# Meat comparison report

Lisa Schneider

April 10, 2020

## Installation and namespaces

The files with the self-written functions for this workflow are loaded with source(). The first source-file "install_packages_lipidome_comparison.R" contains a function that installs all required packages. The installed packages are loaded into the namespace with library().

```
## installation script for all used packages
source("install_packages_lipidome_comparison.R")
# install_packages_lipidome_comparison() # installs all reqired packages for this workflow

## general
library(dplyr) # select part of data
library(stringr) # count separators
library(data.table) # transpose data frame
library(tibble) # data frame manipulation
library(imputeLCMD) # various imputation procedures, including left censored imputation
library(impute) # dependecy for imputeLCMD
## General graphics
library(ggplot2) # plots
library(viridis) # colorblind save color schemes
library(GGally) # paralell plot
library(fmsb) # spider chart
library(scales) # scale opacity of filling (alpha)
library(ggrepel) # avoids overlapping labels in ggplot graphs
# library(gridExtra) # arranging ggplots
library(ggpubr)
## PCA
library(FactoMineR)
library(factoextra) # graphs for PCA
library(ggfortify) # biplot with ggplot
library(corrplot)
## clustering
library(heatmaply) # interactive heatmap
library(gplots) # heatmap
library(plotly) # interactive ggplots
library(dendextend)

## self written functions
source("lipidome_comparison_dataTransformaions.R")
source("lipidome_comparison_EDA.R")
source("lipidome_comparison_pca.R")
source("lipidome_comparison_clustering.R")
source("lipidome_comparison_hypothesis_testing.R")
```

## Set directorys

This is set for my system. The syntax for windows is slightly different. Making this work on all systems, maybe from command line is planned for later.

```r
working_directory <- "/home/lisa/FH/Masterarbeit/LipidomeComparison"
setwd(working_directory)

test_path <- "/home/lisa/FH/Masterarbeit/LipidomeComparison/data/Probe-Datensatz_lisa.csv"
meat_data_path <- "/home/lisa/FH/Masterarbeit/LipidomeComparison/data/meat_fish_final_raw.csv"

plot_path <- paste(working_directory, "/plots", sep = "")
plot_name <- paste(plot_path, "/meat_data", sep = "")
```

## A lot of data preprocessing

The meat data was used without preprocessing, except exproting the excel sheet to csv. The data was read from csv to the script and stripped of meta data. The string "N/F" for not identified lipids was replaced with NA and the data was separated in target data and internal standard data. The target data was then re-formatted, with the compounds as the columns and the sample IDs as the rows. The used function flip_df() is from the "lipidome_comparison_dataTransformaions.R" file. Metadata was extracted from the sample IDs and the data set was separated by extraction method (AS and N).

```r
meat_data <- read.csv(meat_data_path, sep = ",", dec = ".", header = TRUE)
meat_data <- subset(meat_data, select = c(Compound, Type, Filename, Status, Group, Area))
meat_data <- subset(meat_data, Status == "Processed")
# meat_data[meat_data==''] <- NA
meat_data[meat_data=='N/F'] <- NA
meat_data$Area <- as.numeric(meat_data$Area)
meat_target <- subset(meat_data, Type == "Target Compound")
meat_standard <- subset(meat_data, Type == "Internal Standard")

meat_target <- flip_df(meat_target)

meat_target$SID <- sub(".*probe","sample", meat_target$SID)
meat_target$SID <- sub("\\.*pos2","2", meat_target$SID)
meat_target$SID <- sub("\\.*pos","1", meat_target$SID)

meat_target <- subset(meat_target, str_detect(meat_target$SID, "sample") == TRUE)

meat_AS <- meat_target$SID[str_detect(meat_target$SID, "AS") == TRUE]
meat_target$SID[str_detect(meat_target$SID, "AS") == TRUE] <- sub(".*sample","AS_sample", meat_AS)
meat_N <- meat_target$SID[str_detect(meat_target$SID, "AS") == FALSE]
meat_target$SID[str_detect(meat_target$SID, "AS") == FALSE] <- sub(".*sample","N_sample", meat_N)
meat_target$SID <- str_remove(meat_target$SID, "_AS")

meta_info <- read.table(text = meat_target$SID, sep = "_")
colnames(meta_info) <- c("Treatment", "Sample_nr", "Biol_rep", "Tech_rep")
meta_info$Biol_rep <- paste(meta_info$Sample_nr, meta_info$Biol_rep, sep = "_")
meta_info$Tech_rep <- paste(meta_info$Biol_rep, meta_info$Tech_rep, sep = "_")
```

```
meat_target <- cbind(meat_target$SID, meta_info, meat_target[, -1])
meat_target <- droplevels(meat_target)

map <- data.frame(Sample_nr=c("sample1","sample2","sample3",
                              "sample4", "sample5", "sample6", "sample7"),
                  Group_new=c("meat", "meat", "meat",
                              "game", "game", "fish", "meat"))
meat_target <- left_join(meat_target, map, by="Sample_nr")
meat_target$Group <- meat_target$Group_new
meat_target <- meat_target[-ncol(meat_target)]

meat_data <- meat_target
```

## Data imputation

It can be assumed that most of the missing values are due to the concentration being below the detection limit. Therefore the missing values are considdered non-random and left-censored. Imputation only works up to a certain percentage of missing values, without changing the results. I found 20% missing values in the literature (Gurke, 2019, doi: 10.3389/fpsyt.2019.00041), therefore all compounds with more than 20% missing values were filtered. The remaining missing values where imputed using the QRILC (Quantile Regression Imputation of Left-Censored data) procedure. This procedure has the disadvantage of producing negative values (because it performs random draws from a distribution), which is a problem for the calculation of the log2 foldchange (I'm working on that).

```
# remove columns where > 20% of the values are missing
impute_meat <- meat_data[, which(colMeans(!is.na(meat_data)) > 0.8)]
impute_meat <- as.matrix(select_if(impute_meat, is.numeric))

# perform missing data imputation
# meat_imputed <- as.data.frame(impute.QRILC(impute_meat, tune.sigma = 1)[[1]]) #todo imputes negative
meat_imputed <- as.data.frame(impute.MinDet(impute_meat))
meat_imputed <- cbind(meat_data[, 1:6], meat_imputed)
meat_imputed <- droplevels(meat_imputed) # remove unused levels from factors
```

## Reducing the replicates

To not artificially produce more samples than we had and to reduce variation in preprocessing and measurement the means for each sample or biological replicate are calculated.

The self-wirtten functions in this part are in "lipidome_comparison_dataTransformaions.R" and "lipidome_comparison_EDA.R".

```
#### calculate the means for the replicates
meat_groups <- generate_categorical_table(meat_imputed$Group)

meat_numeric <- meat_imputed
meat_numeric$Sample_nr <- as.numeric(meat_numeric$Sample_nr)
meat_numeric$Group <- as.numeric(meat_numeric$Group)
# meat_by_sample <- calc_by_replicate(meat_numeric, meat_numeric$Sample_nr, mean)
meat_by_replicate <- calc_by_replicate(meat_numeric, meat_numeric$Biol_rep, mean)

meat_data <- paste_catecorical_variable(meat_by_replicate, 3, meat_groups)
```

```
colnames(meat_data)[1] <- "Bio_replicate"
meat_data$Sample_nr <- paste("sample", meat_data$Sample_nr, sep = "")
```

## Exploratory data analysis

Exploratory data analysis was performed graphically and using the shapiro-wilk test. The graphical data analysis (using qqplot, histogram and boxplot) turned out not to be verry practical due to the number of compounds, but it gives a good overview over the distributions of the variables. It showed, that some of the variables had normal distribution, while others where multi modal. Therefor the results of the following tests for normal distribution and correlation have to be handelled carefully. All the graphs are in a separate zip-folder due to thier size and runtime.

```
### test for normality
meat_normality <- shapiro_by_factor(meat_data, meat_data$Group)
```

```
## [1] "p < 0.05 ... no normal distribution; p > 0.05 ... normal distribution"
```

```
meat_normality
```

