ANM_Lipid_Preprocessing

Asger Wretlind & Petroula Proitsi

2023-09-28

Note this Markdown is based of the R-script Lipidomics_Script_QC_June2022.Step1 by Petroula Proitsi

```
#Load libraries
library(tidyverse)
```

```
## Warning: pakke 'ggplot2' blev bygget under R version 4.3.1

## Warning: pakke 'purrr' blev bygget under R version 4.3.1

## Warning: pakke 'dplyr' blev bygget under R version 4.3.1

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(here)
```

```
## here() starts at H:/Desktop/ANM_PRS/ANM_Data_Analysis
```

```
#Read in data
#Positive ionization
lipids_pos <- read.csv(here("data-raw/Positive_Mode.csv"), h=T)
dim(lipids_pos)
```

```
## [1] 874 206
```

```
#874 206

#Negative ionization
lipids_neg <- read.csv(here("data-raw/Negative_Mode.csv"), h=T)
dim(lipids_neg)
```

```
## [1] 904  74
```

```
#904 74

#Labels
labels <- read.csv(here("data-raw/Labels.csv"), h=T)
dim(labels)
```

```
## [1] 903   8
```

```
#903  8
```

```
#Checking if any lipids overlap between positive and negative
colnames(lipids_pos)[which(colnames(lipids_pos) %in% colnames(lipids_neg))]
```

```
## [1] "ID"
```

```
#no overlapping lipids

#Merge Positive and Negative mode
lipids_all.1<-merge(lipids_pos, lipids_neg, by="ID")
dim(lipids_all.1)
```

```
## [1] 874 279
```

```
#874  279

#Merge data on both modes with labels
lipids_all.2<-merge(labels, lipids_all.1, by.x="Label", by.y="ID")
dim(lipids_all.2)
```

```
## [1] 873 286
```

```
#873 286
```

```
#exclude three people who seem to have duplicated data (there were no missingness differences so we exc
lipids_dup <- lipids_all.2[duplicated(lipids_all.2$ID_SAD_VISIT),]

#the IDs of the duplicated people are "PRGCTL025_2" & "PRGCTL051_1"
#Check missing patterns to see if any of them has more missing data
#This is also a type of QCing the data to check how well duplicates correlate

lipids_all.2$ID_SAD_VISIT[duplicated(lipids_all.2$ID_SAD_VISIT)]
```

```
## [1] "PRGCTL025_2" "PRGCTL051_1"
```

```
#lipids_all.2[ lipids_all.2$ID_SAD_VISIT=="PRGCTL025_2",]

#Labels 508 and  743 are the same person(PRGCTL025_2)
summary(rowSums(is.na(lipids_all.2[ lipids_all.2$Label=="508",9:286])))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       1       1       1       1       1
```

```r
summary(rowSums(is.na(lipids_all.2[ lipids_all.2$Label=="743",9:286])))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       1       1       1       1       1
```

```r
table(is.na(lipids_all.2[ lipids_all.2$Label=="743",9:286]))
```

```
##
## FALSE  TRUE
##   277     1
```

```r
table(is.na(lipids_all.2[ lipids_all.2$Label=="508",9:286]))
```

```
##
## FALSE  TRUE
##   277     1
```

```r
#they have the same amount of missingness (1 missing)

#lipids_all.2[ lipids_all.2$ID_SAD_VISIT=="PRGCTL051_1",]
#Labels 847 and  859 are the same person(PRGCTL025_2)
summary(rowSums(is.na(lipids_all.2[ lipids_all.2$Label=="847",9:286])))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0       0       0       0       0
```

```r
summary(rowSums(is.na(lipids_all.2[ lipids_all.2$Label=="859",9:286])))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0       0       0       0       0
```

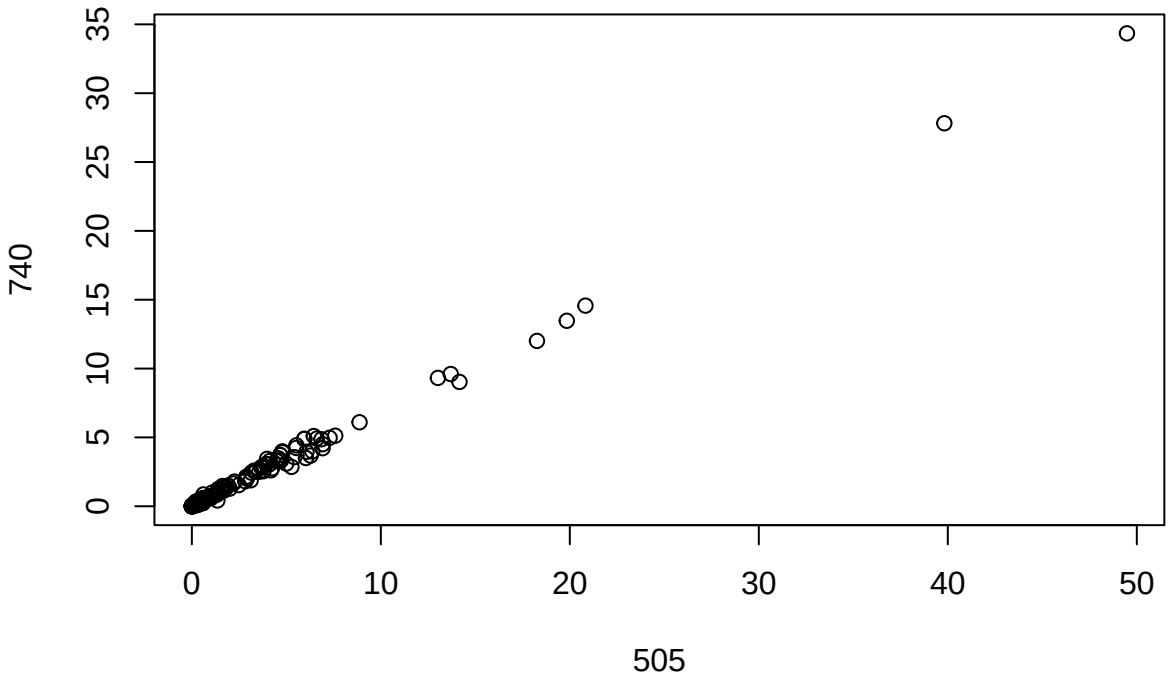
```r
table(is.na(lipids_all.2[ lipids_all.2$Label=="847",9:286]))
```

```
##
## FALSE
##   278
```

