which(dexa$gender=="M")],
          ellipse = T, ell.size = 1) + ggtitle("FMI")
```

```r
ggsave("Graphs/6. PCA/M/comp2_FMI.png")

dexa <- dexa[,-86]
```

```r
# 3rd and 4th components:
fviz_pca_var(PCAM, axes = c(3,4), col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)
```

```r
ggsave("Graphs/6. PCA/M/comp34.png")

PCAM.rotations3 <- PCAM$rotation[which(abs(PCAM$rotation[,3])>0.1),3]
```

```r
# Cleanup
rm(subset, i, ggbiplot2)

PCAlist <- list(PCA.rotations, PCA.rotations2, PCA.rotations3,
                PCAF.rotations, PCAF.rotations2, PCAF.rotations3,
                PCAM.rotations, PCAM.rotations2, PCAM.rotations3)

names(PCAlist) <- c("PCA.rotations", "PCA.rotations2", "PCA.rotations3",
                    "PCAF.rotations", "PCAF.rotations2", "PCAF.rotations3",
                    "PCAM.rotations", "PCAM.rotations2", "PCAM.rotations3")

rm(PCA.rotations, PCA.rotations2, PCA.rotations3,
   PCAF.rotations, PCAF.rotations2, PCAF.rotations3,
   PCAM.rotations, PCAM.rotations2, PCAM.rotations3)
```

Graphical models

GGM undirected

```r
# Drop factors and non-numeric variables:
GGMdexa <- dexa_untrans[,which(lapply(dexa_untrans, class)=="numeric")]
```

```r
# Generate groups to be colored in the graph
groups_items <- list(
  `patient variables [1-3]` = c(1:3),
```

```
  `fat/lean variables [4-27]` = c(4:27),
  `vertebrae bone variables [28-45]` = c(28:45),
  `femoral bone variables [46-57]` = c(46:57),
  `summary variables [58-70]` = c(58:70),
  `minTscore [71]` = c(71),
  `TotalBMD [72]` = c(72),
  `Disease_age [73]` = c(73)
)

# Set up color scales
col_item <- brewer.pal(length(groups_items),name = "Set3")

# Males
glasso_item <-
  qgraph(
    cor(GGMdexa[which(dexa$gender=="M"),]),
    layout = "spring",
    graph = "glasso",
    labels = TRUE,
    threshold = TRUE,
    sampleSize = dim(GGMdexa[which(dexa$gender=="M"),])[1],
    tuning = 0.5,
    minimum = 0.10,
    groups = groups_items,
    legend.cex = 0.65,
    layoutOffset = c(-0.2,0),
    vsize = 3.0,
    color = col_item,
    filename = "Graphs/8. GM/GGM_m",
    filetype = "png"
); rm(glasso_item)

# Females
glasso_item <-
  qgraph(
    cor(GGMdexa[which(dexa$gender=="F"),]),
    layout = "spring",
    graph = "glasso",
    labels = TRUE,
    threshold = TRUE,
    sampleSize = dim(GGMdexa[which(dexa$gender=="F"),])[1],
    tuning = 0.5,
    minimum = 0.10,
    groups = groups_items,
    legend.cex = 0.65,
    layoutOffset = c(-0.2,0),
    vsize = 3.0,
    color = col_item,
    filename = "Graphs/8. GM/GGM_f",
    filetype = "png"
); rm(glasso_item)
```

```r
# Cleanup
rm(groups_items)
```

GGM directed

```r
# Covariance matrix
covmatm <- cov.wt(GGMdexa[which(dexa$gender=="M"),], method = "ML")$cov
covmatf <- cov.wt(GGMdexa[which(dexa$gender=="F"),], method = "ML")$cov
# Correlation matrix
cormatm <- cov2cor(covmatm)
cormatf <- cov2cor(covmatf)

# Generate body of GGM
suffStatm<- list(C = cormatm, n = nrow(GGMdexa[which(dexa$gender=="M"),]))
suffStatf <- list(C = cormatf, n = nrow(GGMdexa[which(dexa$gender=="F"),]))
skeleton.bodym <- skeleton(suffStat = suffStatm, indepTest = gaussCItest,
                           labels = paste(1:length(GGMdexa)), alpha = 0.05)
skeleton.bodyf <- skeleton(suffStat = suffStatf, indepTest = gaussCItest,
                           labels = paste(1:length(GGMdexa)), alpha = 0.05)

# Assign directions
pdag.bodym <- udag2pdag(skeleton.bodym, verbose = 0)
pdag.bodyf <- udag2pdag(skeleton.bodyf, verbose = 0)
nodes(pdag.bodym@graph) = nodes(skeleton.bodym@graph)
nodes(pdag.bodyf@graph) = nodes(skeleton.bodyf@graph)

# Set node visual parameters
nod <- list()
nod$fontsize <- rep(10, dim(GGMdexa)[2])
nod$fillcolor <- c(
  rep("#8DD3C7", length(1:3)), #patient
  rep("#FFFFB3", length(4:27)), # fat/lean
  rep("#BEBADA", length(28:45)), # vertebrae
  rep("#FB8072", length(46:57)), # femoral
  rep("#80B1D3", length(58:70)), # summary
  rep("#FDB462", length(71)), # minTscore
  rep("#B3DE69", length(72)), # TotalBMD
  rep("#FCCDE5", length(73)) # Disease_age
  )

nod <- lapply(nod, function(x) { names(x) <- nodes(skeleton.bodym@graph); x})

# Plot results
png(file = "Graphs/8. GM/GGM_directed_m.png", width = 1200, height = 1000)
plot(pdag.bodym@graph, nodeAttrs = nod)
legend("bottomleft",legend=c("patient variables [1-3]",
                             "fat/lean variables [4-27]",
                             "vertebrae bone variables [28-45]",
                             "femoral bone variables [46-57]",
                             "summary variables [58-70]",
                             "minTscore [71]",
```

```
                          "TotalBMD [72]",
                          "Disease_age [73]"),
       pch=16, col = c("#8DD3C7", "#FFFFB3", "#BEBADA", "#FB8072",
                       "#80B1D3", "#FDB462", "#B3DE69", "#FCCDE5"))
dev.off()

png(file = "Graphs/8. GM/GGM_directed_f.png", width = 1200, height = 1000)
plot(pdag.bodyf@graph, nodeAttrs = nod)
legend("bottomright",legend=c("patient variables [1-3]",
                          "fat/lean variables [4-27]",
                          "vertebrae bone variables [28-45]",
                          "femoral bone variables [46-57]",
                          "summary variables [58-70]",
                          "minTscore [71]",
                          "TotalBMD [72]",
                          "Disease_age [73]"),
       pch=16, col = c("#8DD3C7", "#FFFFB3", "#BEBADA", "#FB8072",
                       "#80B1D3", "#FDB462", "#B3DE69", "#FCCDE5"))
dev.off()

# fitness
pdag.bodym@graph@nodes <- names(GGMdexa)
pdag.bodyf@graph@nodes <- names(GGMdexa)
fitnessm <- fitDag(amat = as(pdag.bodym@graph, "matrix"),
                   S = covmatm, n = nrow(GGMdexa[which(dexa$gender=="M"),]))
fitnessf <- fitDag(amat = as(pdag.bodyf@graph, "matrix"),
                   S = covmatf, n = nrow(GGMdexa[which(dexa$gender=="F"),]))
pchisq(q = fitnessm$dev, df = fitnessm$df, lower.tail = F) # reject Ho (<0.001)
pchisq(q = fitnessf$dev, df = fitnessf$df, lower.tail = F) # reject Ho
```

```
# Cleanup
rm(covmatm, covmatf, cormatm, cormatf, skeleton.bodym,
   skeleton.bodyf, suffStatm, suffStatf, pdag.bodym,
   pdag.bodyf, nod, fitnessm, fitnessf, GGMdexa)
```

MGM

```
# Select data
dexa2 <- dexa_untrans[which(dexa$gender=="M"),-c(1:3)] # remove ID and genders
dexa2 <- dexa2[,-c(80:81)] # remove date variables
dexa2 <- dexa2[, -76] # remove "phenotype", lacks observations in groups

# application
# class of varaibles:
cls <- unlist(lapply(dexa2, class)) # c = categoric, g = gaussian, p = poisson
for (i in 1:length(cls)){
  if (cls[i]=="factor") {
    cls[i] <- "c"} else if (cls[i]=="numeric") {
        cls[i] <- "g"} else {
          cls[i] <- "p"}
}; rm(i)
```

```r
names(cls) <- c()

# Levels of factor variables (non factor set to 1):
levs <- c()
for (i in 1:length(dexa2)){
  levs[i] <- length(levels(dexa2[,i]))
}; rm(i)
levs[which(levs=="0")] <- 1

# convert factors to numbers (i.e. 0L, 1L, 2L...)
dexa2$Age_cat <- as.numeric(dexa2$Age_cat)
dexa2$BMI_cat <- as.numeric(dexa2$BMI_cat)
dexa2$Lipodystrophy <- as.numeric(dexa2$Lipodystrophy)
dexa2$Sarcopenia <- as.numeric(dexa2$Sarcopenia)
dexa2$LipoSarcop <- as.numeric(dexa2$LipoSarcop)
dexa2$Tscore_3cat <- as.numeric(dexa2$Tscore_3cat)