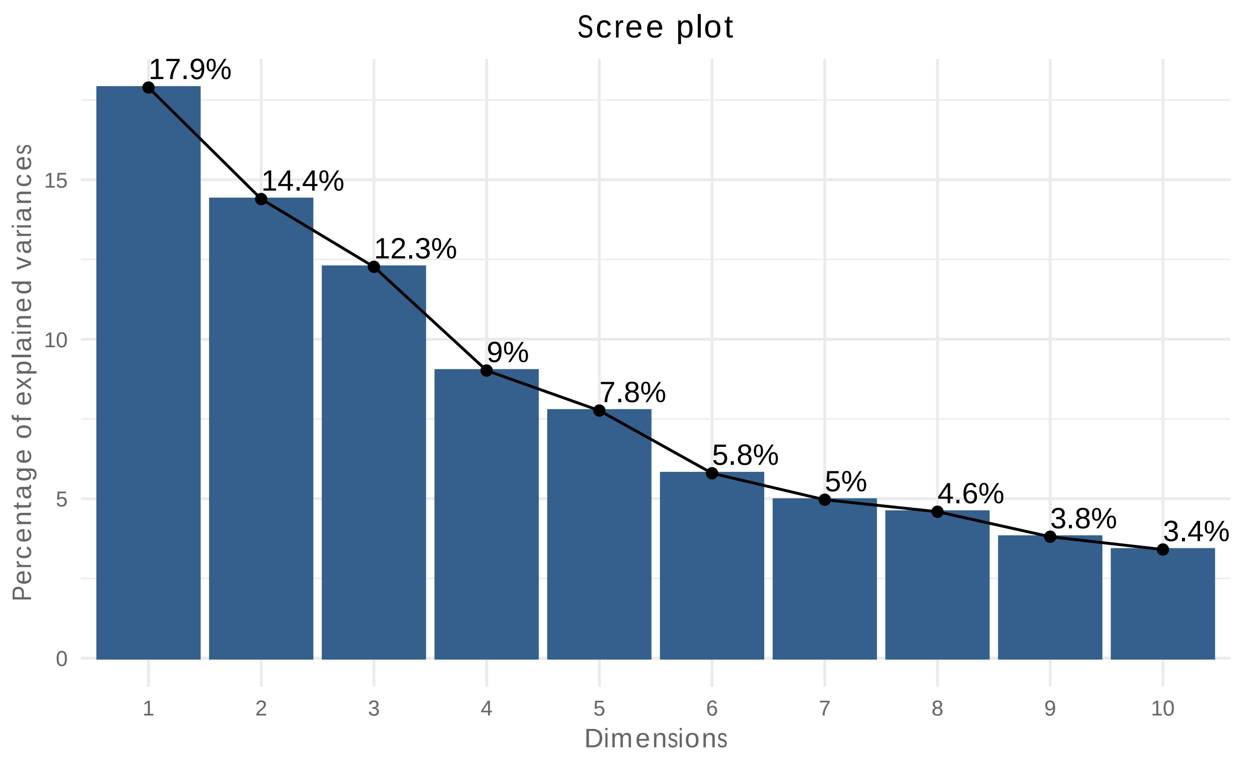
```
##   Group.1 LPC (16:0)_1 LPC (16:0)_2 LPC (16:1)_1 LPC (18:0) LPC (18:1)_1
## 1    fish    0.6242428   0.39308780   0.38606496  0.2629634  0.413618660
## 2    game    0.3562077   0.03532480   0.14474590  0.4647446  0.157442483
## 3    meat    0.4451053   0.01763662   0.03394576  0.6105926  0.009671625
##   LPC (18:1)_2 LPC (18:2)_1 LPC (18:2)_2 LPC (18:3)_1 LPC (18:3)_2 LPC (20:2)_1
## 1    0.2832570   0.42891531   0.99680759    0.4808801    0.4223639   0.44273546
## 2    0.7102686   0.46105477   0.21789486    0.8093868    0.4240651   0.20751062
## 3    0.1605349   0.01040345   0.03688884    0.3224001    0.9981067   0.03900197
##   LPC (20:3)_1 LPC (20:3)_2 LPC (20:4)_1 LPC (20:4)_2 LPC (20:4)_3 LPC (20:5)_1
## 1    0.3211409   0.11788277   0.43115971  0.412834548  0.628892756   0.69721917
## 2    0.3186737   0.69801307   0.24912333  0.232871054  0.296238812   0.17867450
## 3    0.1715485   0.04752961   0.08841033  0.001704179  0.006421907   0.04832043
##   LPC (20:5)_3 LPC (22:4)_1 LPC (22:5)_1 LPC (22:5)_2 LPC (22:6)_1
## 1   0.84684330   0.31358201    0.4959042    0.6492870   0.57499005
## 2   0.33872385   0.36337859    0.1778056    0.2051701   0.51476480
## 3   0.05556692   0.04875934    0.1413944    0.1376963   0.04377268
##   LPE (20:4)_3.51 LPE (20:4)_3.71 LPE (20:6)_3.60 LPS (18:0) PC (32:0)
## 1       0.3487725       0.9675851       0.3135978  0.6056054 0.7387012
## 2       0.5589345       0.3621009       0.5520858  0.2873857 0.1041571
## 3       0.3704043       0.4625618       0.3963873  0.3121761 0.6643278
##   PC (32:1-16:1_16:0ev18:4_14:0) PC (34:1-18:1_16:0) PC (34:2-18:2_16:0)
## 1                     0.26511121          0.08342887         0.848576194
## 2                     0.13678495          0.49553591         0.490370101
## 3                     0.01049319          0.97334888         0.004209002
##   PC (34:3-18:2_16:1ev18:3_16:0) PC (36:0)_nurExactMass PC (36:1-18:1_18:0)
## 1                      0.1194095             0.27508090         0.765099512
## 2                      0.4485266             0.07529868         0.715183111
## 3                      0.5219827             0.27197672         0.002126029
##   PC (36:2-18:2_18:0ev18:1_18:1) PC (36:3-18:2_18:1) PC (36:4-20:4_16:0)
## 1                      0.9740027           0.9226327        2.356925e-01
## 2                      0.1141029           0.3702212        3.508983e-02
## 3                      0.2334065           0.9291712        2.292548e-05
##   PC (36:5-20:5_16:0) PC (36:6-22:6_14:0) PC (38:5-20:4_18:1)
## 1           0.2358408          0.25406555           0.8123476
## 2           0.3552832          0.30477601           0.1405359
## 3           0.1755721          0.06942324           0.1880182
##   PC (38:5-22:5_16:0)_12.07 PC (38:6-22:6_16:0) PC (38:7-22:6_16:1)
## 1                 0.9244540           0.4485872           0.3869896
## 2                 0.2857862           0.8848829           0.3621547
## 3                 0.6875960           0.8749714           0.1588011
##   PC (40:6-22:6_18:0) PC (40:7-22:6_18:1) PC (40:8-22:6_18:2)
## 1           0.5423812          0.19562700          0.24526346
## 2           0.3913458          0.85857418          0.06939318
## 3           0.2216610          0.03268122          0.01356117
##   PC (40:9-22:6_18:3) PE (34:1-18:1_16:0) PE (34:2-18:2_16:0)
## 1           0.6489716           0.8027212           0.4662059
## 2           0.3063483           0.4032588           0.3276673
## 3           0.1236920           0.2989135           0.5293249
##   PE (36:1-18:1_18:0) PE (36:2-18:1_18:1) PE (36:2-18:2_18:0)
## 1          0.15506650           0.1810854          0.21273991
## 2          0.97498138           0.0567217          0.05711383
## 3          0.01700875           0.9140964          0.91377697
##   PE (36:3-18:2_18:1) PE (36:4-20:4_16:0) PE (38:4-20:4_18:0)
## 1          0.28222069           0.8248897        0.2347002745
## 2          0.03945105           0.8760868        0.9013183907
## 3          0.02065460           0.9869218        0.0001726707
##   PE (38:5-20:4_18:1ev20:5_18:0) PE (38:5-22:5_16:0)
## 1                     0.05660985          0.62918796
## 2                     0.39765356          0.28535888
## 3                     0.85176488          0.02093506
##   PE (38:6-20:5_18:1ev20:4_18:2) PE (40:5-22:5_18:0) PE (40:6-22:5_18:1)
## 1                      0.9392441          0.59833387         0.202106560
## 2                      0.2862367          0.51628222         0.007107441
## 3                      0.3933514          0.06084183         0.165859071
##   PI (36:2-18:2_18:0) PI (38:4-18:0_20:4) PI (38:5-18:0_20:5) P-PC (34:0)
## 1           0.9654037          0.50179806           0.1349395  0.28252629
## 2           0.9695227          0.59445189           0.9950586  0.17915636
## 3           0.9876204          0.00873482           0.1090971  0.03937668
##   P-PE (34:1-P16:0_18:1) P-PE (34:2-P16:0_18:2) P-PE (34:3-P16:0_18:3)
## 1            0.304747378              0.6066419              0.4038433
## 2            0.281659985              0.2917950              0.6833120
## 3            0.008852856              0.2629744              0.4194876
##   P-PE (36:0)_nurExactmass P-PE (36:1-P18:0_18:1) P-PE (36:2-P18:0_18:2)
## 1               0.48835910            0.540496791            0.288440924
## 2               0.68382183            0.216148958            0.148065489
## 3               0.08509707            0.002609115            0.005974477
##   P-PE (36:2-P18:1_18:1) P-PE (36:3-18:2_P18:1ev18:3_18:0)
## 1            0.287783091                        0.29021013
## 2            0.582744325                        0.05884234
## 3            0.005995388                        0.39955026
##   P-PE (36:4-P16:0_20:4)_12.94
## 1                    0.5389823
## 2                    0.3341493
## 3                    0.3390594
##   P-PE (36:4-P16:0_20:4ev18:3_P18:1ev18:2_P18:2)_12.52
## 1                                           0.26558569
## 2                                           0.42446923
## 3                                           0.02276557
##   P-PE (36:4-P18:1_18:3)_12.41 P-PE (36:5-P16:0_20:5) P-PE (38:4-P18:0_20:4)
## 1                    0.4670070            0.39081496            0.2032013
## 2                    0.2526467            0.54783214            0.4353353
## 3                    0.2400668            0.01648667            0.2123966
##   P-PE (38:4-P18:0_20:4evP18:1_20:3) P-PE (38:5-P16:0_22:5)
## 1                          0.3154746              0.1852298
## 2                          0.1182072              0.3985865
## 3                          0.1423442              0.1008828
##   P-PE (38:5-P18:0_20:5) P-PE (38:6-P18:0_20:6) P-PE (38:6-P18:1_20:5)
## 1              0.2296204             0.84170410             0.28390976
## 2              0.4555412             0.33230513             0.05581224
## 3              0.9079559             0.01993222             0.97471856
##   P-PE (40:5-P18:0_22:5) P-PE (40:6-P18:0_22:6) P-PE (40:6-P18:1_22:5)
## 1             0.06038325              0.3994432             0.56793538
## 2             0.30755051              0.3268960             0.41554903
## 3             0.98406774              0.6759459             0.06345796
```

```
### test for correlation
meat_correlation <- cor(select_if(meat_data, is.numeric), method = "spearman")
# correlation_heatmap(meat_data, interactive = FALSE)
```