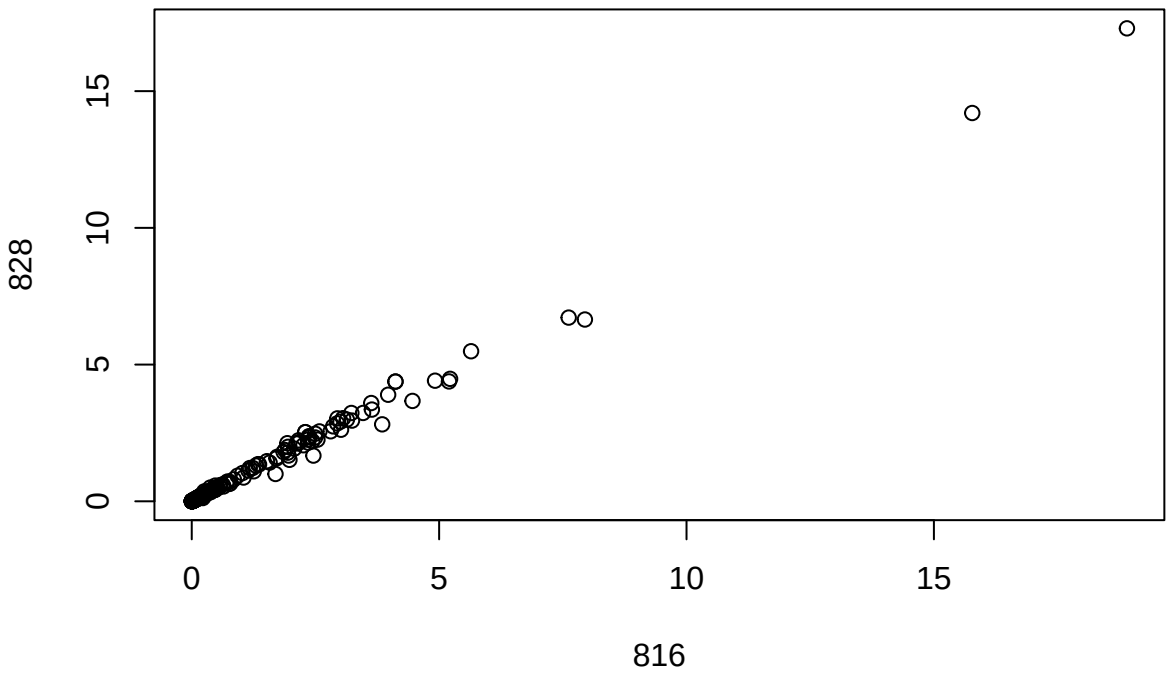
```r
table(is.na(lipids_all.2[ lipids_all.2$Label=="859",9:286]))
```

```
##
## FALSE
##   278
```

```r
#they have the same amount of missingness (0 missing)

#Plot the two pairs of data
plot(t(lipids_all.2[ lipids_all.2$ID_SAD_VISIT=="PRGCTL025_2",9:286]))
```




```
plot(t(lipids_all.2[ lipids_all.2$ID_SAD_VISIT=="PRGCTL051_1",9:286]))
```




```
#! VERY GOOD-almost PERFECT CORRELATION !

#We will now exclude randomly one of the two duplicates
lipids_all.3 <- lipids_all.2[!duplicated(lipids_all.2$ID_SAD_VISIT),]
dim(lipids_all.3)
```

```
## [1] 871 286
```

```
#871 286

dim(lipids_all.3[,9:286])
```

```
## [1] 871 278
```

```
#871 278

lipids_all.3$ID_SAD_VISIT[duplicated(lipids_all.3$ID_SAD_VISIT)]
```

```
## character(0)
```

```
rm(lipids_dup)
```