# Fit the model
fit_ADS2 <- mgm(data = dexa2,
                type = cls,
                level = levs,
                k = 2,
                lambdaSel = 'EBIC',
                lambdaGam = 0.5,
                pbar = TRUE)


fit_ADS3 <- mgm(data = dexa2,
                type = cls,
                level = levs,
                k = 2,
                lambdaSel = 'CV',
                lambdaFolds = 10,
                pbar = TRUE)

# set color parameters:

groups_items <- list(
  `patient variables [1-4]` = c(1:4),
  `fat/lean variables [5-28]` = c(5:28),
  `vertebrae bone variables [29-46]` = c(29:46),
  `femoral bone variables [47-58]` = c(47:58),
  `summary variables [59-72]` = c(59:72),
  `lipodystrophy [73]` = c(73),
  `sarcopenia [74]` = c(74),
  `lipo-sarco [75]` = c(75),
  `minTscore [76]` = c(76),
  `Tscore_3cat [77]` = c(77),
  `TotalBMD [78]` = c(78),
  `Disease_age [79]` = c(79)
)
col_item <- brewer.pal(length(groups_items),name = "Set3")
```

```r
# EBIC 0.5
qgraph(fit_ADS2$pairwise$wadj,
       layout = 'spring', repulsion = 1.25,
       edge.color = fit_ADS2$pairwise$edgecolor,
       nodeNames = names(dexa2),
       color = col_item,
       groups = groups_items,
       legend.mode="groups", legend.cex=.45,
       vsize = 3.4, esize = 15, vTrans = 200,
       title = "MGM of all variables, gender = male",
       filename = "Graphs/9. MGM/EBICmale",
       filetype = "png")


# 10fold CV
qgraph(fit_ADS3$pairwise$wadj,
       layout = 'spring', repulsion = 1.25,
       edge.color = fit_ADS3$pairwise$edgecolor,
       nodeNames = names(dexa2),
       color = col_item,
       groups = groups_items,
       legend.mode="groups", legend.cex=.45,
       vsize = 3.4, esize = 15, vTrans = 200,
       title = "MGM of all variables, gender = male",
       filename = "Graphs/9. MGM/CVmale",
       filetype = "png")
```

```r
# Select data
dexa2 <- dexa_untrans[which(dexa$gender=="F"),-c(1:3)] # remove ID and genders
dexa2 <- dexa2[,-c(80:81)] # remove date variables
dexa2 <- dexa2[, -76] # remove "phenotype", lacks observations in groups


# application
# class of varaibles:
cls <- unlist(lapply(dexa2, class)) # c = categoric, g = gaussian, p = poisson
for (i in 1:length(cls)){
  if (cls[i]=="factor") {
    cls[i] <- "c"} else if (cls[i]=="numeric") {
        cls[i] <- "g"} else {
          cls[i] <- "p"}
}; rm(i)
names(cls) <- c()

# Levels of factor variables (non factor set to 1):
levs <- c()
for (i in 1:length(dexa2)){
  levs[i] <- length(levels(dexa2[,i]))
}; rm(i)
levs[which(levs=="0")] <- 1

# convert factors to numbers (i.e. 0L, 1L, 2L...)
dexa2$Age_cat <- as.numeric(dexa2$Age_cat)
```

```r
dexa2$BMI_cat <- as.numeric(dexa2$BMI_cat)
dexa2$Lipodystrophy <- as.numeric(dexa2$Lipodystrophy)
dexa2$Sarcopenia <- as.numeric(dexa2$Sarcopenia)
dexa2$LipoSarcop <- as.numeric(dexa2$LipoSarcop)
dexa2$Tscore_3cat <- as.numeric(dexa2$Tscore_3cat)


# Fit the model
fit_ADS2 <- mgm(data = dexa2,
                type = cls,
                level = levs,
                k = 2,
                lambdaSel = 'EBIC',
                lambdaGam = 0.5,
                pbar = TRUE)


fit_ADS3 <- mgm(data = dexa2,
                type = cls,
                level = levs,
                k = 2,
                lambdaSel = 'CV',
                lambdaFolds = 10,
                pbar = TRUE)

# set color parameters:

groups_items <- list(
  `patient variables [1-4]` = c(1:4),
  `fat/lean variables [5-28]` = c(5:28),
  `vertebrae bone variables [29-46]` = c(29:46),
  `femoral bone variables [47-58]` = c(47:58),
  `summary variables [59-72]` = c(59:72),
  `lipodystrophy [73]` = c(73),
  `sarcopenia [74]` = c(74),
  `lipo-sarco [75]` = c(75),
  `minTscore [76]` = c(76),
  `Tscore_3cat [77]` = c(77),
  `TotalBMD [78]` = c(78),
  `Disease_age [79]` = c(79)
)
col_item <- brewer.pal(length(groups_items),name = "Set3")

require(qgraph)
# EBIC 0.5
qgraph(fit_ADS2$pairwise$wadj,
       layout = 'spring', repulsion = 1.25,
       edge.color = fit_ADS2$pairwise$edgecolor,
       nodeNames = names(dexa2),
       color = col_item,
       groups = groups_items,
       legend.mode="groups", legend.cex=.45,
       vsize = 3.4, esize = 15, vTrans = 200,
```

```
       title = "MGM of all variables, gender = female",
       filename = "Graphs/9. MGM/EBICfemale",
       filetype = "png")


# 10fold CV
qgraph(fit_ADS3$pairwise$wadj,
       layout = 'spring', repulsion = 1.25,
       edge.color = fit_ADS3$pairwise$edgecolor,
       nodeNames = names(dexa2),
       color = col_item,
       groups = groups_items,
       legend.mode="groups", legend.cex=.45,
       vsize = 3.4, esize = 15, vTrans = 200,
       title = "MGM of all variables, gender = female",
       filename = "Graphs/9. MGM/CVfemale",
       filetype = "png")
```

```
# Cleanup
rm(dexa2, fit_ADS2, fit_ADS3, cls, levs, col_item, groups_items)
```

Directed analysis for Osteoporosis/Osteopenia

Random Forest with `minTscore` and all variables

```
set.seed("12345")
# Omit other predictive variables (keep minTscore)
out <- which(names(dexa_untrans) %in% c("ID", "Lipodystrophy", "LipoSarcop", "Sarcopenia",
                                        "Tscore_3cat", "phenotype", "HIV_date", "dexa_date"))
dexa2 <- dexa_untrans[, -out]; rm(out)

dexa2_m <- dexa2[which(dexa2$gender=="M"),] # select men
dexa2_f <- dexa2[which(dexa2$gender=="F"),] # select women

# Partitioning data
in_train_m <- createDataPartition(dexa2_m$minTscore, p=2/3, list = F)
test_m <- dexa2_m[-in_train_m,]

in_train_f <- createDataPartition(dexa2_f$minTscore, p=2/3, list = F)
test_f <- dexa2_f[-in_train_f,]

# Modeling
RF_m <- randomForest(minTscore ~ ., data = dexa2_m, subset = in_train_m, ntree = 100, localImp = T)
RF_f <- randomForest(minTscore ~ ., data = dexa2_f, subset = in_train_f, ntree = 100, localImp = T)


# Tune mtry:
out.of.bag_m <- c(); out.of.bag_f <- c() # Error over training
testerr_m <- c(); testerr_f <- c()       # Error over test
```

```r
for (mtry in 1:25){
  # Model
  RF_m <- randomForest(minTscore ~ ., data = dexa2_m,
                       subset = in_train_m, ntree = 100, localImp = T, mtry = mtry)
  RF_f <- randomForest(minTscore ~ ., data = dexa2_f,
                       subset = in_train_f, ntree = 100, localImp = T, mtry = mtry)
  out.of.bag_m[mtry] <- RF_m$mse[mtry]
  out.of.bag_f[mtry] <- RF_f$mse[mtry]

  # Eval
  pred_m <- predict(RF_m, test_m)
  pred_f <- predict(RF_f, test_f)
  testerr_m[mtry] <- with(test_m, mean((minTscore - pred_m)^2))
  testerr_f[mtry] <- with(test_f, mean((minTscore - pred_f)^2))
}; rm(mtry)

# Plot errors
png(file = "Graphs/7. RF/mtry_m.png")
matplot(1:25, cbind(out.of.bag_m, testerr_m), pch=20, type = "b", col = c("#00bfe9", "#00bfc4"),
        xlab = "mtry value (nº predictors)", ylab = "MSE", main = "RF minTscore (males)")
legend("topright",legend=c("OOB Error","Test Error"), pch=16, col=c("#00bfe9", "#00bfc4"))
dev.off()


png(file = "Graphs/7. RF/mtry_f.png")
matplot(1:25, cbind(out.of.bag_f, testerr_f), pch=20, type = "b", col = c("#f8666d", "#f9999d"),
        xlab = "mtry value (nº predictors)", ylab = "MSE", main = "RF minTscore (females)")
legend("topright",legend=c("OOB Error","Test Error"), pch=16, col=c("#f8666d", "#f9999d"))
dev.off()

rm(out.of.bag_m); rm(out.of.bag_f); rm(testerr_m); rm(testerr_f); rm(RF_m); rm(RF_f)
rm(pred_m); rm(pred_f)