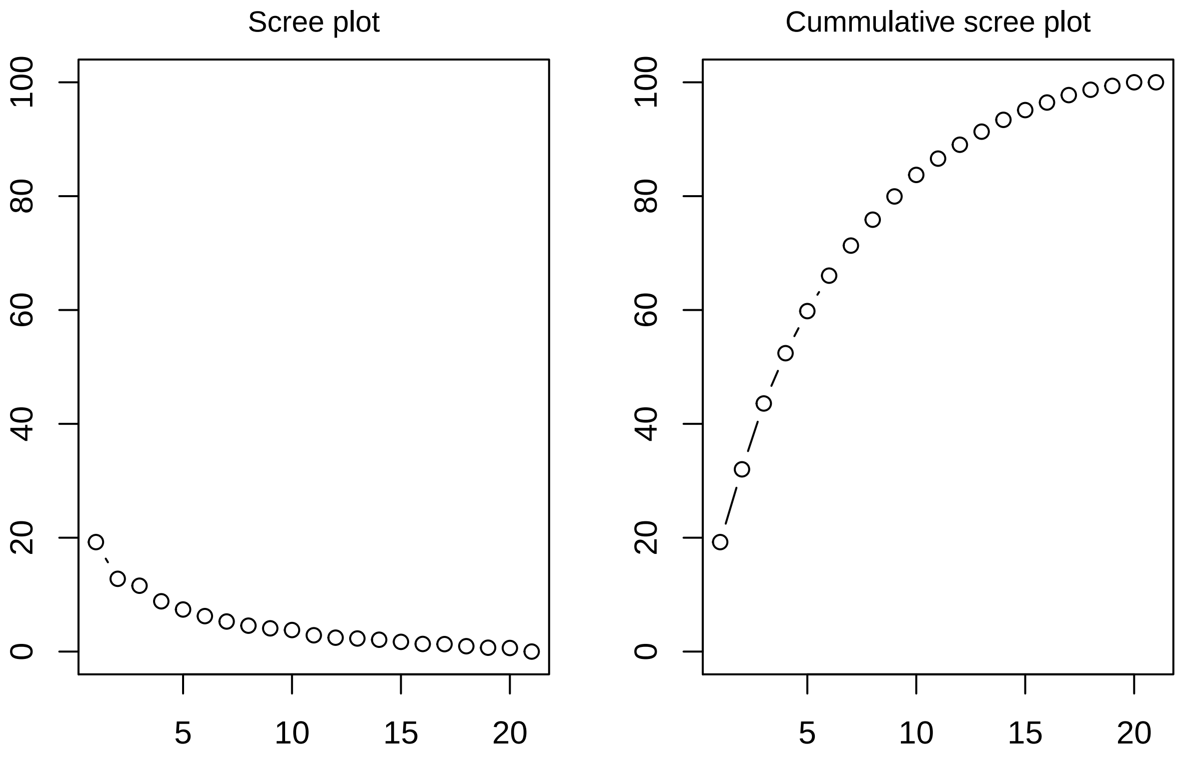
### Principal component analysis

The selection of the number of principal components depends on the explained variance displayed by get_eigenvalue() and the scree plots. There are two different representations of the scree plots, because I find both of them useful. Unfortunately in our data each PC only explains a small percentage of variance, PC1 and PC2 together explain only about 36%, therefore this model is not ideal. The problem with the separation is also visible in the biplots. While the fish-values are close together (due to them being replicates of the same sample) and well separated from the other groups, there is almost no separation between meat and game. The last two plots show which compounds have the highest contribution to the principal components. The matrix gives an overview over all five principal components and all variables. Due to the number of cmponents, the elements of the plot are verry small. A better representation are the barcharts for the top ten contributions to a selected number of principal components (I plottet PC1 and PC2). With this information the number of varibles can be reduced, for example for a parallel plot, a spider plot or for hypothesis generation and testing.

```
meat_pca <- PCA(select_if(meat_data, is.numeric), scale.unit = TRUE, graph = FALSE)
meat_eigenvalue <- get_eigenvalue(meat_pca)
scree_factoextra(meat_pca)
```

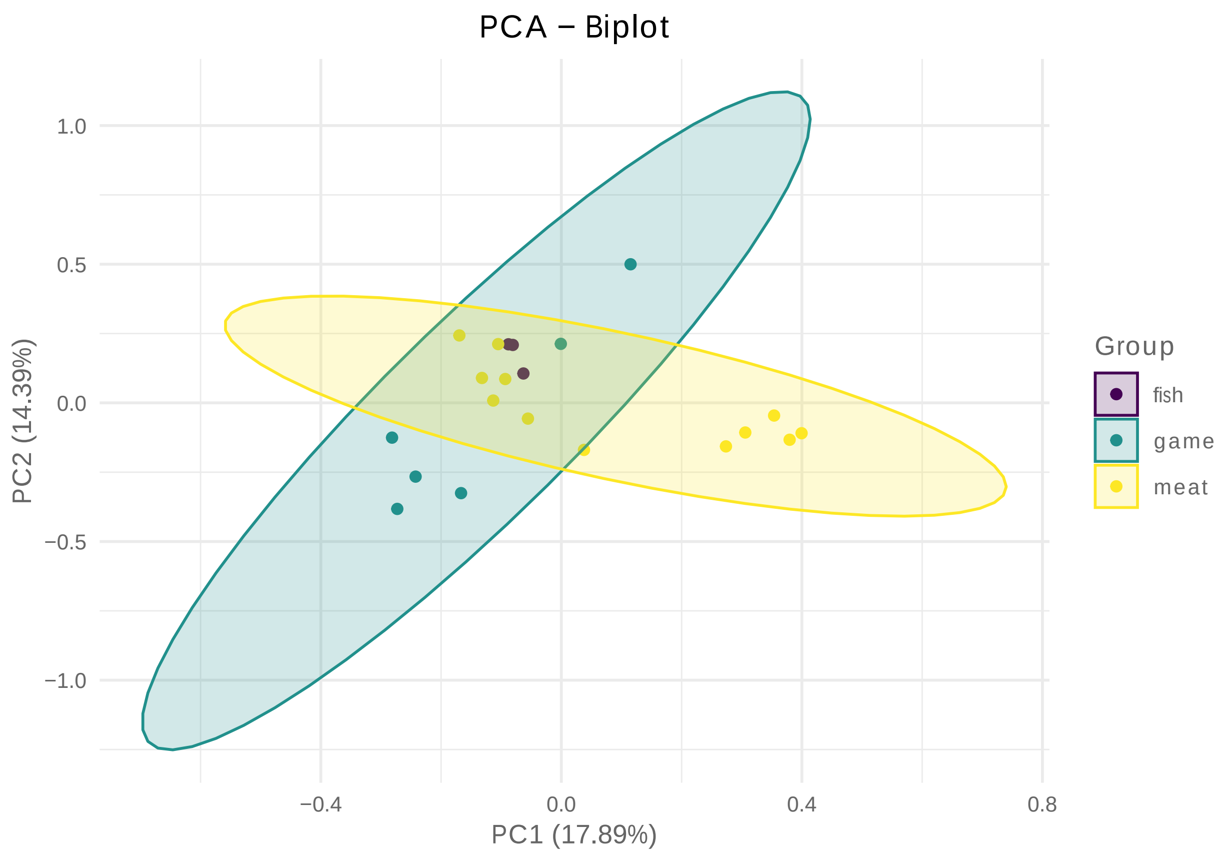


```
scree_base(select_if(meat_data, is.numeric))
```



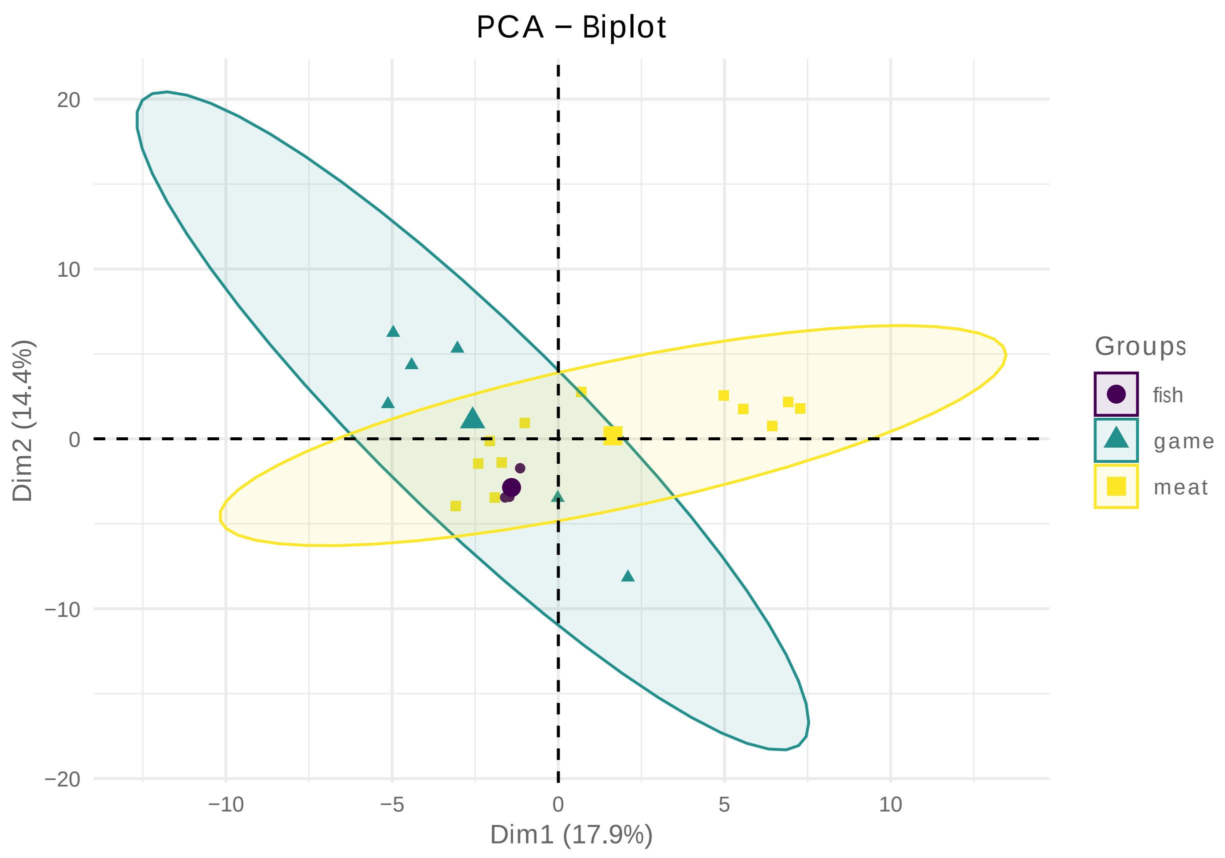
```
biplot_ggplot2(meat_data, "Group", loadings = FALSE, ellipse = TRUE, scale = TRUE)
```

```
## Too few points to calculate an ellipse
```