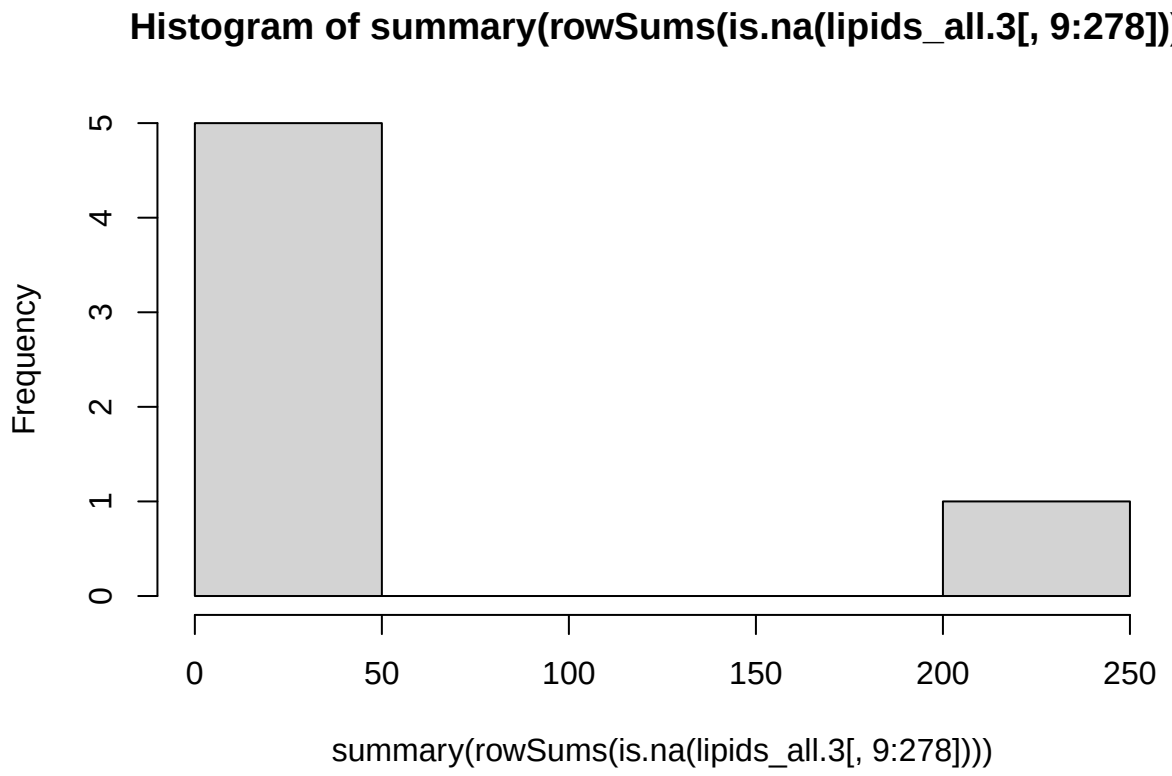
```r
#Check participants with all met data missing
lipids_all.3$allNA <- apply(lipids_all.3[,9:278],
                            MARGIN = 1, FUN = function(x) all(is.na(x)))

summary(lipids_all.3$allNA)
```

```
##    Mode   FALSE
## logical     871
```

```r
#NONE missing- remove allNA column
lipids_all.3$allNA <- NULL


#check missingess per individual
summary(rowSums(is.na(lipids_all.3[,9:278])))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   4.163   1.000 205.000
```

```r
hist(summary(rowSums(is.na(lipids_all.3[,9:278]))))
```




```r
#Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#0.000   0.000   0.000   4.163   1.000 205.000
```

```r
row.names(lipids_all.3)<-lipids_all.3$ID_SAD_VISIT


##Exclude individuals with missing data in more than 20% of lipids -!!!
lipids_all.4<-lipids_all.3[,9:278][which(rowMeans(is.na(lipids_all.3[,9:278])) <0.2),]

dim(lipids_all.4)
```

```
## [1] 857 270
```

```r
#14 individuals have more than 50% of missing-> 857 individuals left

#Exclude metabolites with missing data in more than 20% of metabolites
lipids_all.5<-lipids_all.4[,1:270][,which(colMeans(is.na(lipids_all.4[, 1:270])) < 0.2)]

dim(lipids_all.5)
```

```
## [1] 857 270
```

```r
#[1]  857 270  left
#no lipids missing more than 20%

#bind to the basic demo
lipids_all.6<-merge(lipids_all.3[,1:8],lipids_all.5, by="row.names" )
row.names(lipids_all.6)<-lipids_all.6$Row.names
lipids_all.6$Row.names<-NULL
dim(lipids_all.6)
```

```
## [1] 857 278
```

```r
library(impute)

#Create a dataframe only including lipids
lipids_only <- lipids_all.6[, 9:278]

#Perform KNN 10 Imputation
KNN_r <- impute.knn(as.matrix(lipids_only), k=10)
KNN_r2 <- as.data.frame(KNN_r$data)
dim(KNN_r2)
```

```
## [1] 857 270
```

```r
# 857 270

#Combine labels and imputed lipid data
KNN_all <- cbind(lipids_all.6[,1:8], KNN_r2)
dim(KNN_all)
```

```
## [1] 857 278
```

```r
# 857 278

#Clean
rm(lipids_only, KNN_r, KNN_r2)
rm(lipids_all.1, lipids_all.2, lipids_all.3, lipids_all.4,
   lipids_all.5, lipids_all.6, lipids_pos, lipids_neg, labels)
```

```r
#specific lipid name fix
colnames(KNN_all)[colnames(KNN_all) %in% "PE.36.2."] <- "PE.36.2._A"

colnames(KNN_all)[colnames(KNN_all) %in% "PE.36.3.b"] <- "PE.36.3._D"

#Fix some naming inconsistencies
colnames(KNN_all) <- sub("\\.b", "\\._B", colnames(KNN_all))
colnames(KNN_all) <- sub("\\.a", "\\._C", colnames(KNN_all))

#Turns out PE.36.2._A and PE.36.2._C are actual duplicates
#and so are PC.38.4._A and PC.38.4._B
#PE.36.2._C and PC.38.4._B are dropped

KNN_all <- KNN_all %>%
    select(!all_of(c("PE.36.2._C", "PC.38.4._B")))

# #Create short names and calculate RSD
# tmp_RSD <- tibble(KNN_all) %>%
#     select(where(is.double) & contains("_")) %>%
#     summarise(across(everything(), ~ (sd(.)/mean(.))*100)) %>%
#     pivot_longer(everything(), names_to = "Long", values_to = "RSD") %>%
#     mutate("Short" = gsub("_.", "", Long))
#
# #Find the lowest RSD for each short name, merge with long names based on match RSD,
# #Extract names of the features to keep
# peaks_to_keep <- tmp_RSD %>%
#     select(-Long) %>%
#     pivot_wider(names_from = Short, values_from = "RSD",
#                 values_fn = list) %>%
#     summarise(across(everything(), ~ min(.[[1]]))) %>%
#     pivot_longer(everything(), names_to = "Short", values_to = "RSD") %>%
#     left_join(., tmp_RSD) %>%
#     pull(Long)
#
# #Turns out PE.36.2._Z and PE.36.2._X are actual duplicates (same data)
# #So PE.36.2._Z is also dropped here
#
# #Peaks to drop
# peaks_to_drop <- tibble(KNN_all) %>%
#     select(where(is.double) & contains("_")) %>%
#     .[!(colnames(.) %in% peaks_to_keep)] %>%
#     colnames() %>%
#     append(.,"PE.36.2._Z")
#
# #Drop duplicates
# KNN_all_nodup <- KNN_all %>%
```

```
#     select(!all_of(peaks_to_drop))
#
# dim(KNN_all_nodup)
# #857 individuals and 257 lipids
#
#
#
# #Continuing with "KNN_all" in order to work with the next piece of code
# KNN_all_yesdup <- KNN_all
# KNN_all <- KNN_all_nodup
#
# #Clean
# rm(tmp_RSD, peaks_to_keep, peaks_to_drop, KNN_all_nodup, KNN_all_yesdup)
```

```
library(gridExtra)
```

```
##
## Vedhæfter pakke: 'gridExtra'
```

```
## Det følgende objekt er maskeret fra 'package:dplyr':
##
##     combine
```

```
#Vector of all lipids to iterate through
Metabolite_names <- KNN_all %>%
    select(where(is.double)) %>%
    colnames()

#Visualize histograms
p <- list()
 for (j in Metabolite_names) {
   p[[paste0(j, " - Raw")]] <- ggplot(KNN_all, aes(x=(KNN_all[,j]))) +
     geom_histogram(aes_string(x=KNN_all[,j]), bins = 30) + xlab(paste0(names(KNN_all[j]), " - Raw")) +
     theme(axis.text=element_text(size=5), axis.title=element_text(size=5),
           legend.text=element_text(size=5), legend.title=element_text(size=5))

   p[[paste0(j, " - log10")]] <- ggplot(KNN_all, aes(x=log10(KNN_all[,j]))) +
     geom_histogram(aes_string(x=log10(KNN_all[,j])), bins = 30) +
       xlab(paste0(names(KNN_all[j]), " - log10")) +
     theme(axis.text=element_text(size=5),
           axis.title=element_text(size=5),
           legend.text=element_text(size=5),
           legend.title=element_text(size=5))}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

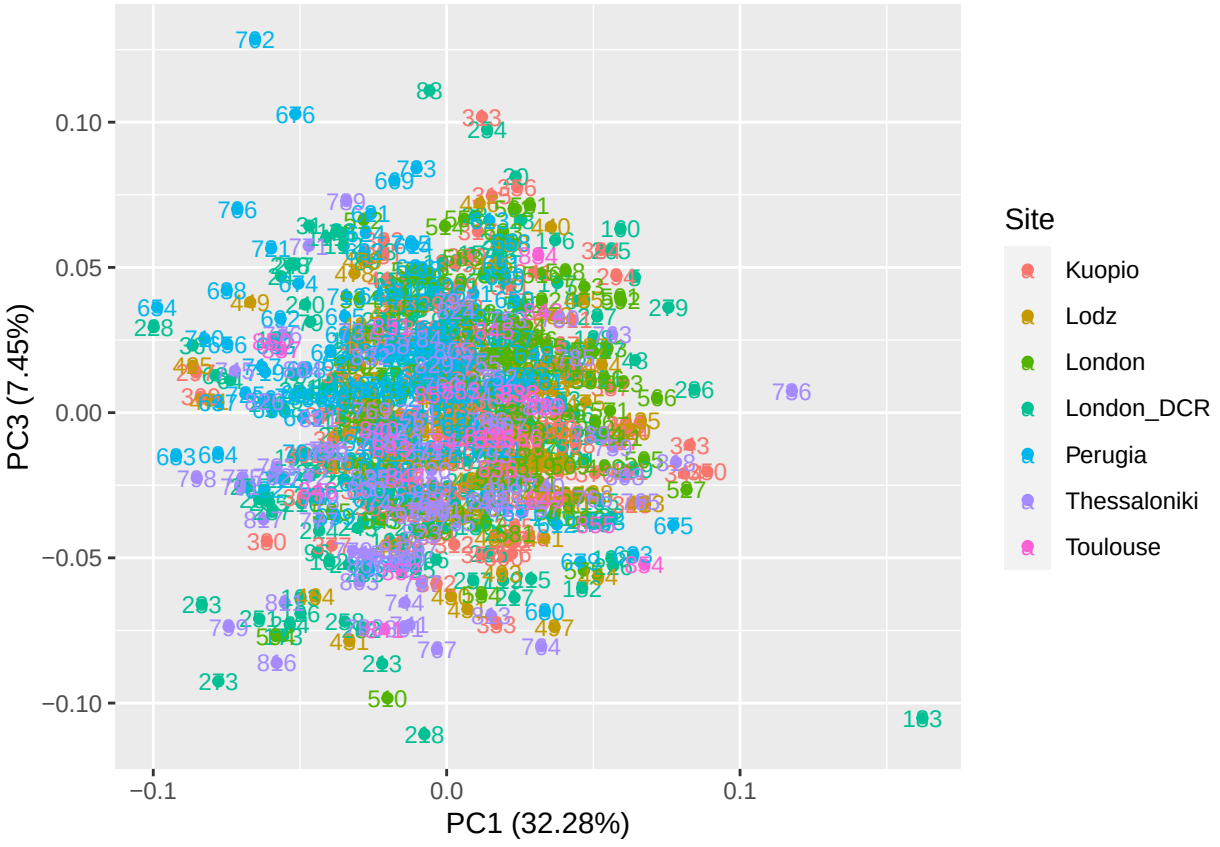
```
# #Write pdf with all figures
# ggsave("Histograms_raw_vs_log10_KNN_all.pdf",
#        marrangeGrob(grobs=p, nrow=2, ncol=2),
#        path = here("figures"))
# dev.off()
```