# Models with the best performance (min errors):

# M: mtry = 11 variables

RF_m <- randomForest(minTscore ~ ., data = dexa2_m,
                       subset = in_train_m, ntree = 100, localImp = T, mtry = 11)

# F: mtry = 13 variables

RF_f <- randomForest(minTscore ~ ., data = dexa2_f,
                       subset = in_train_f, ntree = 100, localImp = T, mtry = 13)


# Print graphics
png(file = "Graphs/7. RF/RFb_m.png"); plot(RF_m); dev.off()
png(file = "Graphs/7. RF/RFb_m_variables.png")
varImpPlot(RF_m, n.var = 10, col = "#00bfe9", main = "Variable importance"); dev.off()

png(file = "Graphs/7. RF/RFb_f.png"); plot(RF_f); dev.off()
png(file = "Graphs/7. RF/RFb_f_variables.png")
```

```r
varImpPlot(RF_f, n.var = 10, col = "#f8666d", main = "Variable importance"); dev.off()

# Evaluating the model
pred_m <- predict(RF_m, test_m)
pred_f <- predict(RF_f, test_f)

# RMSE:
RMSE_m <- RMSE(pred_m, test_m$minTscore)
RMSE_f <- RMSE(pred_f, test_f$minTscore)

# Predicted vs observed plots:
png(file = "Graphs/7. RF/PvsO_b_m.png")
plot(pred_m, test_m$minTscore, main = paste("RMSE:", round(RMSE_m,2)), col = "#00bfe9",
     xlab = "Predicted values", ylab = "Observed values"); abline(a=0, b=1)
dev.off()

png(file = "Graphs/7. RF/PvsO_b_f.png")
plot(pred_f, test_f$minTscore, main = paste("RMSE:", round(RMSE_f,2)), col = "#f8666d",
     xlab = "Predicted values", ylab = "Observed values"); abline(a=0, b=1)
dev.off()
```

```r
# Cleanup
rm(RMSE_f, RMSE_m, pred_m, pred_f, dexa2_f, dexa2_m, dexa2,
   in_train_f, in_train_m, RF_f, RF_m, test_f, test_m)
```

Random Forest with `minTscore` without bone variables

```r
set.seed("12345")
# Omit other predictive variables (keep minTscore)
out <- which(names(dexa_untrans) %in% c("ID", "Lipodystrophy", "LipoSarcop", "Sarcopenia",
                                "Tscore_3cat", "phenotype", "HIV_date", "dexa_date"))
dexa2 <- dexa_untrans[, -out]; rm(out)
dexa2 <- dexa2[, -c(31:60, 76)] # omit bone variables (L1BMD to totalFZ and totalBMD)

dexa2_m <- dexa2[which(dexa2$gender=="M"),] # select men
dexa2_f <- dexa2[which(dexa2$gender=="F"),] # select women

# Partitioning data
in_train_m <- createDataPartition(dexa2_m$minTscore, p=2/3, list = F)
test_m <- dexa2_m[-in_train_m,]

in_train_f <- createDataPartition(dexa2_f$minTscore, p=2/3, list = F)
test_f <- dexa2_f[-in_train_f,]

# Modeling
RF_m <- randomForest(minTscore ~ ., data = dexa2_m, subset = in_train_m, ntree = 100, localImp = T)
RF_f <- randomForest(minTscore ~ ., data = dexa2_f, subset = in_train_f, ntree = 100, localImp = T)


# Tune mtry:
out.of.bag_m <- c(); out.of.bag_f <- c() # Error over training
```

```r
testerr_m <- c(); testerr_f <- c()       # Error over test

for (mtry in 1:25){
  # Model
  RF_m <- randomForest(minTscore ~ ., data = dexa2_m,
                       subset = in_train_m, ntree = 100, localImp = T, mtry = mtry)
  RF_f <- randomForest(minTscore ~ ., data = dexa2_f,
                       subset = in_train_f, ntree = 100, localImp = T, mtry = mtry)
  out.of.bag_m[mtry] <- RF_m$mse[mtry]
  out.of.bag_f[mtry] <- RF_f$mse[mtry]

  # Eval
  pred_m <- predict(RF_m, test_m)
  pred_f <- predict(RF_f, test_f)
  testerr_m[mtry] <- with(test_m, mean((minTscore - pred_m)^2))
  testerr_f[mtry] <- with(test_f, mean((minTscore - pred_f)^2))
}; rm(mtry)

# Plot errors
png(file = "Graphs/7. RF/mtry_m_nobone.png")
matplot(1:25, cbind(out.of.bag_m, testerr_m), pch=20, type = "b", col = c("#00bfe9", "#00bfc4"),
        xlab = "mtry value (nº predictors)", ylab = "MSE", main = "RF minTscore (males)")
legend("topright",legend=c("OOB Error","Test Error"), pch=16, col=c("#00bfe9", "#00bfc4"))
dev.off()


png(file = "Graphs/7. RF/mtry_f_nobone.png")
matplot(1:25, cbind(out.of.bag_f, testerr_f), pch=20, type = "b", col = c("#f8666d", "#f9999d"),
        xlab = "mtry value (nº predictors)", ylab = "MSE", main = "RF minTscore (females)")
legend("topright",legend=c("OOB Error","Test Error"), pch=16, col=c("#f8666d", "#f9999d"))
dev.off()

rm(out.of.bag_m); rm(out.of.bag_f); rm(testerr_m); rm(testerr_f); rm(RF_m); rm(RF_f)
rm(pred_m); rm(pred_f)

# Models with the best performance (min errors):
# M: mtry = 16 variables

RF_m <- randomForest(minTscore ~ ., data = dexa2_m,
                       subset = in_train_m, ntree = 100, localImp = T, mtry = 16)

# F: mtry = 21 variables

RF_f <- randomForest(minTscore ~ ., data = dexa2_f,
                       subset = in_train_f, ntree = 100, localImp = T, mtry = 20)


# Print graphics
png(file = "Graphs/7. RF/RFb_m_nobone.png"); plot(RF_m); dev.off()
png(file = "Graphs/7. RF/RFb_m_variables_nobone.png")
varImpPlot(RF_m, n.var = 10, col = "#00bfe9", main = "Variable importance"); dev.off()

png(file = "Graphs/7. RF/RFb_f_nobone.png"); plot(RF_f); dev.off()
```

```r
png(file = "Graphs/7. RF/RFb_f_variables_nobone.png")
varImpPlot(RF_f, n.var = 10, col = "#f8666d", main = "Variable importance"); dev.off()


# Evaluating the model
pred_m <- predict(RF_m, test_m)
pred_f <- predict(RF_f, test_f)

# RMSE:
RMSE_m <- RMSE(pred_m, test_m$minTscore)
RMSE_f <- RMSE(pred_f, test_f$minTscore)

# Predicted vs observed plots:
png(file = "Graphs/7. RF/PvsO_b_m_nobone.png")
plot(pred_m, test_m$minTscore, main = paste("RMSE:", round(RMSE_m,2)), col = "#00bfe9",
     xlab = "Predicted values", ylab = "Observed values"); abline(a=0, b=1)
dev.off()

png(file = "Graphs/7. RF/PvsO_b_f_nobone.png")
plot(pred_f, test_f$minTscore, main = paste("RMSE:", round(RMSE_f,2)), col = "#f8666d",
     xlab = "Predicted values", ylab = "Observed values"); abline(a=0, b=1)
dev.off()
```

```r
# Cleanup
rm(RMSE_f, RMSE_m, pred_m, pred_f, dexa2_f, dexa2_m, dexa2,
   in_train_f, in_train_m, RF_f, RF_m, test_f, test_m)
```

Random Forest with `Tscore_3cat`

```r
set.seed("12345")
# Omit other predictive variables (keep Tscore_3cat)
out <- which(names(dexa_untrans) %in% c("ID", "Lipodystrophy", "LipoSarcop", "Sarcopenia",
                                        "minTscore", "phenotype", "HIV_date", "dexa_date"))
dexa2 <- dexa_untrans[, -out]; rm(out)

dexa2_m <- dexa2[which(dexa2$gender=="M"),] # select men
dexa2_f <- dexa2[which(dexa2$gender=="F"),] # select women

# Partitioning data
in_train_m <- createDataPartition(dexa2_m$Tscore_3cat, p=2/3, list = F)
test_m <- dexa2_m[-in_train_m,]

in_train_f <- createDataPartition(dexa2_f$Tscore_3cat, p=2/3, list = F)
test_f <- dexa2_f[-in_train_f,]

# We can see that classification variable has unbalanced groups:
prop.table(table(dexa2_m[in_train_m,]$Tscore_3cat))
prop.table(table(dexa2_f[in_train_f,]$Tscore_3cat))