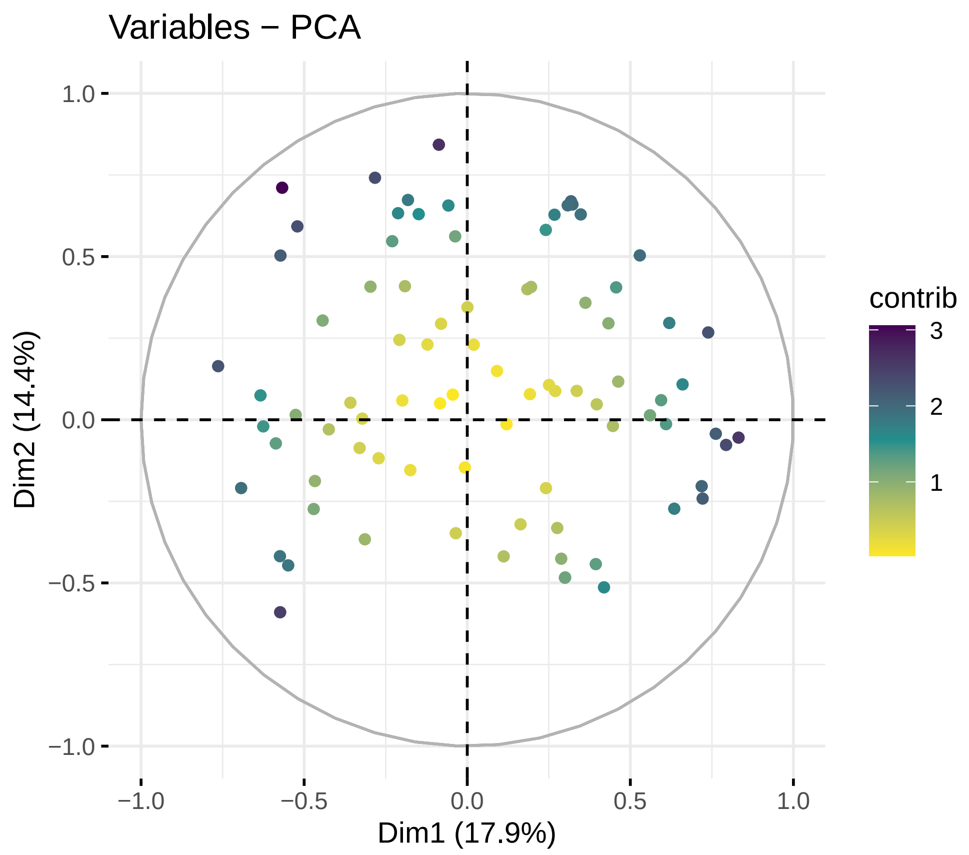


```
biplot_factoextra(meat_pca, meat_data$Group, ellipse = TRUE)
```

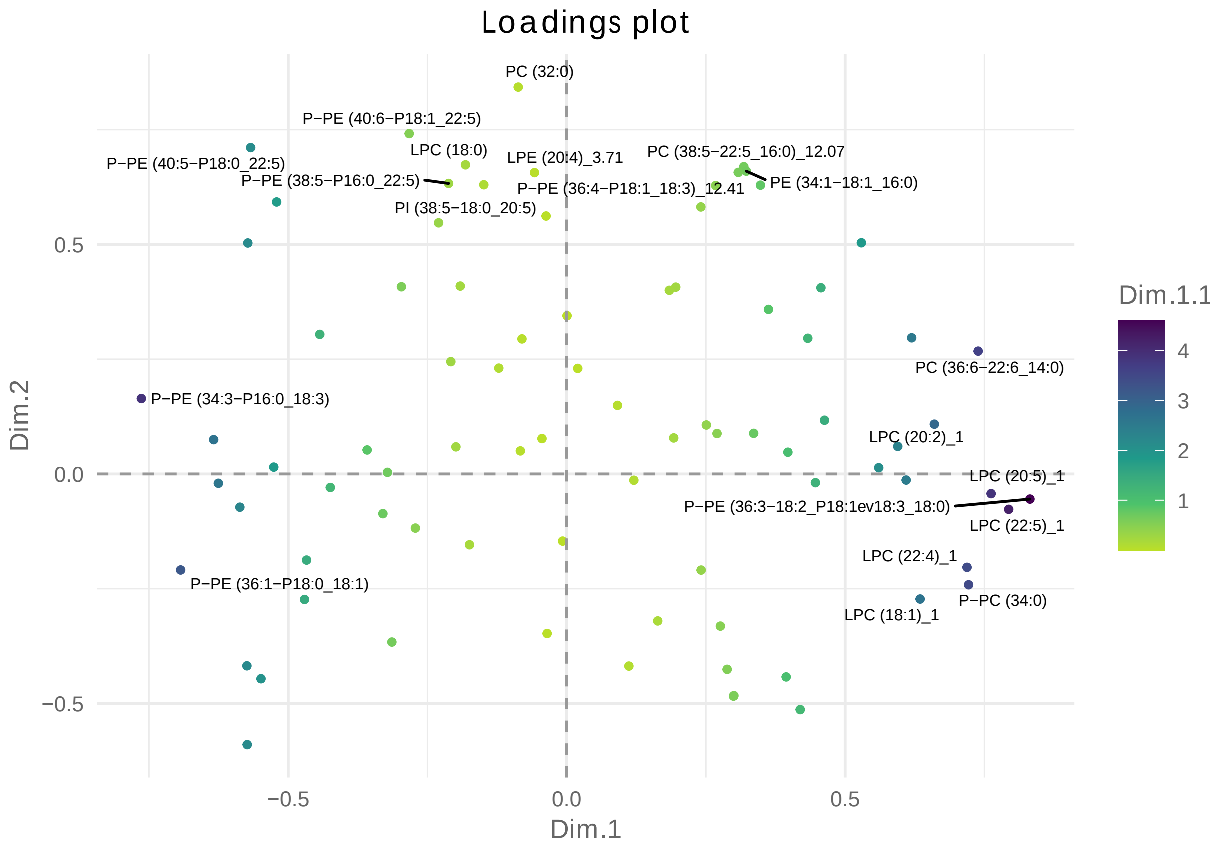
```
## Too few points to calculate an ellipse
```



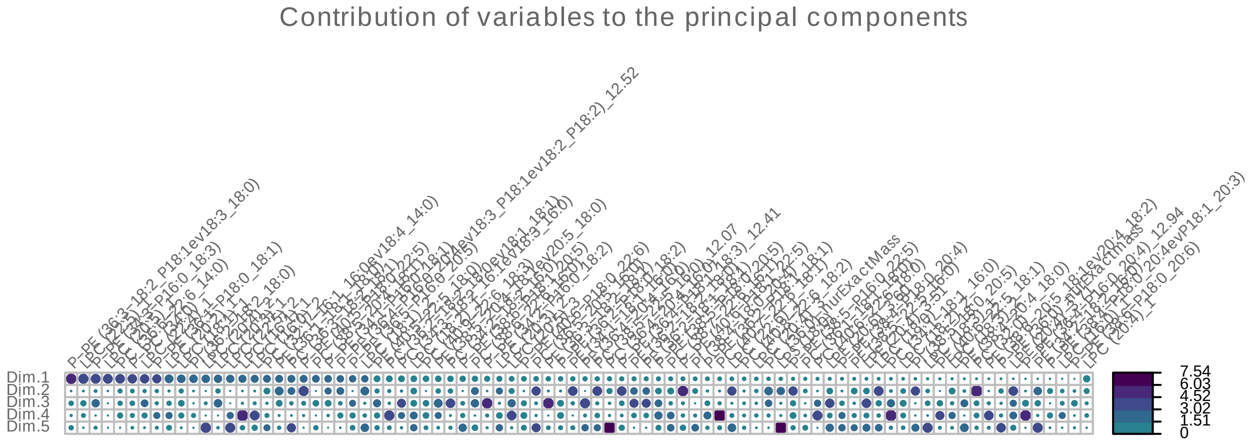
```r
fviz_pca_var(meat_pca, # factoextra
             geom = c("point"),
             col.var = "contrib",
             gradient.cols = viridis(n = 3, direction = -1),
             repel = TRUE)
```



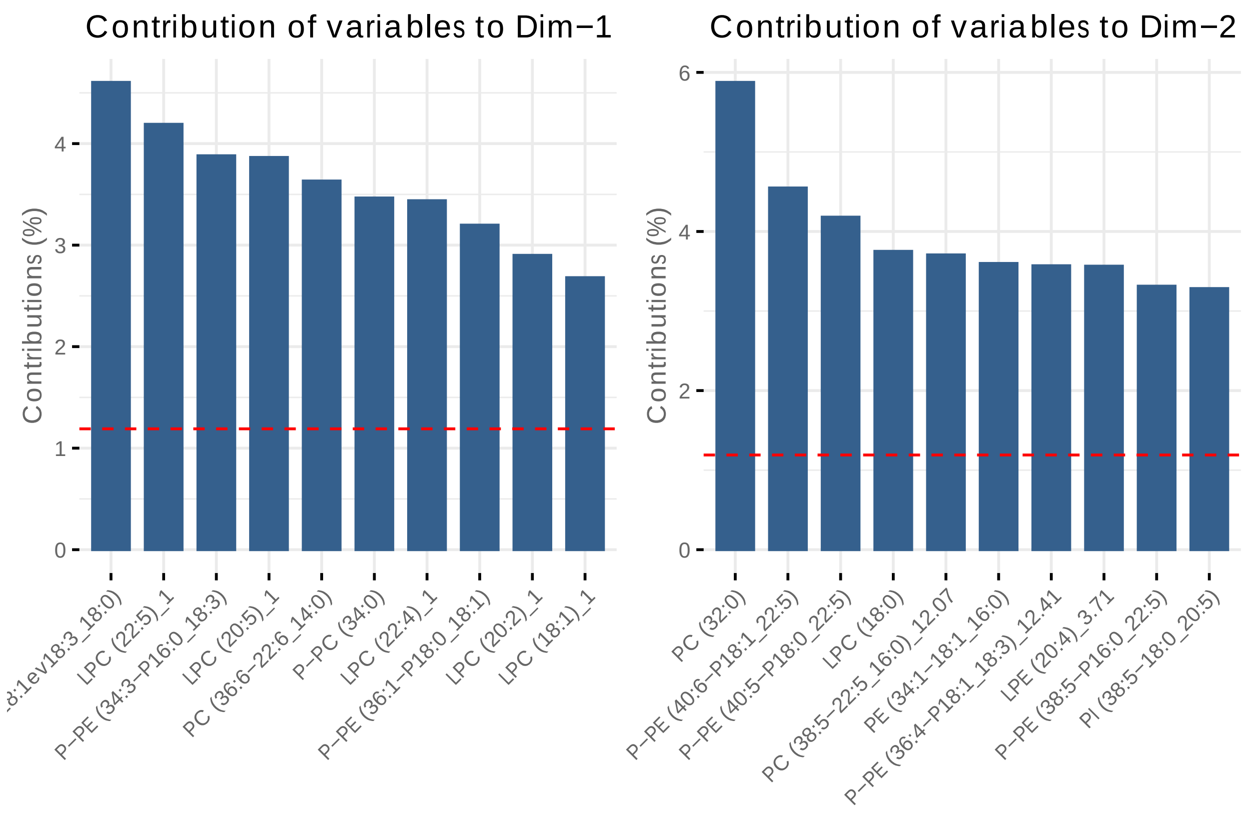
```
plot_loadings(meat_pca, colour = TRUE, top_loadings = 10) # diy wth ggplot
```