```
#Log10 Transform
data_lipid_nolog10 <- KNN_all
data_lipid <- tibble(KNN_all) %>%
    mutate(across(where(is.double), ~ log10(.)))

#Clean
rm(j, p, Metabolite_names, KNN_all)
```

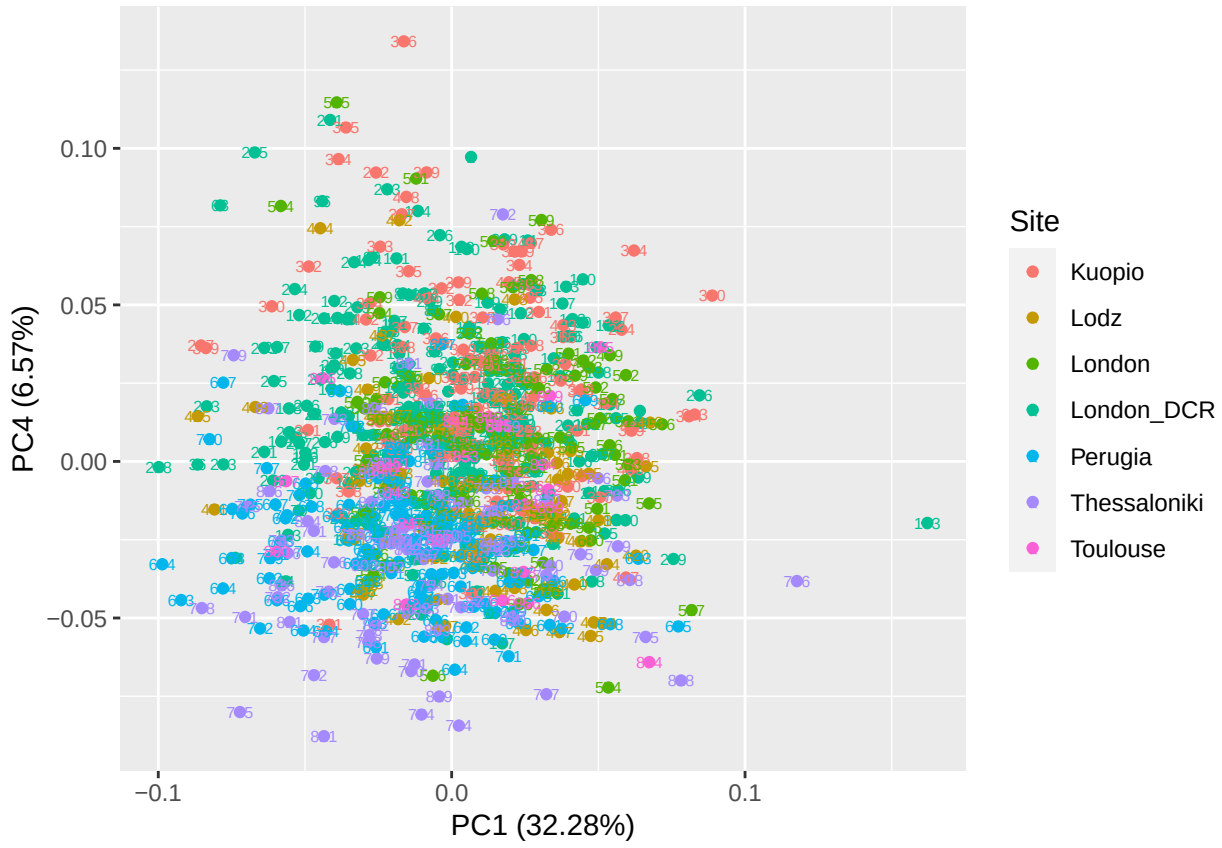
```
library(ggfortify)
```