# Model will have a bias towards the majoritary class.
## Modeling
```

```r
RF_m <- randomForest(Tscore_3cat ~ ., data=dexa2_m, subset=in_train_m, ntree=600, localImp=T)
RF_f <- randomForest(Tscore_3cat ~ ., data=dexa2_f, subset=in_train_f, ntree=600, localImp=T)

## Print graphics
png(file = "Graphs/7. RF/RFb_cat_m.png")
plot(RF_m, col = c("black", "forestgreen", "goldenrod", "brown"), lty=1, lwd = 1)
legend("topright", legend=c("OOB","Healthy","Osteopenia", "Osteoporosis"),
       col = c("black", "forestgreen", "goldenrod", "brown"), pch=15)
dev.off()

png(file = "Graphs/7. RF/RFb_cat_m_variables.png")
varImpPlot(RF_m, n.var = 10, col = "#00bfe9", main = "Variable importance"); dev.off()

png(file = "Graphs/7. RF/RFb_cat_f.png")
plot(RF_f, col = c("black", "forestgreen", "goldenrod", "brown"), lty=1, lwd = 1)
legend("topright", legend=c("OOB","Healthy","Osteopenia", "Osteoporosis"),
       col = c("black", "forestgreen", "goldenrod", "brown"), pch=15)
dev.off()

png(file = "Graphs/7. RF/RFb_cat_f_variables.png")
varImpPlot(RF_f, n.var = 10, col = "#f8666d", main = "Variable importance"); dev.off()


## Evaluating the model
pred_m <- predict(RF_m, test_m)
pred_f <- predict(RF_f, test_f)

## Confusion Matrix (caret package)
CM_m <- confusionMatrix(pred_m, test_m$Tscore_3cat)
CM_f <- confusionMatrix(pred_f, test_f$Tscore_3cat)

## ROC and AUC:
pred_m <- predict(RF_m, test_m, type = "prob")
pred_f <- predict(RF_f, test_f, type = "prob")

## Edit Tscore_3cat to be ploted: dummy variables with 2 factors
test_m$dummy <- double(length(test_m$Tscore_3cat))
test_m$dummy[which(test_m$Tscore_3cat=="Osteopenia")] <- 1

test_m$dummy2 <- double(length(test_m$Tscore_3cat))
test_m$dummy2[which(test_m$Tscore_3cat=="Osteoporosis")] <- 1

test_f$dummy <- double(length(test_f$Tscore_3cat))
test_f$dummy[which(test_f$Tscore_3cat=="Osteopenia")] <- 1

test_f$dummy2 <- double(length(test_f$Tscore_3cat))
test_f$dummy2[which(test_f$Tscore_3cat=="Osteoporosis")] <- 1

p.rocm <- prediction(pred_m[,2], test_m$dummy)
p.rocm2 <- prediction(pred_m[,3], test_m$dummy2)

p.rocf <- prediction(pred_f[,2], test_f$dummy)
p.rocf2 <- prediction(pred_f[,3], test_f$dummy2)
```

```
performm <- performance(p.rocm, "tpr", "fpr")
performm2 <- performance(p.rocm2, "tpr", "fpr")
performf <- performance(p.rocf, "tpr", "fpr")
performf2 <- performance(p.rocf2, "tpr", "fpr")

## Plots:
png(file = "Graphs/7. RF/ROCm.png")
plot(performm, col="goldenrod", lwd=3, main="ROC"); plot(performm2, add=T, col="brown", lwd=3)
legend("bottomright", legend=c("Osteopenia","Osteoporosis"), col=c("goldenrod", "brown"), pch=15)
dev.off()

png(file = "Graphs/7. RF/ROCf.png")
plot(performf, col="goldenrod", lwd=3, main="ROC"); plot(performf2, add=T, col="brown", lwd=3)
legend("bottomright", legend=c("Osteopenia","Osteoporosis"), col=c("goldenrod", "brown"), pch=15)
dev.off()

## AUC values:
paste("Males:")
paste("Ostepoenia:", round(performance(p.rocm, "auc")@y.values[[1]], 4))
paste("Osteporosis:", round(performance(p.rocm2, "auc")@y.values[[1]], 4))
paste("")
paste("Females:")
paste("Ostepoenia:", round(performance(p.rocf, "auc")@y.values[[1]], 4))
paste("Osteporosis:", round(performance(p.rocf2, "auc")@y.values[[1]], 4))
```

```
# Cleanup
rm(dexa2_f, dexa2_m, in_train_f, in_train_m, RF_f, RF_m, test_f, test_m,
   CM_f, CM_m, dexa2, p.rocf, p.rocf2, p.rocm, p.rocm2, performm,
   performm2, performf, performf2, pred_f, pred_m)
```

Random Forest with `Tscore_3cat` without bone variables

```
set.seed("12345")
# Omit other predictive variables (keep Tscore_3cat)
out <- which(names(dexa_untrans) %in% c("ID", "Lipodystrophy", "LipoSarcop", "Sarcopenia",
                                        "minTscore", "phenotype", "HIV_date", "dexa_date"))
dexa2 <- dexa_untrans[, -out]; rm(out)
dexa2 <- dexa2[, -c(31:60, 76)] # remove bone variables

dexa2_m <- dexa2[which(dexa2$gender=="M"),] # select men
dexa2_f <- dexa2[which(dexa2$gender=="F"),] # select women

# Partitioning data
in_train_m <- createDataPartition(dexa2_m$Tscore_3cat, p=2/3, list = F)
test_m <- dexa2_m[-in_train_m,]

in_train_f <- createDataPartition(dexa2_f$Tscore_3cat, p=2/3, list = F)
test_f <- dexa2_f[-in_train_f,]

# Modeling
RF_m <- randomForest(Tscore_3cat ~ ., data=dexa2_m, subset=in_train_m, ntree=600, localImp=T)
```

```r
RF_f <- randomForest(Tscore_3cat ~ ., data=dexa2_f, subset=in_train_f, ntree=600, localImp=T)

# Print graphics
png(file = "Graphs/7. RF/RFb_cat_m_nobone.png")
plot(RF_m, col = c("black", "forestgreen", "goldenrod", "brown"), lty=1, lwd = 1)
legend("topright", legend=c("OOB","Healthy","Osteopenia", "Osteoporosis"),
       col = c("black", "forestgreen", "goldenrod", "brown"), pch=15)
dev.off()

png(file = "Graphs/7. RF/RFb_cat_m_variables_nobone.png")
varImpPlot(RF_m, n.var = 10, col = "#00bfe9", main = "Variable importance"); dev.off()

png(file = "Graphs/7. RF/RFb_cat_f_nobone.png")
plot(RF_f, col = c("black", "forestgreen", "goldenrod", "brown"), lty=1, lwd = 1)
legend("topright", legend=c("OOB","Healthy","Osteopenia", "Osteoporosis"),
       col = c("black", "forestgreen", "goldenrod", "brown"), pch=15)
dev.off()

png(file = "Graphs/7. RF/RFb_cat_f_variables_nobone.png")
varImpPlot(RF_f, n.var = 10, col = "#f8666d", main = "Variable importance"); dev.off()

# Evaluating the model
pred_m <- predict(RF_m, test_m)
pred_f <- predict(RF_f, test_f)

# Confusion Matrix (caret package)
CM_m <- confusionMatrix(pred_m, test_m$Tscore_3cat)
CM_f <- confusionMatrix(pred_f, test_f$Tscore_3cat)

# ROC and AUC:
pred_m <- predict(RF_m, test_m, type = "prob")
pred_f <- predict(RF_f, test_f, type = "prob")

## Edit Tscore_3cat to be ploted: dummy variables with 2 factors
test_m$dummy <- double(length(test_m$Tscore_3cat))
test_m$dummy[which(test_m$Tscore_3cat=="Osteopenia")] <- 1

test_m$dummy2 <- double(length(test_m$Tscore_3cat))
test_m$dummy2[which(test_m$Tscore_3cat=="Osteoporosis")] <- 1

test_f$dummy <- double(length(test_f$Tscore_3cat))
test_f$dummy[which(test_f$Tscore_3cat=="Osteopenia")] <- 1

test_f$dummy2 <- double(length(test_f$Tscore_3cat))
test_f$dummy2[which(test_f$Tscore_3cat=="Osteoporosis")] <- 1

p.rocm <- prediction(pred_m[,2], test_m$dummy)
p.rocm2 <- prediction(pred_m[,3], test_m$dummy2)

p.rocf <- prediction(pred_f[,2], test_f$dummy)
p.rocf2 <- prediction(pred_f[,3], test_f$dummy2)

performm <- performance(p.rocm, "tpr", "fpr")
```

```
performm2 <- performance(p.rocm2, "tpr", "fpr")
performf <- performance(p.rocf, "tpr", "fpr")
performf2 <- performance(p.rocf2, "tpr", "fpr")