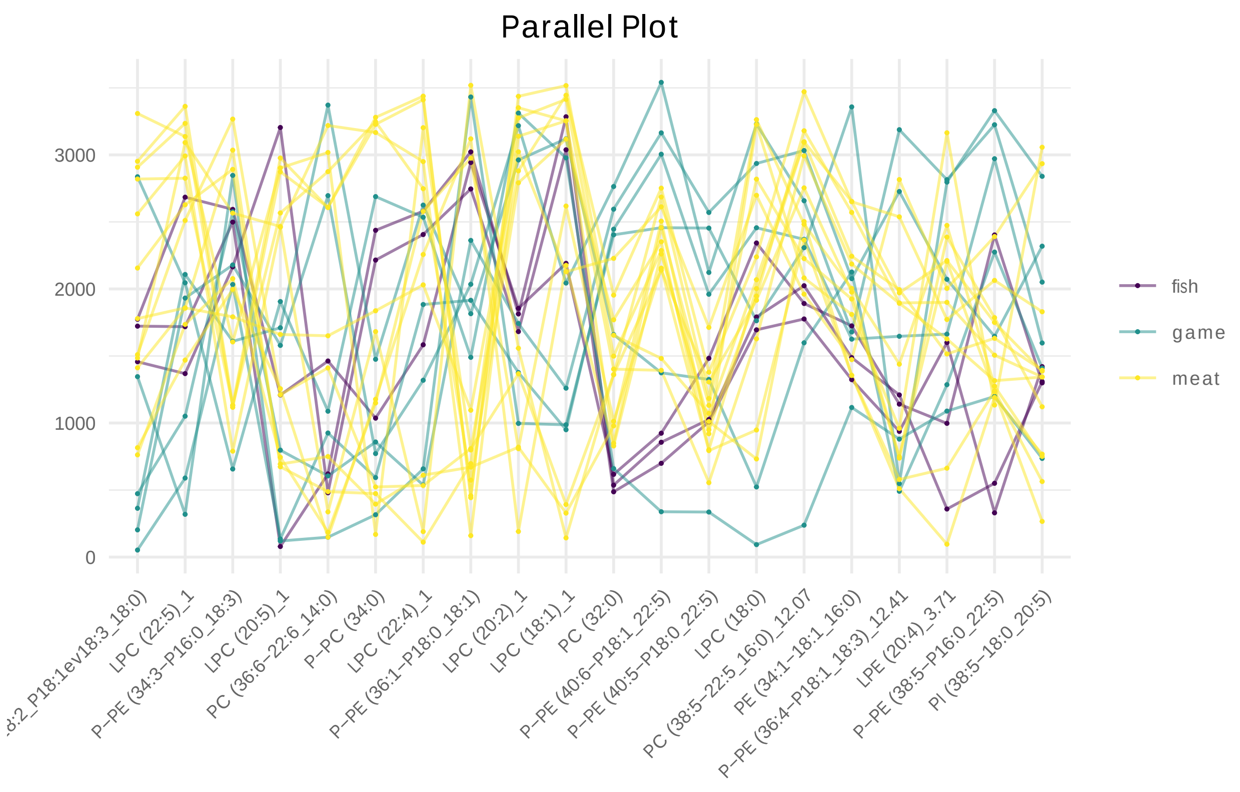
```r
plot_contrib_to_pc(meat_pca)
```



```r
pc1_contrib_plot <- fviz_contrib(meat_pca, choice = "var", axes = 1, top = 10,
             fill = viridis(n = 1, begin = 0.3), color = viridis(n = 1, begin = 0.3),
             ggtheme = my_theme)
pc2_contrib_plot <- fviz_contrib(meat_pca, choice = "var", axes = 2, top = 10,
             fill = viridis(n = 1, begin = 0.3), color = viridis(n = 1, begin = 0.3),
             ggtheme = my_theme,
             linecolor = "black")


ggarrange(pc1_contrib_plot, pc2_contrib_plot, ncol = 2, nrow = 1, align = "h", widths = c(0.9, 0.9))
```



```r
meat_var <- meat_pca$var
meat_contrib <- as.data.frame(meat_var$contrib)
pc1_contrib_table <- meat_contrib[order(meat_contrib$Dim.1, decreasing = TRUE),]
pc1_contrib <- rownames(pc1_contrib_table)[1:10]
pc2_contrib_table <- meat_contrib[order(meat_contrib$Dim.2, decreasing = TRUE),]
pc2_contrib <- rownames(pc2_contrib_table)[1:10]
meat_pca_sub <- subset(meat_data, select = c("Sample_nr", "Group", pc1_contrib, pc2_contrib))

parallel_plot(meat_pca_sub, meat_pca_sub$Group)
```



```r
means_meat_pca <- calc_by_replicate(meat_pca_sub, meat_pca_sub$Group, funct = mean)
# spider_chart(means_meat_pca, legend_lab = means_meat_pca$)
```

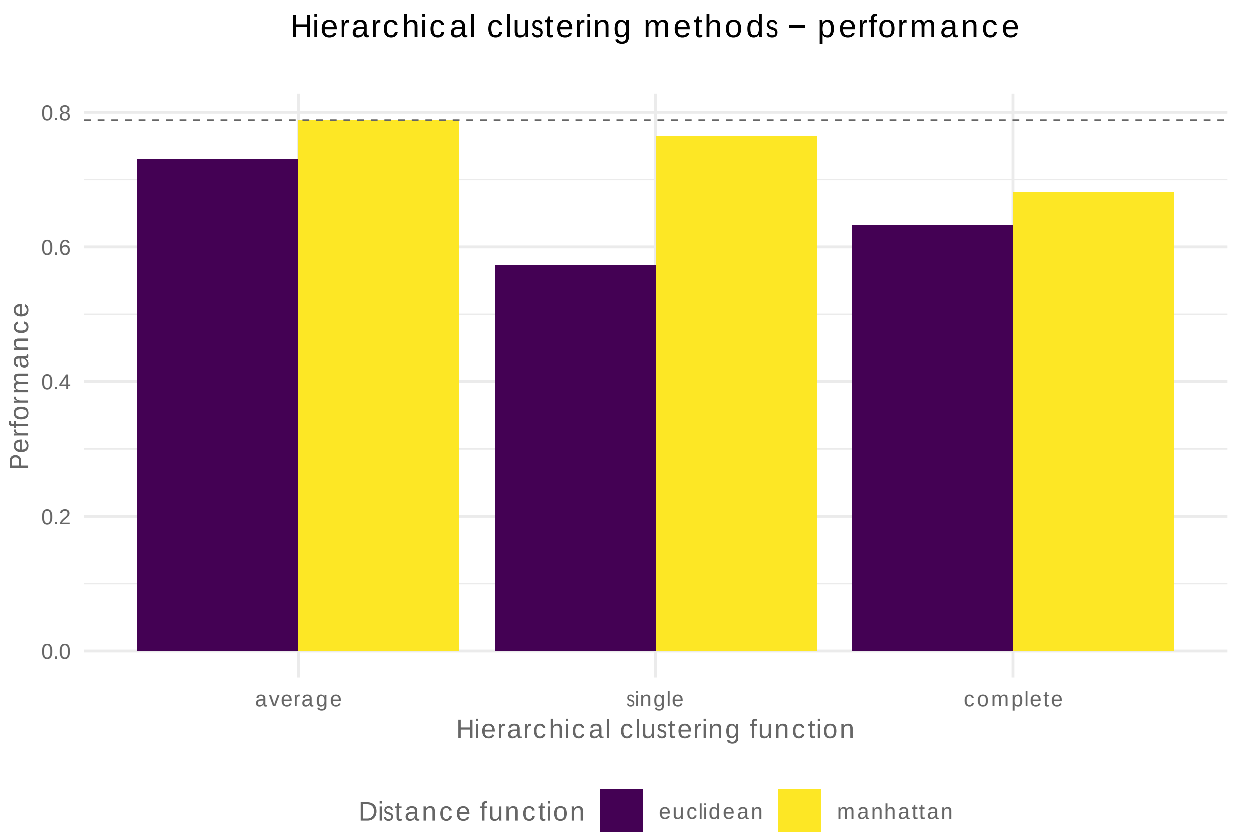
## Hierarchical clustering

The functions hclust_performance_table and hclust_performance_plot both use the dend_expend function to find the best performing distance and hierarchical clustering methods. The barplot displays the values from the "optim" column graphically. In hierarchical clustering, game is put in one cluster and meat and fish are clustered together. In the heatmap it is visible, that there are two different groups of game. In the zip-folder with the EDA-plots there is also a html-file for the interactive heatmap, because printing it into pdf did not work.

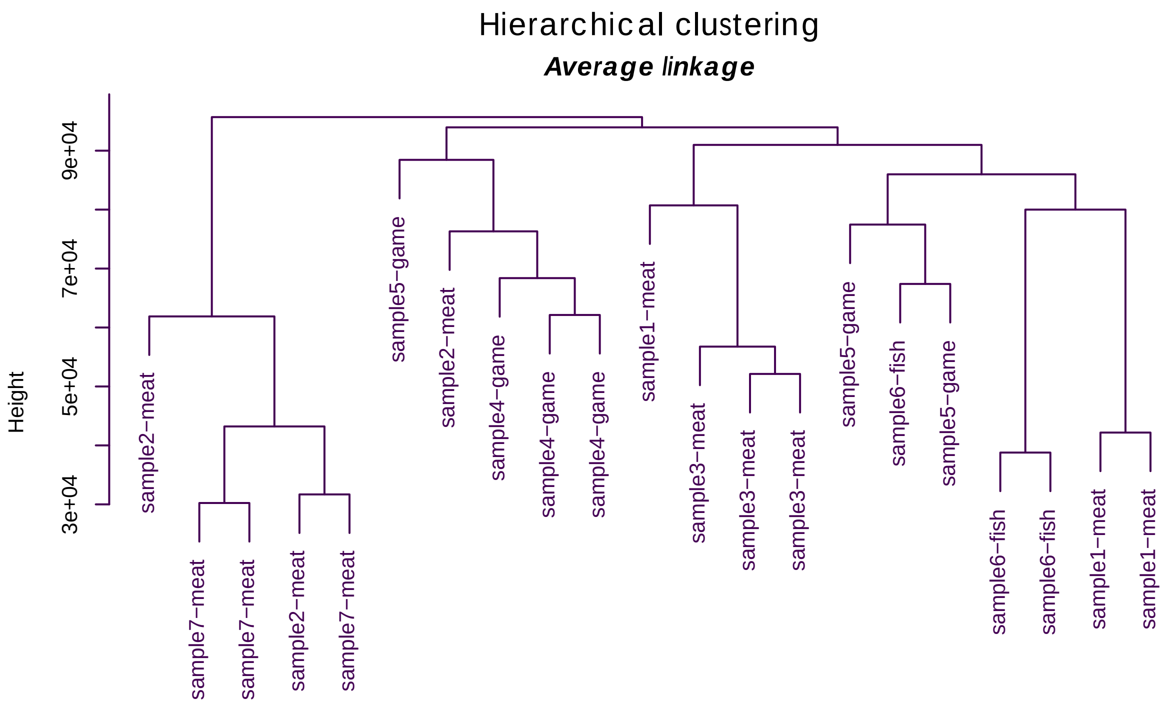
```r
meat_clust <- data.frame(Group = meat_data$Group)
meat_clust <- cbind(meat_clust, select_if(meat_data, is.numeric))
rownames(meat_clust) <- meat_data$SID

hclust_performance_table(meat_clust)
```

```
##   dist_methods hclust_methods     optim
## 1    euclidean        average 0.7297937
## 2    manhattan        average 0.7881068
## 3    euclidean         single 0.5728971
## 4    manhattan         single 0.7640592
## 5    euclidean       complete 0.6321702
## 6    manhattan       complete 0.6822020
```