```
## Warning: pakke 'ggfortify' blev bygget under R version 4.3.1
```

```
#check for outliers using a combination of PCA and kmeans clustering
autoplot(pca_all<-prcomp(data_lipid[, 9:ncol(data_lipid)], center=TRUE, scale.=TRUE), data=data_lipid,c
```




```
autoplot(pca_all<-prcomp(data_lipid[, 9:ncol(data_lipid)], center=TRUE, scale.=TRUE), data=data_lipid,c
```




```
#there seem to be few outliers- particularly
# DCR00395_2 DCR00380_3 THSADC002_1 DCR00041_2  LNDADC_039_2 KPOADC011_2 KPOADC005_1 LNDADC002_1 LNDCTL

#investigate further with k-means clustering
kmeans.result <- kmeans(data_lipid[, 9:ncol(data_lipid)], centers=3)

# "centers" is a data frame of 3 centers but the length of iris dataset so we can canlculate distance d
centers <- kmeans.result$centers[kmeans.result$cluster, ]

#calculate distance
distances <- sqrt(rowSums((data_lipid[, 9:ncol(data_lipid)] - centers)^2))
outliers_ids<-cbind(  distances,data_lipid)
outliers_ids2<-cbind(kmeans.result$cluster,   outliers_ids)

outliers <-outliers_ids2[order(outliers_ids2$distances),]

#show ID of observations with the largest distance to the clusters
tibble(outliers) %>% select(distances, ID_SAD_VISIT) %>% arrange(-distances)
```

```
## # A tibble: 857 x 2
##    distances ID_SAD_VISIT
##        <dbl> <chr>
##  1      8.85 DCR00395_2
##  2      6.25 THSADC002_1
##  3      6.24 LNDADC039_2
##  4      6.15 PRGCTL055_1
```

```
##  5      6.04 THSCTL002_2
##  6      5.98 DCR00041_2
##  7      5.72 PRGADC055_1
##  8      5.72 LDZADC007_2
##  9      5.66 KPOADC011_2
## 10      5.57 DCR00638_1
## # i 847 more rows
```

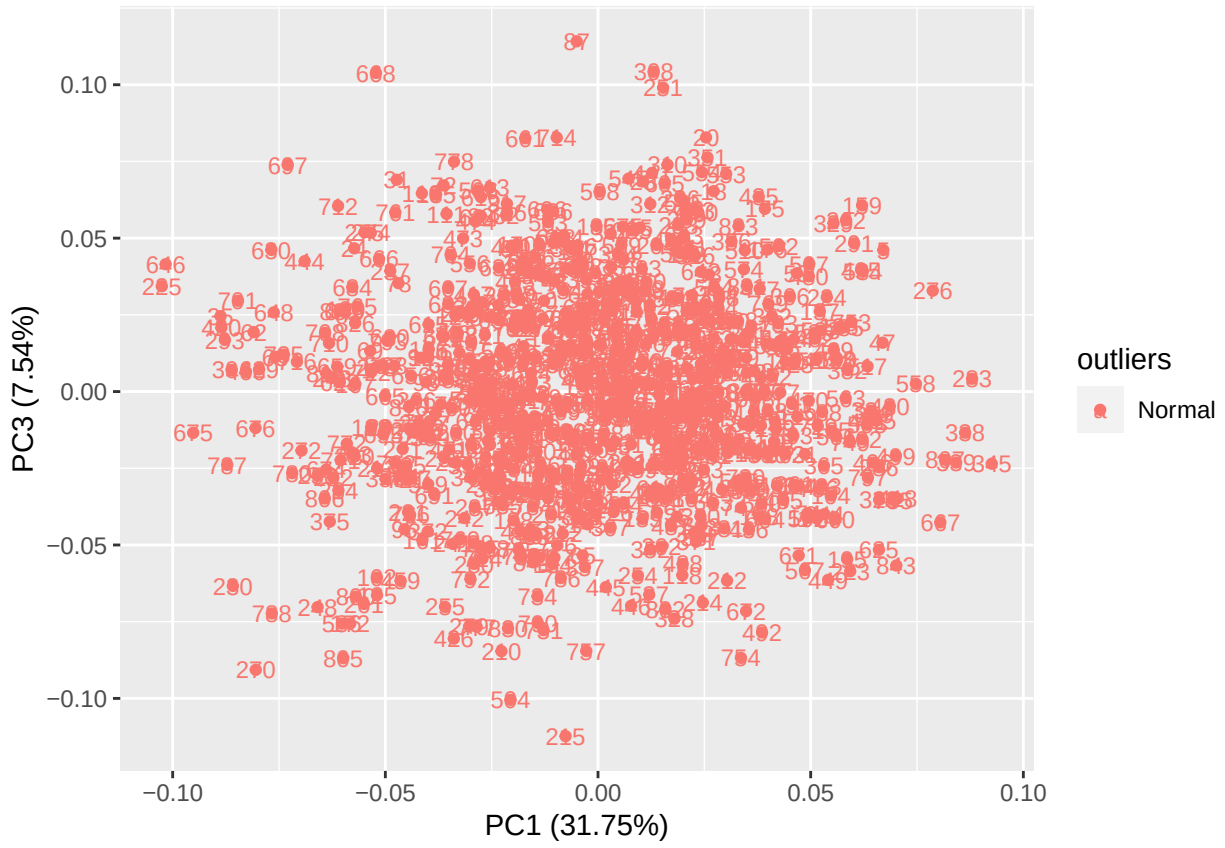
```
#all outliers with distances over 6
outliers[outliers$distances > 6, "ID_SAD_VISIT"]
```