# Plots:
png(file = "Graphs/7. RF/ROCm_nobone.png")
plot(performm, col="goldenrod", lwd=3, main="ROC"); plot(performm2, add=T, col="brown", lwd=3)
legend("bottomright", legend=c("Osteopenia","Osteoporosis"), col=c("goldenrod","brown"), pch=15)
dev.off()

png(file = "Graphs/7. RF/ROCf_nobone.png")
plot(performf, col="goldenrod", lwd=3, main="ROC"); plot(performf2, add=T, col="brown", lwd=3)
legend("bottomright", legend=c("Osteopenia","Osteoporosis"), col=c("goldenrod","brown"), pch=15)
dev.off()

# AUC values:
paste("Males:")
paste("Ostepoenia:", round(performance(p.rocm, "auc")@y.values[[1]], 4))
paste("Osteporosis:", round(performance(p.rocm2, "auc")@y.values[[1]], 4))
paste("")
paste("Females:")
paste("Ostepoenia:", round(performance(p.rocf, "auc")@y.values[[1]], 4))
paste("Osteporosis:", round(performance(p.rocf2, "auc")@y.values[[1]], 4))
```

```
# Cleanup
rm(dexa2_f, dexa2_m, in_train_f, in_train_m, RF_f, RF_m, test_f, test_m, CM_f,
   CM_m, dexa2, p.rocf, p.rocf2, p.rocm, p.rocm2, performm, performm2, performf,
   performf2, pred_f, pred_m, train_f, train_f_balanced, train_m, train_m_balanced)
```

```
set.seed("12345")
# Omit other predictive variables (keep Tscore_3cat)
out <- which(names(dexa_untrans) %in% c("ID", "Lipodystrophy", "LipoSarcop", "Sarcopenia",
                                        "minTscore", "phenotype", "HIV_date", "dexa_date"))
dexa2 <- dexa_untrans[, -out]; rm(out)
dexa2 <- dexa2[, -c(31:60, 76)] # Remove bone variables

dexa2_m <- dexa2[which(dexa2$gender=="M"),] # select men
dexa2_f <- dexa2[which(dexa2$gender=="F"),] # select women

# Partitioning data
in_train_m <- createDataPartition(dexa2_m$Tscore_3cat, p=2/3, list = F)
train_m <- dexa2_m[in_train_m,]
test_m <- dexa2_m[-in_train_m,]

in_train_f <- createDataPartition(dexa2_f$Tscore_3cat, p=2/3, list = F)
train_f <- dexa2_f[in_train_f,]
test_f <- dexa2_f[-in_train_f,]

# Adressing the class unbalance issue with SMOTE
test_m_balanced <- SmoteClassif(Tscore_3cat ~., test_m, C.perc = "balance", k = 5, dist="HEOM")
train_m_balanced <- SmoteClassif(Tscore_3cat ~., train_m, C.perc = "balance", k = 5, dist="HEOM")
test_f_balanced <- SmoteClassif(Tscore_3cat ~., test_f, C.perc = "balance", k = 5, dist="HEOM")
train_f_balanced <- SmoteClassif(Tscore_3cat ~., train_f, C.perc = "balance", k = 5, dist="HEOM")
```

```r
RF_m_bal <- randomForest(Tscore_3cat ~ ., data = train_m_balanced, ntree = 600, localImp = T)
RF_f_bal <- randomForest(Tscore_3cat ~ ., data = train_f_balanced, ntree = 600, localImp = T)

# Print graphics
png(file = "Graphs/7. RF/RFb_cat_m_nobone_bal.png")
plot(RF_m_bal, col = c("black", "forestgreen", "goldenrod", "brown"), lty=1, lwd = 1)
legend("topright", legend=c("OOB","Healthy","Osteopenia", "Osteoporosis"),
       col = c("black", "forestgreen", "goldenrod", "brown"), pch=15)
dev.off()

png(file = "Graphs/7. RF/RFb_cat_m_variables_nobone_bal.png")
varImpPlot(RF_m_bal, n.var = 10, col = "#00bfe9", main = "Variable importance"); dev.off()


png(file = "Graphs/7. RF/RFb_cat_f_nobone_bal.png")
plot(RF_f_bal, col = c("black", "forestgreen", "goldenrod", "brown"), lty=1, lwd = 1)
legend("topright", legend=c("OOB","Healthy","Osteopenia", "Osteoporosis"),
       col = c("black", "forestgreen", "goldenrod", "brown"), pch=15)
dev.off()

png(file = "Graphs/7. RF/RFb_cat_f_variables_nobone_bal.png")
varImpPlot(RF_f_bal, n.var = 10, col = "#f8666d", main = "Variable importance"); dev.off()

# Evaluating the model
pred_m_bal <- predict(RF_m_bal, test_m_balanced)
pred_f_bal <- predict(RF_f_bal, test_f_balanced)

# Confusion Matrix (caret package)
CM_m_bal <- confusionMatrix(pred_m_bal, test_m_balanced$Tscore_3cat)
CM_f_bal <- confusionMatrix(pred_f_bal, test_f_balanced$Tscore_3cat)

# ROC and AUC:
pred_m_bal <- predict(RF_m_bal, test_m_balanced, type = "prob")
pred_f_bal <- predict(RF_f_bal, test_f_balanced, type = "prob")

## Edit Tscore_3cat to be ploted: dummy variables with 2 factors
test_m_balanced$dummy <- double(length(test_m_balanced$Tscore_3cat))
test_m_balanced$dummy[which(test_m_balanced$Tscore_3cat=="Osteopenia")] <- 1

test_m_balanced$dummy2 <- double(length(test_m_balanced$Tscore_3cat))
test_m_balanced$dummy2[which(test_m_balanced$Tscore_3cat=="Osteoporosis")] <- 1

test_f_balanced$dummy <- double(length(test_f_balanced$Tscore_3cat))
test_f_balanced$dummy[which(test_f_balanced$Tscore_3cat=="Osteopenia")] <- 1

test_f_balanced$dummy2 <- double(length(test_f_balanced$Tscore_3cat))
test_f_balanced$dummy2[which(test_f_balanced$Tscore_3cat=="Osteoporosis")] <- 1

p.rocm_bal <- prediction(pred_m_bal[,2], test_m_balanced$dummy)
p.rocm2_bal <- prediction(pred_m_bal[,3], test_m_balanced$dummy2)

p.rocf_bal <- prediction(pred_f_bal[,2], test_f_balanced$dummy)
```

```
p.rocf2_bal <- prediction(pred_f_bal[,3], test_f_balanced$dummy2)

performm_bal <- performance(p.rocm_bal, "tpr", "fpr")
performm2_bal <- performance(p.rocm2_bal, "tpr", "fpr")
performf_bal <- performance(p.rocf_bal, "tpr", "fpr")
performf2_bal <- performance(p.rocf2_bal, "tpr", "fpr")

# Plots:
png(file = "Graphs/7. RF/ROCm_nobone_bal.png")
plot(performm_bal,col="goldenrod",lwd=3,main="ROC"); plot(performm2_bal,add=T,col="brown",lwd=3)
legend("bottomright", legend=c("Osteopenia","Osteoporosis"), col=c("goldenrod","brown"), pch=15)
dev.off()

png(file = "Graphs/7. RF/ROCf_nobone_bal.png")
plot(performf_bal,col="goldenrod",lwd=3,main="ROC"); plot(performf2_bal,add=T,col="brown",lwd=3)
legend("bottomright", legend=c("Osteopenia","Osteoporosis"), col=c("goldenrod","brown"), pch=15)
dev.off()

# AUC values:
paste("Males:")
paste("Ostepoenia:", round(performance(p.rocm_bal, "auc")@y.values[[1]], 4))
paste("Osteporosis:", round(performance(p.rocm2_bal, "auc")@y.values[[1]], 4))
paste("")
paste("Females:")
paste("Ostepoenia:", round(performance(p.rocf_bal, "auc")@y.values[[1]], 4))
paste("Osteporosis:", round(performance(p.rocf2_bal, "auc")@y.values[[1]], 4))
```

```
# Cleanup
rm(dexa2_f_bal, dexa2_m_bal, in_train_f, in_train_m, RF_f_bal, RF_m_bal,
   test_f_balanced, test_m_balanced, CM_f_bal, CM_m_bal, dexa2, p.rocf_bal,
   p.rocf2_bal, p.rocm_bal, p.rocm2_bal, performm_bal, performm2_bal,
   performf_bal, performf2_bal, pred_f_bal, pred_m_bal)
```

Random Forest with `Tscore_3cat` without bone variables, 2 categories (healthy, bone disease)

```
set.seed("12345")
# Omit other predictive variables (keep Tscore_3cat)
out <- which(names(dexa_untrans) %in% c("ID", "Lipodystrophy", "LipoSarcop", "Sarcopenia",
                                        "minTscore", "phenotype", "HIV_date", "dexa_date"))
dexa2 <- dexa_untrans[, -out]; rm(out)
dexa2 <- dexa2[, -c(31:60, 76)] # Remove bone variables

# collapse osteopenia and osteoporosis into "disease"
dexa2$Tscore_3cat <- as.character(dexa2$Tscore_3cat)
dexa2$Tscore_3cat <- factor(ifelse(dexa2$Tscore_3cat=="Healthy", "Healthy","Diseased"))


dexa2_m <- dexa2[which(dexa2$gender=="M"),] # select men
dexa2_f <- dexa2[which(dexa2$gender=="F"),] # select women
```