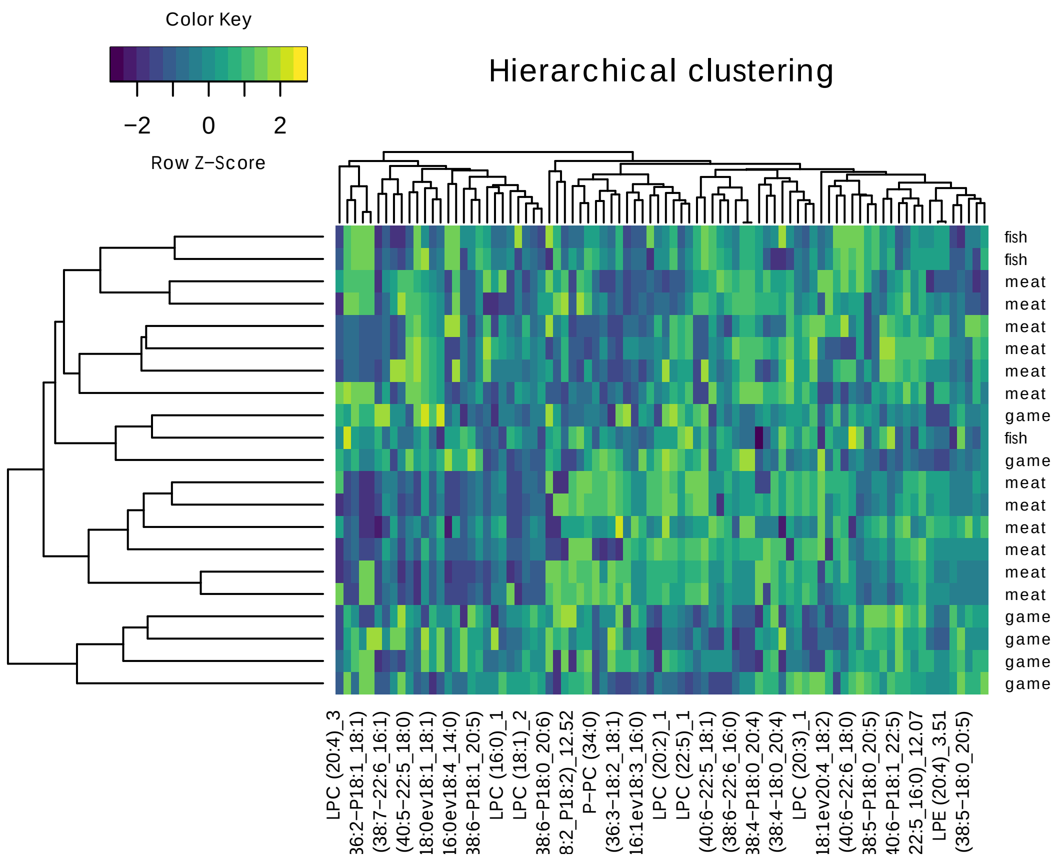
```r
hclust_performance_plot(meat_clust)
```



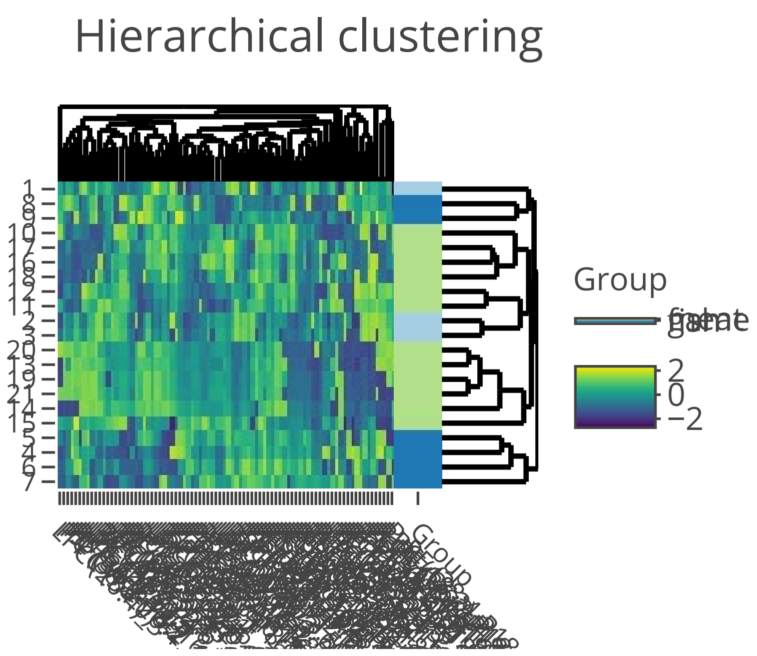
```r
meat_dist <- dist(select_if(meat_clust, is.numeric), method = "manhattan")
meat_hclust <- hclust(meat_dist, method = "average")
hclust_dendrogram(meat_hclust,
                  labs = paste(meat_data$Sample_nr,
                               meat_clust$Group, sep = "-"))
```



```r
hclust_heatmap(meat_clust,
               dist_method = "manhattan",
               hclust_method = "average",
               row_names = meat_clust$Group)
```



```r
hclust_heatmap_interactive(meat_clust,
                           dist_method = "manhattan",
                           hclust_method = "average",
                           html_path = "meat_heatmap.html")
```



### Hypothesis testing and volcano plot

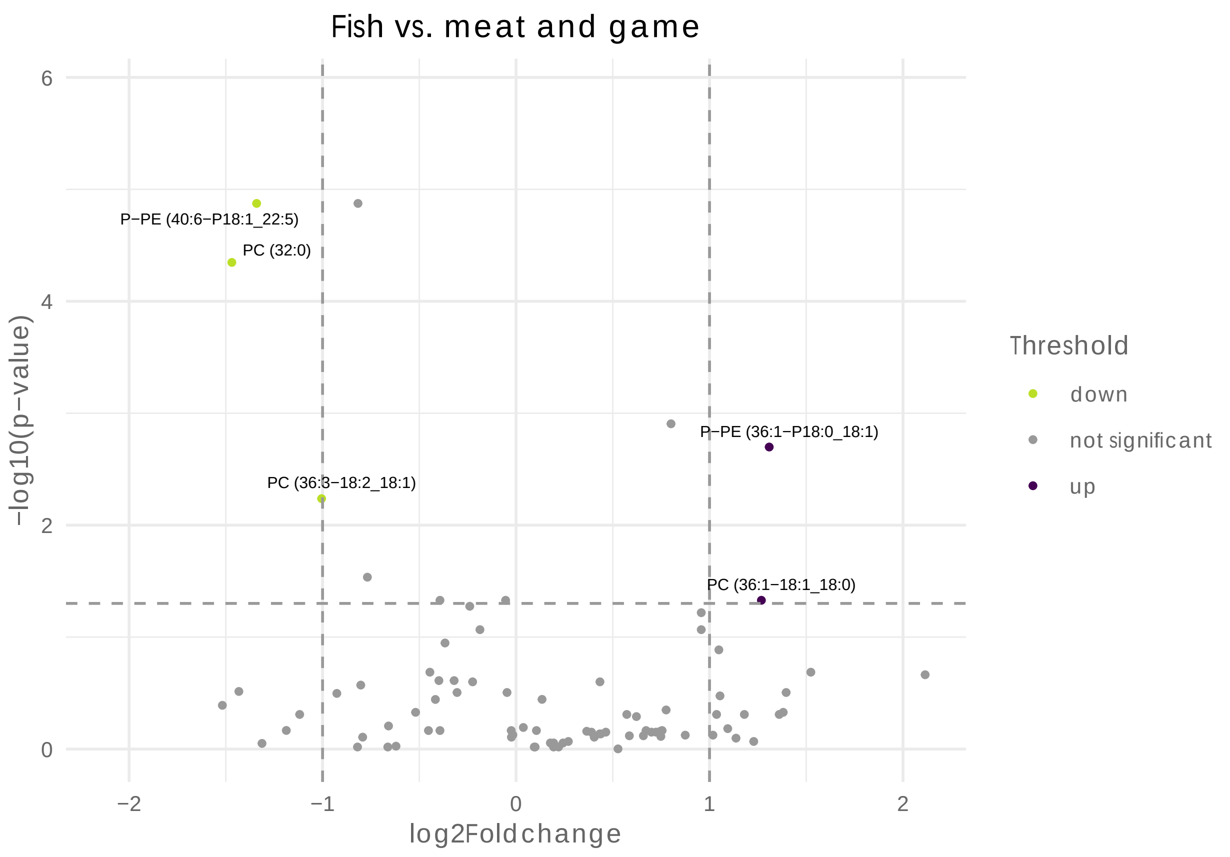
To find variables with a significant difference in at least one group, Kruskal-Wallis-test (because not all variabels were normally distributed) was performed column-wise. The resulting table can be used to select the significant variables of the original data frame for further analysis.