```
## [1] "THSCTL002_2" "PRGCTL055_1" "LNDADC039_2" "THSADC002_1" "DCR00395_2"
```

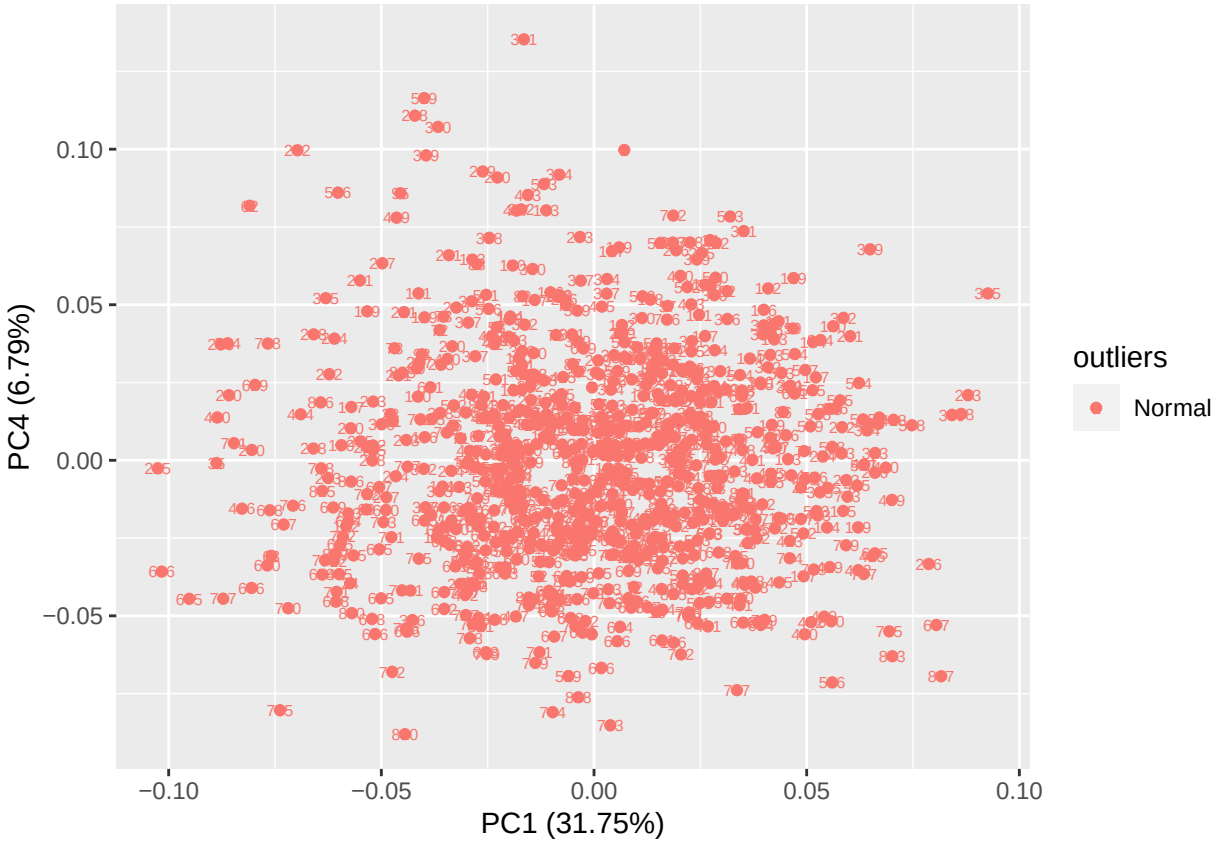
```
#"THSCTL002_2" "PRGCTL055_1" "LNDADC039_2" "THSADC002_1" "DCR00395_2"

#Non log10 distance over 40
#"KPOADC005_1" "LNDADC002_1" "PRGCTL031_1" "DCR00368_1"  "KPOMCI031_1"
#"DCR00814_1"  "PRGADC007_1" "LNDADC039_2" "DCR00395_2"

#Outlier observations, known outliers from protocol (line 265) added
tmp_outliers <- append(outliers[outliers$distances > 6, "ID_SAD_VISIT"],
       c("DCR00380_3",
         "DCR00041_2",
         "KPOADC011_2",
         "KPOADC005_1",
         "LNDADC002_1",
         "LNDCTL017_2"))

#Removing outliers
data_lipid <- data_lipid %>%
    filter(!ID_SAD_VISIT %in% tmp_outliers)
dim(data_lipid)
```

```
## [1] 846 276
```

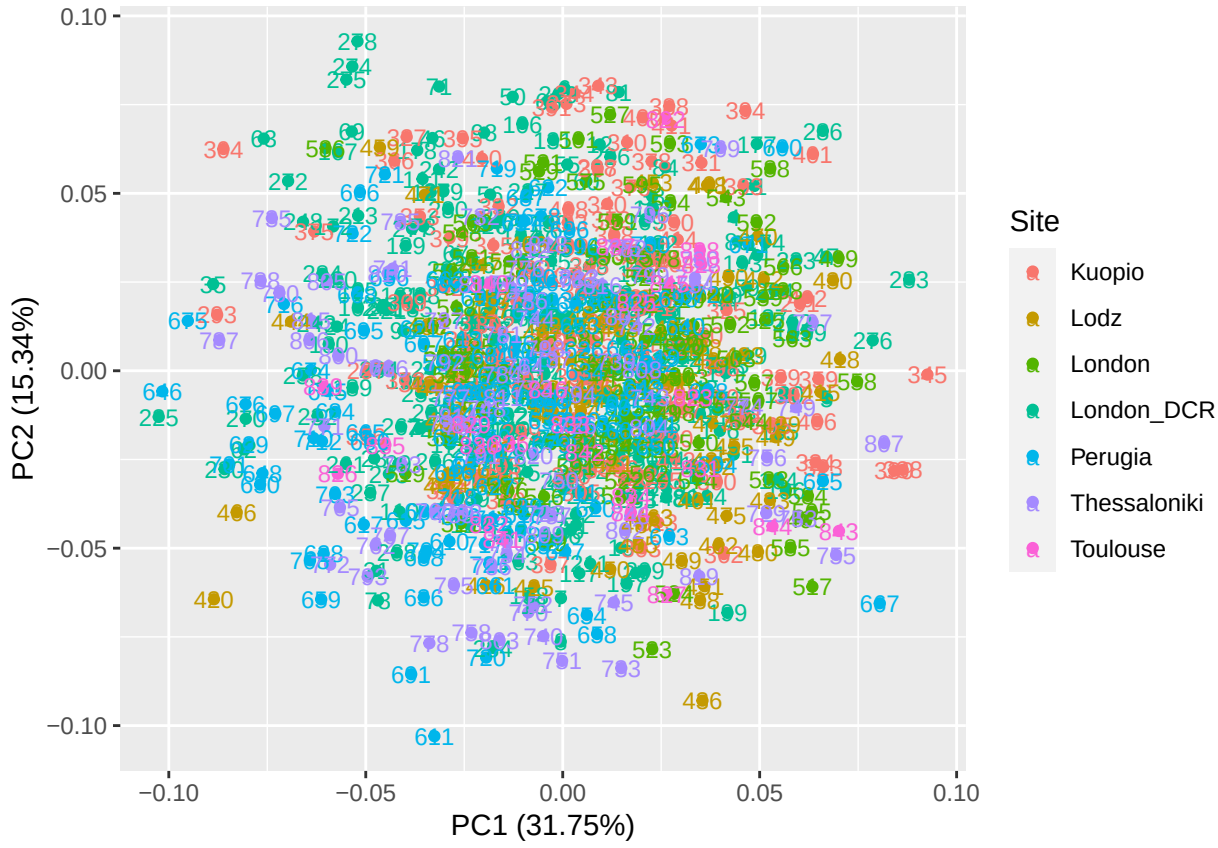
```
#846 276

#temporary vector of outliers base on distance
tmp_outliers_2 <- outliers[outliers$distances > 6, "ID_SAD_VISIT"]