```r
# Partitioning data
in_train_m <- createDataPartition(dexa2_m$Tscore_3cat, p=2/3, list = F)
test_m <- dexa2_m[-in_train_m,]

in_train_f <- createDataPartition(dexa2_f$Tscore_3cat, p=2/3, list = F)
test_f <- dexa2_f[-in_train_f,]

# Modeling
RF_m <- randomForest(Tscore_3cat ~ ., data=dexa2_m, subset=in_train_m, ntree=600, localImp=T)
RF_f <- randomForest(Tscore_3cat ~ ., data=dexa2_f, subset=in_train_f, ntree=600, localImp=T)

# Print graphics
png(file = "Graphs/7. RF/RFb_cat_m_nobone2.png")
plot(RF_m, col = c("black", "forestgreen", "brown"), lty=1, lwd = 1)
legend("topright", legend=c("OOB","Healthy", "Diseased"),
       col = c("black", "forestgreen", "brown"), pch=15)
dev.off()

png(file = "Graphs/7. RF/RFb_cat_m_variables_nobone2.png")
varImpPlot(RF_m, n.var = 10, col = "#00bfe9", main = "Variable importance"); dev.off()


png(file = "Graphs/7. RF/RFb_cat_f_nobone2.png")
plot(RF_f, col = c("black", "forestgreen", "brown"), lty=1, lwd = 1)
legend("topright", legend=c("OOB","Healthy", "Diseased"),
       col = c("black", "forestgreen", "brown"), pch=15)
dev.off()

png(file = "Graphs/7. RF/RFb_cat_f_variables_nobone2.png")
varImpPlot(RF_f, n.var = 10, col = "#f8666d", main = "Variable importance"); dev.off()


# Evaluating the model
pred_m <- predict(RF_m, test_m)
pred_f <- predict(RF_f, test_f)

# Confusion Matrix (caret package)
CM_m <- confusionMatrix(pred_m, test_m$Tscore_3cat)
CM_f <- confusionMatrix(pred_f, test_f$Tscore_3cat)

# ROC and AUC:
pred_m <- predict(RF_m, test_m, type = "prob")
pred_f <- predict(RF_f, test_f, type = "prob")

## Edit Tscore_3cat to be ploted: dummy variables with 2 factors
test_m$dummy <- double(length(test_m$Tscore_3cat))
test_m$dummy[which(test_m$Tscore_3cat=="Diseased")] <- 1

test_f$dummy <- double(length(test_f$Tscore_3cat))
test_f$dummy[which(test_f$Tscore_3cat=="Diseased")] <- 1

p.rocm <- prediction(pred_m[,1], test_m$dummy)
p.rocf <- prediction(pred_f[,1], test_f$dummy)
```

```
performm <- performance(p.rocm, "tpr", "fpr")
performf <- performance(p.rocf, "tpr", "fpr")

# Plots:
png(file = "Graphs/7. RF/ROCm_nobone2.png")
plot(performm, col = "brown", lwd = 3, main = "ROC"); abline(a=0, b=1)
legend("bottomright", legend = "Diseased", col = "brown", pch=15)
dev.off()

png(file = "Graphs/7. RF/ROCf_nobone2.png")
plot(performf, col = "brown", lwd = 3, main = "ROC"); abline(a=0, b=1)
legend("bottomright", legend = "Diseased", col = "brown", pch=15)
dev.off()

# AUC values:
paste("Males:")
paste("Disease:", round(performance(p.rocm, "auc")@y.values[[1]], 4))
paste("")
paste("Females:")
paste("Disease:", round(performance(p.rocf, "auc")@y.values[[1]], 4))
```

```
# Cleanup
rm(dexa2_f, dexa2_m, in_train_f, in_train_m, RF_f, RF_m, test_f, test_m,
   CM_f, CM_m, dexa2, p.rocf, p.rocm, performm, performf, pred_m)
```

```
set.seed("12345")
# Omit other predictive variables (keep Tscore_3cat)
out <- which(names(dexa_untrans) %in% c("ID", "Lipodystrophy", "LipoSarcop", "Sarcopenia",
                                        "minTscore", "phenotype", "HIV_date", "dexa_date"))

dexa2 <- dexa_untrans[, -out]; rm(out)
dexa2 <- dexa2[, -c(31:60, 76)]

# collapse osteopenia and osteoporosis into "disease"
dexa2$Tscore_3cat <- as.character(dexa2$Tscore_3cat)
dexa2$Tscore_3cat <- factor(ifelse(dexa2$Tscore_3cat=="Healthy", "Healthy","Diseased"))


dexa2_m <- dexa2[which(dexa2$gender=="M"),] # select men
dexa2_f <- dexa2[which(dexa2$gender=="F"),] # select women

dexa2_m <- dexa2[which(dexa2$gender=="M"),] # select men
dexa2_f <- dexa2[which(dexa2$gender=="F"),] # select women

# Partitioning data
in_train_m <- createDataPartition(dexa2_m$Tscore_3cat, p=2/3, list = F)
train_m <- dexa2_m[in_train_m,]
test_m <- dexa2_m[-in_train_m,]

in_train_f <- createDataPartition(dexa2_f$Tscore_3cat, p=2/3, list = F)
train_f <- dexa2_f[in_train_f,]
test_f <- dexa2_f[-in_train_f,]
```

```r
# Adressing the class unbalance issue with SMOTE
test_m_balanced <- SmoteClassif(Tscore_3cat ~., test_m, C.perc = "balance", k = 5, dist="HEOM")
train_m_balanced <- SmoteClassif(Tscore_3cat ~., train_m, C.perc = "balance", k = 5, dist="HEOM")
test_f_balanced <- SmoteClassif(Tscore_3cat ~., test_f, C.perc = "balance", k = 5, dist="HEOM")
train_f_balanced <- SmoteClassif(Tscore_3cat ~., train_f, C.perc = "balance", k = 5, dist="HEOM")


RF_m_bal <- randomForest(Tscore_3cat ~ ., data = train_m_balanced, ntree = 600, localImp = T)
RF_f_bal <- randomForest(Tscore_3cat ~ ., data = train_f_balanced, ntree = 600, localImp = T)

# Print graphics
png(file = "Graphs/7. RF/RFb_cat_f_nobone2_bal.png")
plot(RF_f_bal, col = c("black", "forestgreen", "brown"), lty=1, lwd = 1)
legend("topright", legend=c("OOB","Healthy", "Diseased"),
       col = c("black", "forestgreen", "brown"), pch=15)
dev.off()

png(file = "Graphs/7. RF/RFb_cat_f_variables_nobone2_bal.png")
varImpPlot(RF_f_bal, n.var = 10, col = "#f8666d", main = "Variable importance"); dev.off()


png(file = "Graphs/7. RF/RFb_cat_m_nobone2_bal.png")
plot(RF_m_bal, col = c("black", "forestgreen", "brown"), lty=1, lwd = 1)
legend("topright", legend=c("OOB","Healthy", "Diseased"),
       col = c("black", "forestgreen", "brown"), pch=15)
dev.off()

png(file = "Graphs/7. RF/RFb_cat_m_variables_nobone2_bal.png")
varImpPlot(RF_m_bal, n.var = 10, col = "#00bfe9", main = "Variable importance"); dev.off()


# Evaluating the model
pred_m_bal <- predict(RF_m_bal, test_m_balanced)
pred_f_bal <- predict(RF_f_bal, test_f_balanced)

# Confusion Matrix (caret package)
CM_m_bal <- confusionMatrix(pred_m_bal, test_m_balanced$Tscore_3cat)
CM_f_bal <- confusionMatrix(pred_f_bal, test_f_balanced$Tscore_3cat)

# ROC and AUC:
pred_m_bal <- predict(RF_m_bal, test_m_balanced, type = "prob")
pred_f_bal <- predict(RF_f_bal, test_f_balanced, type = "prob")

## Edit Tscore_3cat to be ploted: dummy variables with 2 factors
test_m_balanced$dummy <- double(length(test_m_balanced$Tscore_3cat))
test_m_balanced$dummy[which(test_m_balanced$Tscore_3cat=="Diseased")] <- 1

test_f_balanced$dummy <- double(length(test_f_balanced$Tscore_3cat))
test_f_balanced$dummy[which(test_f_balanced$Tscore_3cat=="Diseased")] <- 1

p.rocm_bal <- prediction(pred_m_bal[,1], test_m_balanced$dummy)
p.rocf_bal <- prediction(pred_f_bal[,1], test_f_balanced$dummy)
```

```r
performm_bal <- performance(p.rocm_bal, "tpr", "fpr")
performf_bal <- performance(p.rocf_bal, "tpr", "fpr")