The volcano plots were performed for two groups each. The significance was assessed using the Wilcox rank sum test (again because of the missing normal distribution in some variables). In addition a multiple testing correction using FDR was performed. Log2-foldchange was calculated by performing log2 for all variables in both groups and then subtracting the test-group from the control-group values. There is a problem with the negative values imputed by impute.QRILC (I am working on finding out how to constrain the imputation method so it does not impute negative values): The log2-function produces NaNs from negative values. Until the imputation problem is solved, the variables with NaNs are excluded.

The volcano_plot function prints a dotplot with the log2-foldchange on the x-axis and the negative log10 of the p-value on the y-axis. The significant up- or down regulated lipids are colored and labelled. Because the number of variables is relatively low for a volcano plot (85 variables), the shape does not resemble a volcano, this should be improved with an increasing number of identified lipids.

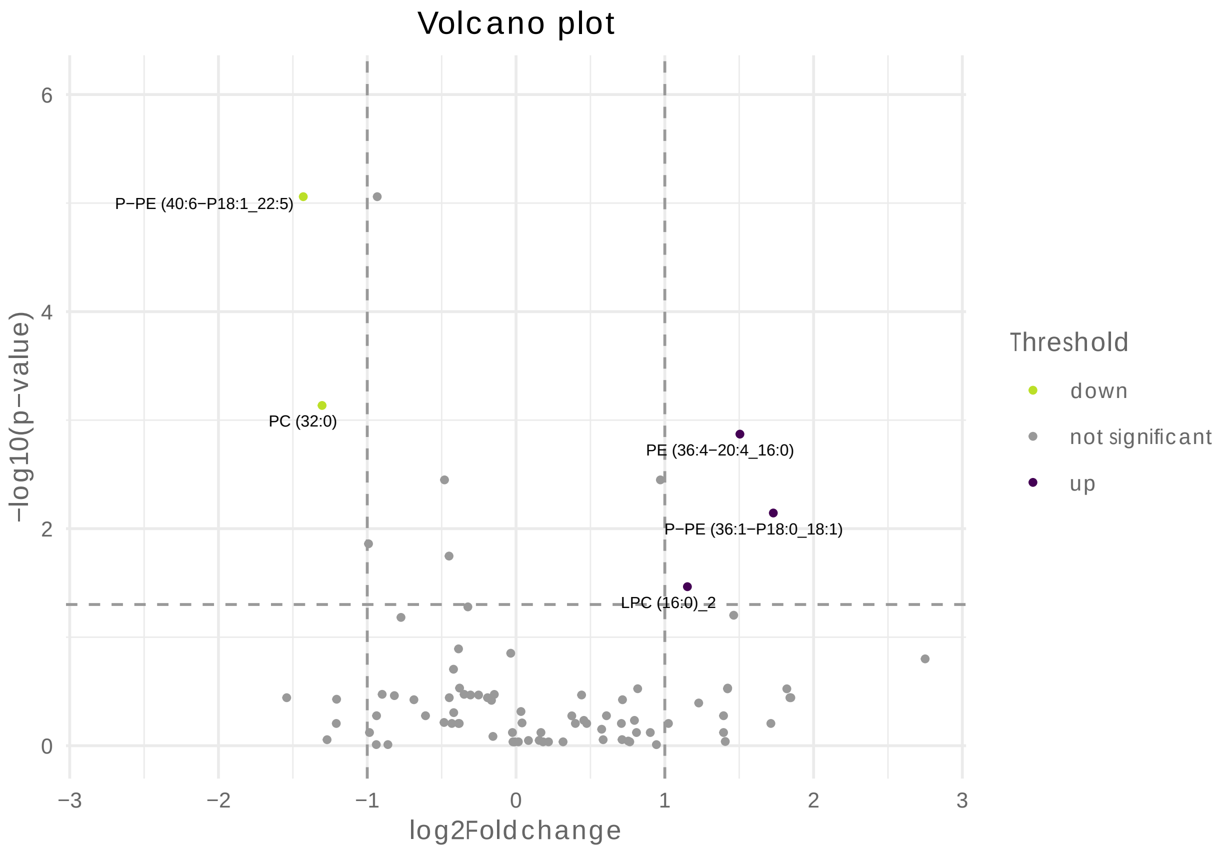
```
# anova
meat_anova <- one_way_anova_by_col(meat_data, "Group")
meat_anova$p_adj <- p.adjust(meat_anova$p_value, method = "fdr")
meat_significant_a <- subset(meat_anova, meat_anova$p_value <= 0.05)

# fish vs meat and game
fish_vs_meat_game <- meat_data
levels(fish_vs_meat_game$Group)[levels(fish_vs_meat_game$Group) == "meat" |
                                  levels(fish_vs_meat_game$Group) == "game"] <- "meat/game"

p_fish_vs_meat_game <- one_sample_test_by_col(fish_vs_meat_game, fish_vs_meat_game$Group, method = t.tes
adj_fish_vs_meat_game <- p.adjust(p_fish_vs_meat_game$p_values, method = "fdr")
fc_fish_vs_meat_game <- log2_foldchange(fish_vs_meat_game,
                                   fish_vs_meat_game$Group,
```

```r
                                   control_group = "fish",
                                   test_group = "meat/game")

volcano_df <- data.frame(p_value = p_fish_vs_meat_game, adj_p_value = adj_fish_vs_meat_game, log2_foldc
volcano_df <- volcano_df[complete.cases(volcano_df),]

volcano_plot(volcano_df,
             foldchange_col = volcano_df$log2_foldchange,
             significance_col = volcano_df$adj_p_value,
             foldchange = 1,
             significance = 0.05,
             title = "Fish vs. meat and game")
```



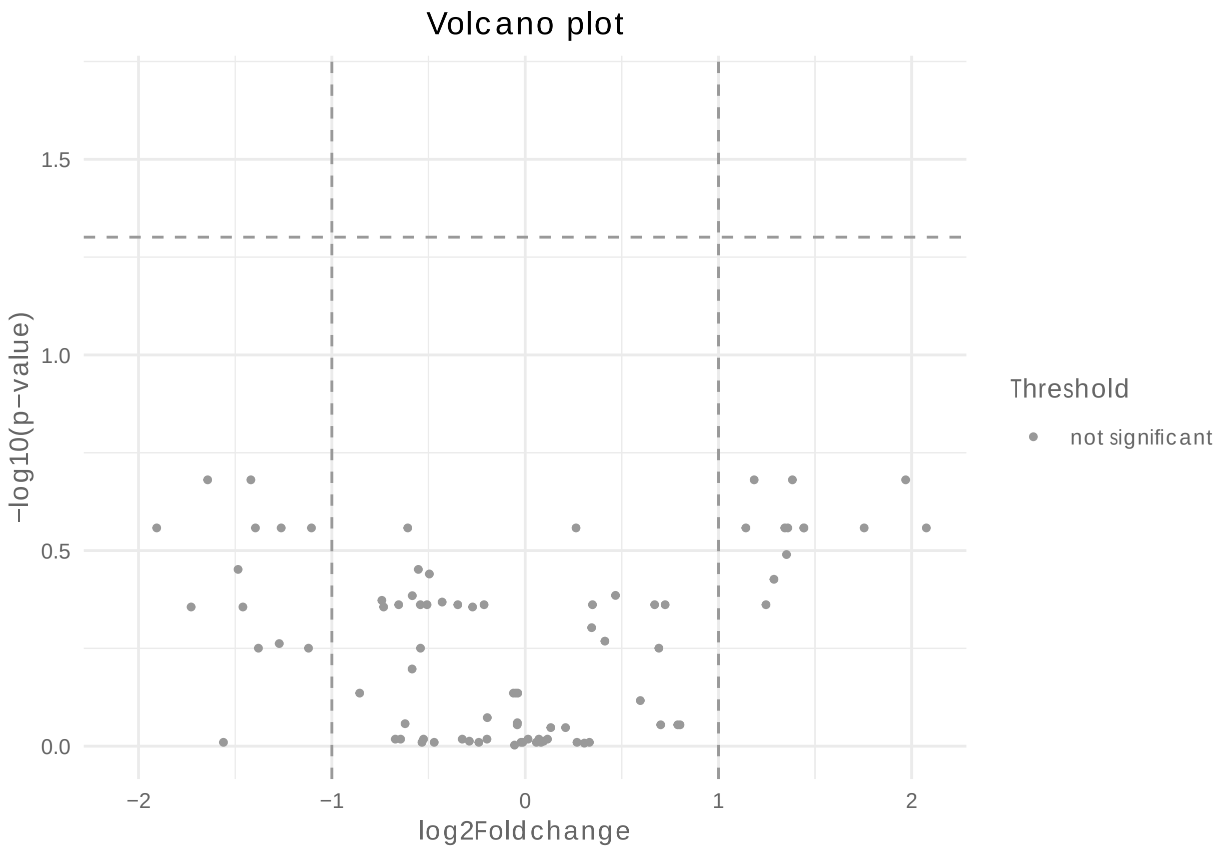
```r
{ # meat vs fish volcano plot
meat_vs_fish <- subset(meat_data, Group == "fish" | Group == "meat")
meat_vs_fish <- droplevels(meat_vs_fish)

p_meat_vs_fish <- one_sample_test_by_col(meat_vs_fish, meat_vs_fish$Group, method = t.test)
adj_meat_vs_fish <- p.adjust(p_meat_vs_fish$p_values, method = "fdr")
fc_meat_vs_fish <- log2_foldchange(meat_vs_fish,
                                   meat_vs_fish$Group,
                                   control_group = "fish",
                                   test_group = "meat")

meat_fish_volcano <- data.frame(p_value = p_meat_vs_fish, adj_p_value = adj_meat_vs_fish, log2_foldchan
meat_fish_volcano <- meat_fish_volcano[complete.cases(meat_fish_volcano),]
volcano_plot(meat_fish_volcano,
             foldchange_col = meat_fish_volcano$log2_foldchange,
             significance_col = meat_fish_volcano$adj_p_value,
             foldchange = 1,
             significance = 0.05,
             out_path = plot_name)
}
```

```
## [1] "Saving plot to /home/lisa/FH/Masterarbeit/LipidomeComparison/plots/meat_data_volcano.png"
```