#make temporary dataframe with outliers for pca visualization
tmp_outliers_3 <- data_lipid %>%
    mutate(outliers = if_else(ID_SAD_VISIT %in% tmp_outliers,
                              "Outlier", "Normal")) %>%
    mutate(outliers = if_else(ID_SAD_VISIT %in% tmp_outliers_2,
                              "Outlier_distance", outliers)) %>%
    relocate(outliers, .after = ID_SAD_VISIT)

#plot PCA with outliers shown
autoplot(pca_all <- prcomp(tmp_outliers_3[, 10:ncol(tmp_outliers_3)],
                           center=TRUE, scale.=TRUE),
         data = tmp_outliers_3,
         colour="outliers", x=1, y=3, label=TRUE, label.size=3)
```




```
autoplot(pca_all<-prcomp(tmp_outliers_3[, 10:ncol(tmp_outliers_3)],
                         center=TRUE, scale.=TRUE),
         data=tmp_outliers_3,
         colour="outliers",x=1,y=4, label=TRUE, label.size=2)
```




```
#plot PCA after removal of outliers
autoplot(pca_all<-prcomp(data_lipid[, 9:ncol(data_lipid)], center=TRUE, scale.=TRUE), data=data_lipid,c
```



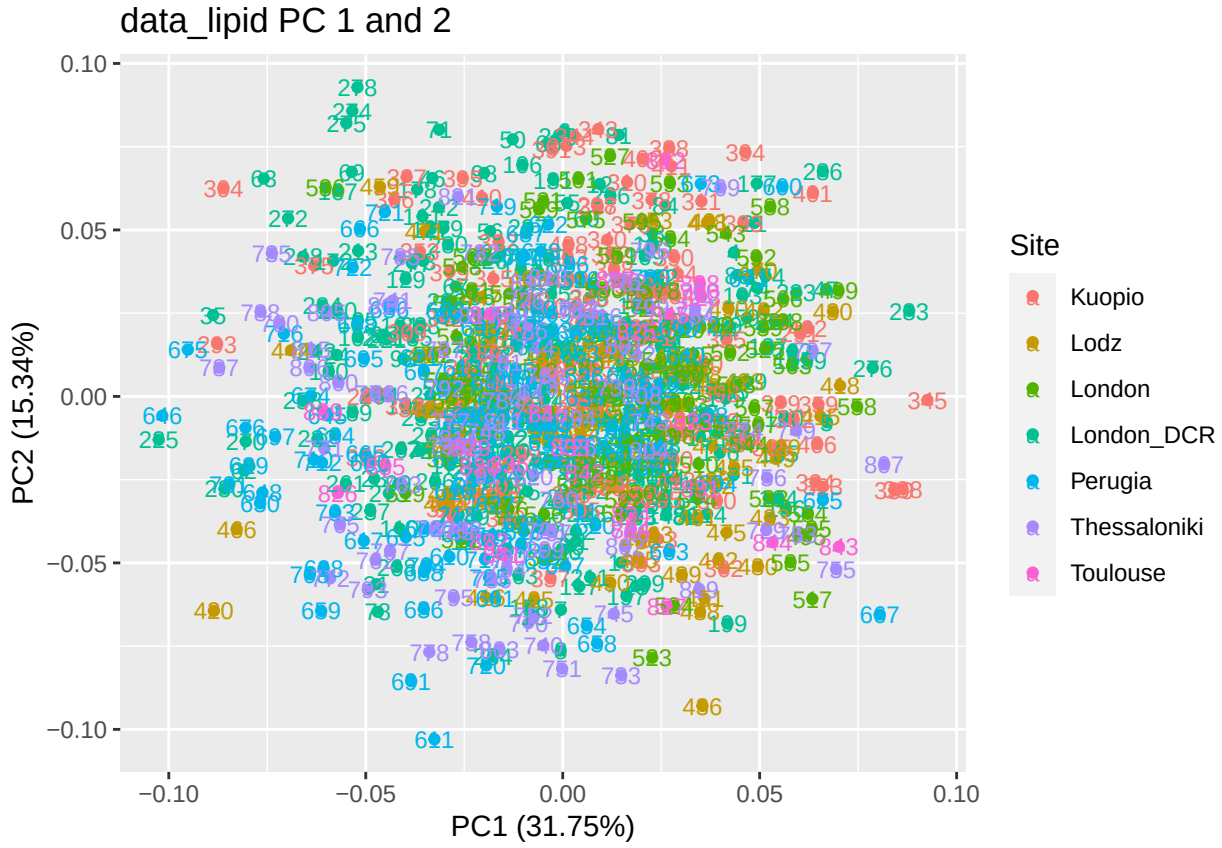

```
#Clean
rm(centers, kmeans.result, pca_all, distances, tmp_outliers, tmp_outliers_2, tmp_outliers_3, outliers, c
```