# Plots:
png(file = "Graphs/7. RF/ROCm_nobone2_bal.png")
plot(performm_bal, col = "brown", lwd = 3, main = "ROC"); abline(a=0, b=1)
legend("bottomright", legend = "Diseased", col = "brown", pch=15)
dev.off()

png(file = "Graphs/7. RF/ROCf_nobone2_bal.png")
plot(performf_bal, col = "brown", lwd = 3, main = "ROC"); abline(a=0, b=1)
legend("bottomright", legend = "Diseased", col = "brown", pch=15)
dev.off()

# AUC values:
paste("Males:")
paste("Disease:", round(performance(p.rocm_bal, "auc")@y.values[[1]], 4))
paste("")
paste("Females:")
paste("Disease:", round(performance(p.rocf_bal, "auc")@y.values[[1]], 4))
```

```r
# Cleanup
rm(dexa2_f_bal, dexa2_m_bal, in_train_f, in_train_m, RF_f_bal, RF_m_bal,
   test_f_balanced, test_m_balanced, CM_f_bal, CM_m_bal, dexa2, p.rocf_bal,
   p.rocf2_bal, p.rocm_bal, p.rocm2_bal, performm_bal, performm2_bal,
   performf_bal, performf2_bal, pred_f_bal, pred_m_bal, dexa2_f, dexa2_m,
   test_f, test_m, train_f, train_f_balanced, train_m, train_m_balanced)
```

SVM

```r
set.seed("12345")
# Omit other predictive variables (keep Tscore_3cat)
out <- which(names(dexa_untrans) %in% c("ID", "Lipodystrophy", "LipoSarcop", "Sarcopenia",
                                        "minTscore", "phenotype", "HIV_date", "dexa_date"))
dexa2 <- dexa_untrans[, -out]; rm(out)
dexa2 <- dexa2[, -c(31:60, 76)]

# scale
for (i in 1:length(dexa2)){
  if(class(dexa2[,i])=="numeric"){
    dexa2[,i] <- scale(dexa2[,i])
  }
}; rm(i)

dexa2_m <- dexa2[which(dexa2$gender=="M"),] # select men
dexa2_f <- dexa2[which(dexa2$gender=="F"),] # select women

# Partitioning data
in_train_m <- createDataPartition(dexa2_m$Tscore_3cat, p=2/3, list = F)
train_m <- dexa2_m[in_train_m,]
test_m <- dexa2_m[-in_train_m,]
```

```r
in_train_f <- createDataPartition(dexa2_f$Tscore_3cat, p=2/3, list = F)
train_f <- dexa2_f[in_train_f,]
test_f <- dexa2_f[-in_train_f,]


# Tuning the cost parameter:
for (c in 1:10){ # males
  # model
  SVM_m <- ksvm(Tscore_3cat ~ ., data=train_m, kernel="vanilladot", prob.model=T, C=c)

  # predict
  pred <- predict(SVM_m, test_m)
  CM <- confusionMatrix(pred, test_m$Tscore_3cat)
  print(paste("C val: ", c, ", Accuracy:", CM$overall[1]))
}; rm(c); rm(pred); rm(CM) # max C = 10

for (c in 1:10){ # females
  # model
  SVM_f <- ksvm(Tscore_3cat ~ ., data=train_f, kernel="vanilladot", prob.model=T, C=c)

  # predict
  pred <- predict(SVM_f, test_f)
  CM <- confusionMatrix(pred, test_f$Tscore_3cat)
  print(paste("C val: ", c, ", Accuracy:", CM$overall[1]))
}; rm(c); rm(pred); rm(CM) # max C = 3

SVM_m <- ksvm(Tscore_3cat ~ ., data=train_m, kernel="vanilladot", prob.model=T, C=10)
SVM_f <- ksvm(Tscore_3cat ~ ., data=train_f, kernel="vanilladot", prob.model=T, C=3)
# predict
pred_m <- predict(SVM_m, test_m)
pred_f <- predict(SVM_f, test_f)
CM_m <- confusionMatrix(pred_m, test_m$Tscore_3cat)
CM_f <- confusionMatrix(pred_f, test_f$Tscore_3cat)
```

```r
# Cleaning
rm(dexa2); rm(dexa2_m); rm(dexa2_f); rm(in_train_m); rm(in_train_f); rm(train_m); rm(train_f)
rm(test_m); rm(test_f); rm(SVM_m); rm(SVM_f); rm(pred_m); rm(pred_f)
```

```r
set.seed("12345")
# Omit other predictive variables (keep Tscore_3cat)
out <- which(names(dexa) %in% c("ID", "Lipodystrophy", "LipoSarcop", "Sarcopenia",
                                "minTscore", "phenotype", "HIV_date", "dexa_date"))
dexa2 <- dexa[, -out]; rm(out)
dexa2 <- dexa2[, -c(31:60, 76)]

# scale
for (i in 1:length(dexa2)){
  if(class(dexa2[,i])=="numeric"){
    dexa2[,i] <- scale(dexa2[,i])
  }
}; rm(i)

dexa2_m <- dexa2[which(dexa2$gender=="M"),] # select men
```

```r
dexa2_f <- dexa2[which(dexa2$gender=="F"),] # select women

# Partitioning data
in_train_m <- createDataPartition(dexa2_m$Tscore_3cat, p=2/3, list = F)
train_m <- dexa2_m[in_train_m,]
test_m <- dexa2_m[-in_train_m,]

in_train_f <- createDataPartition(dexa2_f$Tscore_3cat, p=2/3, list = F)
train_f <- dexa2_f[in_train_f,]
test_f <- dexa2_f[-in_train_f,]

# Balancing
test_m_balanced <- SmoteClassif(Tscore_3cat ~., test_m, C.perc = "balance", k=5, dist="HEOM")
train_m_balanced <- SmoteClassif(Tscore_3cat ~., train_m, C.perc = "balance", k=5, dist="HEOM")
test_f_balanced <- SmoteClassif(Tscore_3cat ~., test_f, C.perc = "balance", k=5, dist="HEOM")
train_f_balanced <- SmoteClassif(Tscore_3cat ~., train_f, C.perc = "balance", k=5, dist="HEOM")


# Tuning the cost parameter:
for (c in 1:10){ #males
  # model
  SVM_m <- ksvm(Tscore_3cat ~ ., data=train_m_balanced, kernel="vanilladot", prob.model=T, C=c)

  # predict
  pred <- predict(SVM_m, test_m_balanced)
  CM <- confusionMatrix(pred, test_m_balanced$Tscore_3cat)
  print(paste("C val: ", c, ", Accuracy:", CM$overall[1]))
}; rm(c); rm(pred); rm(CM) # max C = 9

for (c in 1:10){ #females
  # model
  SVM_f <- ksvm(Tscore_3cat ~ ., data=train_f_balanced, kernel="vanilladot", prob.model=T, C=c)

  # predict
  pred <- predict(SVM_f, test_f_balanced)
  CM <- confusionMatrix(pred, test_f_balanced$Tscore_3cat)
  print(paste("C val: ", c, ", Accuracy:", CM$overall[1]))
}; rm(c); rm(pred); rm(CM) # max C = 9

SVM_m <- ksvm(Tscore_3cat ~ ., data=train_m_balanced, kernel="vanilladot", prob.model=T, C=9)
SVM_f <- ksvm(Tscore_3cat ~ ., data=train_f_balanced, kernel="vanilladot", prob.model=T, C=9)
# predict
pred_m <- predict(SVM_m, test_m_balanced)
pred_f <- predict(SVM_f, test_f_balanced)
CM2_m <- confusionMatrix(pred_m, test_m_balanced$Tscore_3cat)
CM2_f <- confusionMatrix(pred_f, test_f_balanced$Tscore_3cat)
```

```r
# Cleanup
rm(dexa2, dexa2_m, dexa2_f, in_train_m, in_train_f, train_m,
   train_f, test_m, test_f, SVM_m, SVM_f, pred_m, pred_f,
   test_m_balanced, test_f_balanced, train_m_balanced, train_f_balanced)
```

```r
set.seed("12345")
# Omit other predictive variables (keep Tscore_3cat)
out <- which(names(dexa) %in% c("ID", "Lipodystrophy", "LipoSarcop", "Sarcopenia",
                                "minTscore", "phenotype", "HIV_date", "dexa_date"))
dexa2 <- dexa[, -out]; rm(out)
dexa2 <- dexa2[, -c(31:60, 76)]

# scale
for (i in 1:length(dexa2)){
  if(class(dexa2[,i])=="numeric"){
    dexa2[,i] <- scale(dexa2[,i])
  }
}; rm(i)

# Transform into 2 categories
dexa2$Tscore_3cat <- as.character(dexa2$Tscore_3cat)
for (i in 1:length(dexa2$Tscore_3cat)){
  dexa2$Tscore_3cat[i] <- ifelse(dexa2$Tscore_3cat[i] == "Healthy", "Healthy", "Diseased")
}
dexa2$Tscore_3cat <- as.factor(dexa2$Tscore_3cat)

dexa2_m <- dexa2[which(dexa2$gender=="M"),] # select men
dexa2_f <- dexa2[which(dexa2$gender=="F"),] # select women