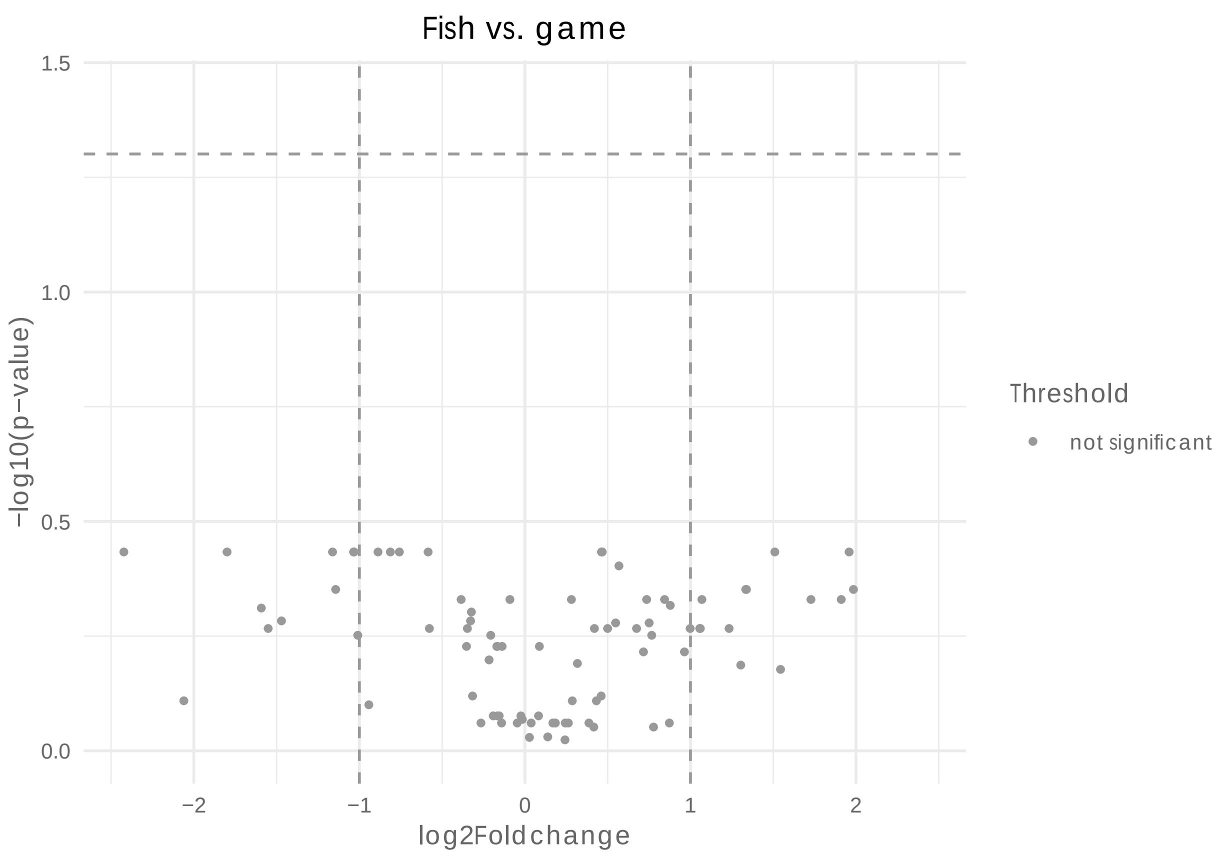
```r
{ # meat vs game volcano plot
  meat_vs_game <- subset(meat_data, Group == "game" | Group == "meat")
  meat_vs_game <- droplevels(meat_vs_game)

  p_meat_vs_game <- one_sample_test_by_col(meat_vs_game, meat_vs_game$Group, method = t.test)
  adj_meat_vs_game <- p.adjust(p_meat_vs_game$p_values, method = "fdr")
  fc_meat_vs_game <- log2_foldchange(meat_vs_game,
                                     meat_vs_game$Group,
                                     control_group = "meat",
                                     test_group = "game")

  meat_game_volcano <- data.frame(p_value = p_meat_vs_game, adj_p_value = adj_meat_vs_game, log2_foldcha
  meat_game_volcano <- meat_game_volcano[complete.cases(meat_game_volcano),]

  volcano_plot(meat_game_volcano,
               foldchange_col = meat_game_volcano$log2_foldchange,
               significance_col = meat_game_volcano$adj_p_value,
               foldchange = 1,
               out_path = plot_name)
}
```

```
## [1] "Saving plot to /home/lisa/FH/Masterarbeit/LipidomeComparison/plots/meat_data_volcano.png"
```



```
{ # game vs fish volcano plot
  # game vs fish volcano plot
  game_vs_fish <- subset(meat_data, Group == "fish" | Group == "game")
  game_vs_fish <- droplevels(game_vs_fish)

  p_game_vs_fish <- one_sample_test_by_col(game_vs_fish, game_vs_fish$Group, method = t.test)
  adj_game_vs_fish <- p.adjust(p_game_vs_fish$p_values, method = "fdr")
  fc_game_vs_fish <- log2_foldchange(game_vs_fish,
                                     game_vs_fish$Group,
                                     control_group = "fish",
                                     test_group = "game")

  volcano_df <- data.frame(p_value = p_game_vs_fish, adj_p_value = adj_game_vs_fish, log2_foldchange = f
  volcano_df <- volcano_df[complete.cases(volcano_df),]

  volcano_plot(volcano_df,
               foldchange_col = volcano_df$log2_foldchange,
               significance_col = volcano_df$adj_p_value,
               foldchange = 1,
               significance = 0.05,
               title = "Fish vs. game",
                out_path = plot_name)
}
```

```
## [1] "Saving plot to /home/lisa/FH/Masterarbeit/LipidomeComparison/plots/meat_data_volcano.png"
```



```
#
#
#
# # kruskal-wallis
# meat_kruskal <- kruskal_test_by_col(meat_data, "Group")
# meat_kruskal$p_adj <- p.adjust(meat_kruskal$p_value, method = "fdr")
# meat_significant <- subset(meat_kruskal, meat_kruskal$p_value <= 0.05)
# head(meat_significant, 3)
#
# # fish vs meat & game volcano plot
# fish_vs_meat_game <- meat_data
# levels(fish_vs_meat_game$Group)[levels(fish_vs_meat_game$Group) == "meat" |
#                                     levels(fish_vs_meat_game$Group) == "game"] <- "meat/game"
#
# p_fish_vs_meat_game <- one_sample_test_by_col(fish_vs_meat_game, fish_vs_meat_game$Group, method = wi
# adj_fish_vs_meat_game <- p.adjust(p_fish_vs_meat_game$p_values, method = "fdr")
# fc_fish_vs_meat_game <- log2_foldchange(fish_vs_meat_game,
#                                          fish_vs_meat_game$Group,
#                                          control_group = "fish",
#                                          test_group = "meat/game")
#
# volcano_df <- data.frame(p_value = p_fish_vs_meat_game, adj_p_value = adj_fish_vs_meat_game, log2_fol
# volcano_df <- volcano_df[complete.cases(volcano_df),]
#
# volcano_plot(volcano_df,
#              foldchange_col = volcano_df$log2_foldchange,
#              significance_col = volcano_df$adj_p_value,
#              foldchange = 1,
#              significance = 0.05,
#              title = "Fish vs. meat and game")
#
#
# # meat vs fish volcano plot
# meat_vs_fish <- subset(meat_data, Group == "fish" | Group == "meat")
# meat_vs_fish <- droplevels(meat_vs_fish)
#
# p_meat_vs_fish <- one_sample_test_by_col(meat_vs_fish, meat_vs_fish$Group, method = wilcox.test)
# adj_meat_vs_fish <- p.adjust(p_meat_vs_fish$p_values, method = "fdr")
# fc_meat_vs_fish <- log2_foldchange(meat_vs_fish,
#                                    meat_vs_fish$Group,
#                                    control_group = "fish",
#                                    test_group = "meat")
#
# volcano_df <- data.frame(p_value = p_meat_vs_fish, adj_p_value = adj_meat_vs_fish, log2_foldchange = 
# volcano_df <-volcano_df[complete.cases(volcano_df),]
#
# volcano_plot(volcano_df,
#              foldchange_col = volcano_df$log2_foldchange,
#              significance_col = volcano_df$adj_p_value,
#              foldchange = 1,
#              significance = 0.05,
#              title = "Fish vs. meat")
#
# # meat vs game volcano plot
# meat_vs_game <- subset(meat_imputed, Group == "game" | Group == "meat")
# meat_vs_game <- droplevels(meat_vs_game)
#
# p_meat_vs_game <- one_sample_test_by_col(meat_vs_game, meat_vs_game$Group, method = wilcox.test)
# adj_meat_vs_game <- p.adjust(p_meat_vs_game$p_values, method = "fdr")
# fc_meat_vs_game <- log2_foldchange(meat_vs_game,
#                                    meat_vs_game$Group,
#                                    control_group = "meat",
#                                    test_group = "game")
#
# volcano_df <- data.frame(p_value = p_meat_vs_game, adj_p_value = adj_meat_vs_game, log2_foldchange = 
# volcano_df <- volcano_df[complete.cases(volcano_df),]
#
# volcano_plot(volcano_df,
#              foldchange_col = volcano_df$log2_foldchange,
#              significance_col = volcano_df$adj_p_value,
#              foldchange = 1,
#              title = "Meat vs. game")
#
# # game vs fish volcano plot
# game_vs_fish <- subset(meat_imputed, Group == "fish" | Group == "game")
# game_vs_fish <- droplevels(game_vs_fish)
#
# p_game_vs_fish <- one_sample_test_by_col(game_vs_fish, game_vs_fish$Group, method = wilcox.test)
# adj_game_vs_fish <- p.adjust(p_game_vs_fish$p_values, method = "fdr")
# fc_game_vs_fish <- log2_foldchange(game_vs_fish,
#                                    game_vs_fish$Group,
#                                    control_group = "fish",
#                                    test_group = "game")
#
# volcano_df <- data.frame(p_value = p_game_vs_fish, adj_p_value = adj_game_vs_fish, log2_foldchange = 
# volcano_df <- volcano_df[complete.cases(volcano_df),]
#
# volcano_plot(volcano_df,
#              foldchange_col = volcano_df$log2_foldchange,
#              significance_col = volcano_df$adj_p_value,
#              foldchange = 1,
#              significance = 0.05,
#              title = "Fish vs. game")
```

## Outlook

- Fix imputation of negative values
- Play around with normalization methods
- Supervised learning with random forest:
  - Makes sense for establishing if and how the kind of meat can be predicted, because we already have training data.
  - Variable selection is already implemented in the randomForest r-ackage.
  - The only problem is that there is only one small data set, which might be to small to produce training and test data.
- Add option to easily run the workflow (either as a shiny app or commandline interface)
- Prepare installation script for dependencies that works on windows as well