# Partitioning data
in_train_m <- createDataPartition(dexa2_m$Tscore_3cat, p=2/3, list = F)
train_m <- dexa2_m[in_train_m,]
test_m <- dexa2_m[-in_train_m,]

in_train_f <- createDataPartition(dexa2_f$Tscore_3cat, p=2/3, list = F)
train_f <- dexa2_f[in_train_f,]
test_f <- dexa2_f[-in_train_f,]

# Balancing
test_m_balanced <- SmoteClassif(Tscore_3cat ~., test_m, C.perc="balance", k=5, dist="HEOM")
train_m_balanced <- SmoteClassif(Tscore_3cat ~., train_m, C.perc="balance", k=5, dist="HEOM")
test_f_balanced <- SmoteClassif(Tscore_3cat ~., test_f, C.perc="balance", k=5, dist="HEOM")
train_f_balanced <- SmoteClassif(Tscore_3cat ~., train_f, C.perc="balance", k=5, dist="HEOM")


# Tuning the cost parameter:
for (c in 1:10){ #males
  # model
  SVM_m <- ksvm(Tscore_3cat ~ ., data=train_m_balanced, kernel="vanilladot", prob.model=T, C=c)

  # predict
  pred <- predict(SVM_m, test_m_balanced)
  CM <- confusionMatrix(pred, test_m_balanced$Tscore_3cat)
  print(paste("C val: ", c, ", Accuracy:", CM$overall[1]))
}; rm(c); rm(pred); rm(CM) # max C = 9

for (c in 1:10){ #females
  # model
```

```
  SVM_f <- ksvm(Tscore_3cat ~ ., data=train_f_balanced, kernel="vanilladot", prob.model=T, C=c)

  # predict
  pred <- predict(SVM_f, test_f_balanced)
  CM <- confusionMatrix(pred, test_f_balanced$Tscore_3cat)
  print(paste("C val: ", c, ", Accuracy:", CM$overall[1]))
}; rm(c); rm(pred); rm(CM) # max C = 9

SVM_m <- ksvm(Tscore_3cat ~ ., data=train_m_balanced, kernel="vanilladot", prob.model=T, C=10)
SVM_f <- ksvm(Tscore_3cat ~ ., data=train_f_balanced, kernel="vanilladot", prob.model=T, C=4)
# predict
pred_m <- predict(SVM_m, test_m_balanced)
pred_f <- predict(SVM_f, test_f_balanced)
CM22_m <- confusionMatrix(pred_m, test_m_balanced$Tscore_3cat)
CM22_f <- confusionMatrix(pred_f, test_f_balanced$Tscore_3cat)
```

```
# Cleanup
rm(dexa2, dexa2_m, dexa2_f, in_train_m, in_train_f, train_m, train_f,
   test_m, test_f, SVM_m, SVM_f, pred_m, pred_f, test_m_balanced,
   test_f_balanced, train_m_balanced, train_f_balanced)
```

k-NN

```
# Omit other predictive variables (keep Tscore_3cat)

dexa2 <- dexa_untrans[,sapply(dexa_untrans, is.numeric)] # select numeric
dexa2 <- data.frame(scale(dexa2)) # Scale data to make distances equally important
dexa2 <- dexa2[, -c(28:57, 72)] # omit bone variables
dexa2$Tscore_3cat <- dexa_untrans[,"Tscore_3cat"]
dexa2$gender <- dexa_untrans$gender

# collapse osteopenia and osteoporosis into "disease"
dexa2$Tscore_3cat <- as.character(dexa2$Tscore_3cat)
dexa2$Tscore_3cat <- factor(ifelse(dexa2$Tscore_3cat=="Healthy", "Healthy", "Diseased"))


dexa2_m <- dexa2[which(dexa_untrans$gender=="M"),] # select men
dexa2_f <- dexa2[which(dexa_untrans$gender=="F"),] # select women
dexa2_m <- dexa2_m[, -44] #delete gender
dexa2_f <- dexa2_f[, -44] #delete gender

# Partitioning data
in_train_m <- createDataPartition(dexa2_m$Tscore_3cat, p=2/3, list = F)
train_m <- dexa2_m[in_train_m,]
test_m <- dexa2_m[-in_train_m,]

in_train_f <- createDataPartition(dexa2_f$Tscore_3cat, p=2/3, list = F)
train_f <- dexa2_f[in_train_f,]
test_f <- dexa2_f[-in_train_f,]

# Tunning
```

```r
for (k in 1:15){
  knn_m <- knn(train_m[, -43], test_m[, -43], cl = train_m$Tscore_3cat, k = k)
  knn_f <- knn(train_f[, -43], test_f[, -43], cl = train_f$Tscore_3cat, k = k)

  CT_m <- table(test_m$Tscore_3cat, knn_m)
  CT_f <- table(test_f$Tscore_3cat, knn_f)
  print(paste(confusionMatrix(CT_m)$overall[1], "M, k=", k))
  print(paste(confusionMatrix(CT_f)$overall[1], "F, k=", k))

}; rm(k) # M=7; F=5

knn_m <- knn(train_m[, -43], test_m[, -43], cl = train_m$Tscore_3cat, k=6)
knn_f <- knn(train_f[, -43], test_f[, -43], cl = train_f$Tscore_3cat, k=14)

CT_m <- table(test_m$Tscore_3cat, knn_m)
CT_f <- table(test_f$Tscore_3cat, knn_f)
CM3_m <- confusionMatrix(CT_m)
CM3_f <- confusionMatrix(CT_f)
```

```r
# Cleanup
rm(dexa2, dexa2_m, dexa2_f, in_train_m, in_train_f, train_m,
   train_f, test_m, test_f, knn_m, knn_f, CT_m, CT_f)
```

```r
# Omit other predictive variables (keep Tscore_3cat)

dexa2 <- dexa_untrans[,sapply(dexa_untrans, is.numeric)] # select numeric
dexa2 <- data.frame(scale(dexa2)) # Scale data to make distances equally important
dexa2 <- dexa2[, -c(28:57, 72)] # omit bone variables
dexa2$Tscore_3cat <- dexa_untrans[,"Tscore_3cat"]
dexa2$gender <- dexa_untrans$gender

# collapse osteopenia and osteoporosis into "disease"
dexa2$Tscore_3cat <- as.character(dexa2$Tscore_3cat)
dexa2$Tscore_3cat <- factor(ifelse(dexa2$Tscore_3cat=="Healthy","Healthy","Diseased"))


dexa2_m <- dexa2[which(dexa_untrans$gender=="M"),] # select men
dexa2_f <- dexa2[which(dexa_untrans$gender=="F"),] # select women
dexa2_m <- dexa2_m[, -44] #delete gender
dexa2_f <- dexa2_f[, -44] #delete gender

# Partitioning data
in_train_m <- createDataPartition(dexa2_m$Tscore_3cat, p=2/3, list = F)
train_m <- dexa2_m[in_train_m,]
test_m <- dexa2_m[-in_train_m,]

in_train_f <- createDataPartition(dexa2_f$Tscore_3cat, p=2/3, list = F)
train_f <- dexa2_f[in_train_f,]
test_f <- dexa2_f[-in_train_f,]

# Balancing
test_m_balanced <- SmoteClassif(Tscore_3cat ~., test_m, C.perc="balance", k=5, dist="HEOM")
train_m_balanced <- SmoteClassif(Tscore_3cat ~., train_m, C.perc="balance", k=5, dist="HEOM")
```

```r
test_f_balanced <- SmoteClassif(Tscore_3cat ~., test_f, C.perc="balance", k=5, dist="HEOM")
train_f_balanced <- SmoteClassif(Tscore_3cat ~., train_f, C.perc="balance", k=5, dist="HEOM")

# Tunning
for (k in 1:15){
  knn_m <- knn(train_m_balanced[,-43],test_m_balanced[,-43],cl=train_m_balanced$Tscore_3cat,k=k)
  knn_f <- knn(train_f_balanced[,-43],test_f_balanced[,-43],cl=train_f_balanced$Tscore_3cat,k=k)

  CT_m <- table(test_m_balanced$Tscore_3cat, knn_m)
  CT_f <- table(test_f_balanced$Tscore_3cat, knn_f)
  print(paste(confusionMatrix(CT_m)$overall[1], "M, k=", k))
  print(paste(confusionMatrix(CT_f)$overall[1], "F, k=", k))

}; rm(k) # M=5; F=3

knn_m <- knn(train_m_balanced[,-43],test_m_balanced[,-43],cl=train_m_balanced$Tscore_3cat,k=9)
knn_f <- knn(train_f_balanced[,-43],test_f_balanced[,-43],cl=train_f_balanced$Tscore_3cat,k=4)

CT_m <- table(test_m_balanced$Tscore_3cat, knn_m)
CT_f <- table(test_f_balanced$Tscore_3cat, knn_f)
CM4_m <- confusionMatrix(CT_m)
CM4_f <- confusionMatrix(CT_f)
```

```r
# Cleanup
rm(dexa2, dexa2_m, dexa2_f, in_train_m, in_train_f, train_m, train_f,
   test_m, test_f, knn_m, knn_f, CT_m, CT_f, test_m_balanced,
   test_f_balanced, train_m_balanced, train_f_balanced)
```