```
library(ggfortify)
library(DescTools)
```

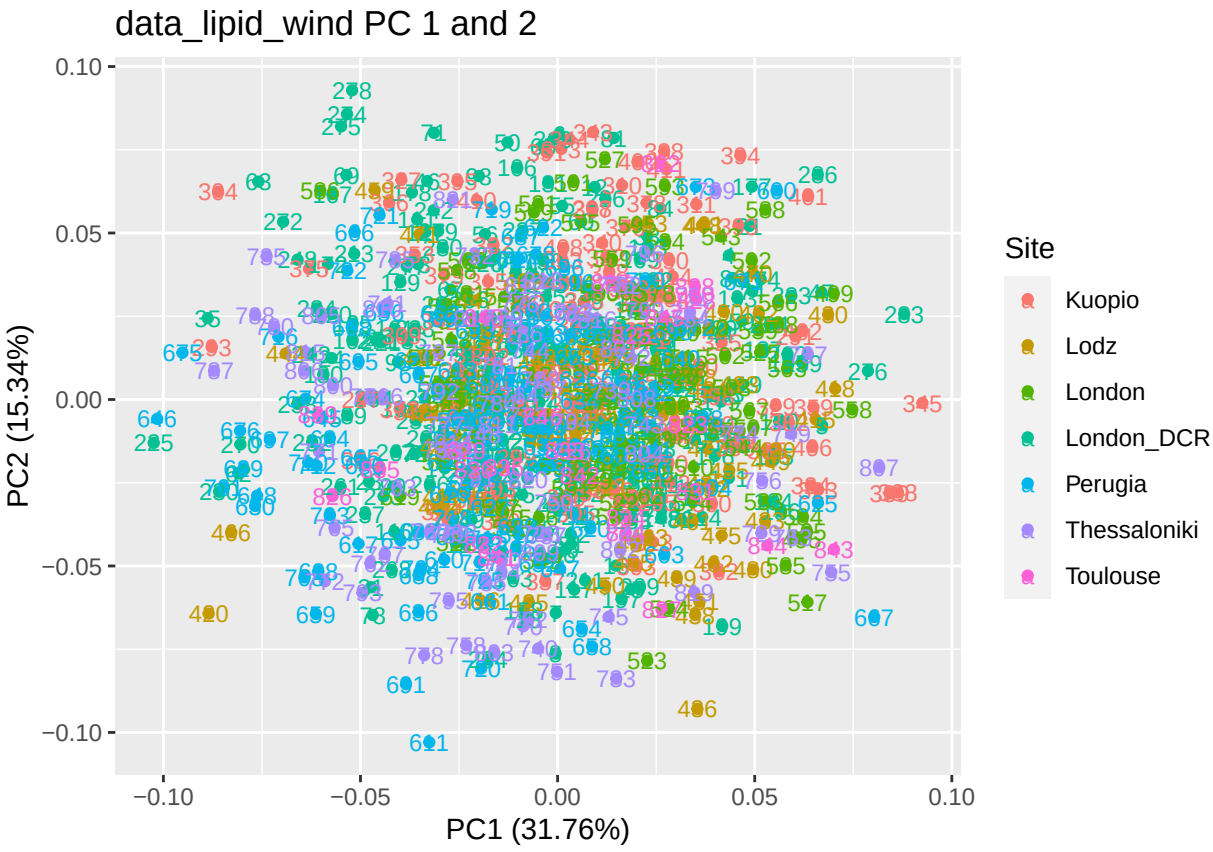
```
## Warning: pakke 'DescTools' blev bygget under R version 4.3.1
```

```
#Define outliers as anything outside the 0.01%-99.99% quantiles,
#equivalent of  4xSD away from the mean.
#Alternatively use 0.3%-99.7% equivalent of 3xSD from the mean.
tmp_outlier_range <- c(0.0001, 0.9999)

#Windsorizing outliers
data_lipid_wind <- tibble(data_lipid) %>%
    mutate(across(where(is.double),
                  ~ Winsorize(., probs = tmp_outlier_range))) %>%
    data.frame()


# #Truncating values more than 3 sd away from the median for each lipid
# data_lipid_trunc <- tibble(data_lipid) %>%
#     mutate(across(where(is.double),
#                   ~ ifelse(. > quantile(., probs = tmp_outlier_range)[[2]],
#                            quantile(., probs = tmp_outlier_range)[[2]], .))) %>%
```

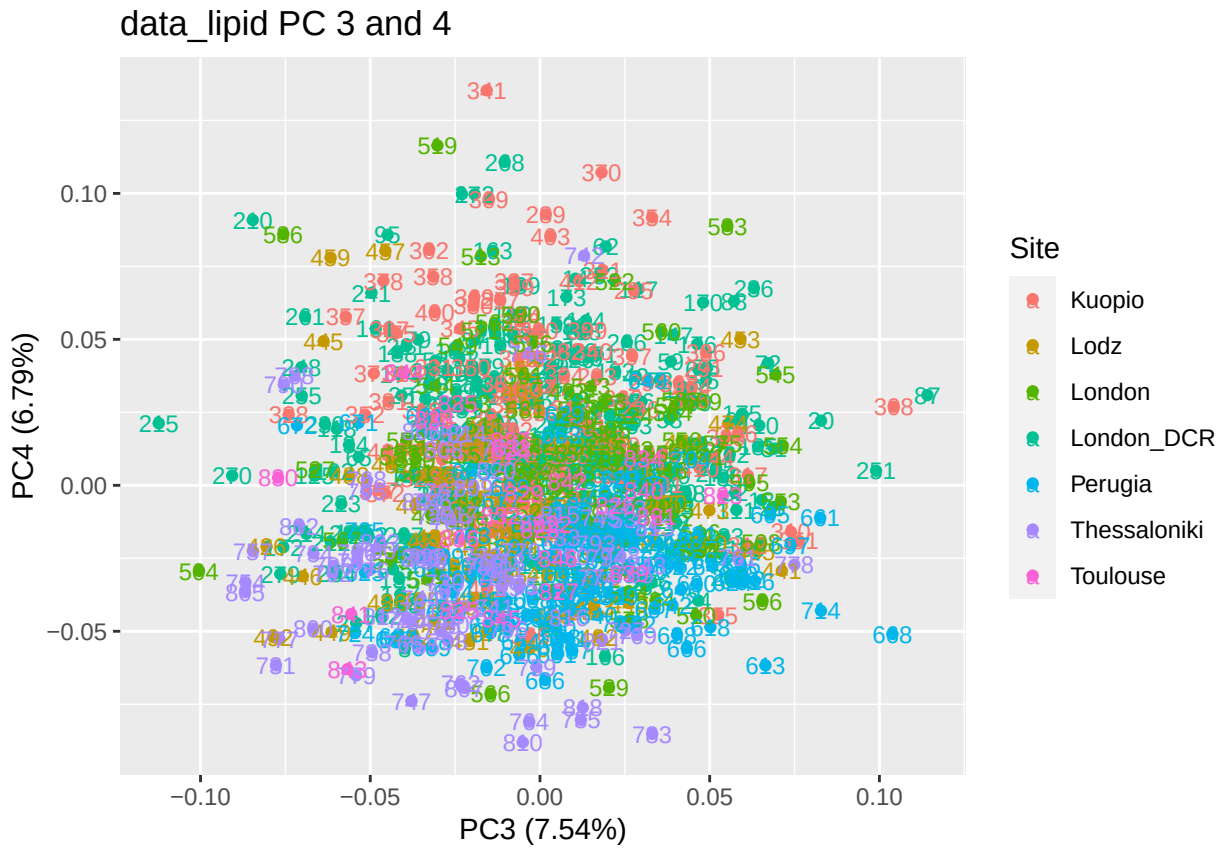
```
#     mutate(across(where(is.double),
#                   ~ ifelse(. < quantile(., probs = tmp_outlier_range)[[1]],
#                            quantile(., probs = tmp_outlier_range)[[1]], .))) %>%
#     data.frame()
#
# #Truncation and Windosorize are the same
# data_lipid_trunc$PC.31.1. == data_lipid_wind$PC.31.1.


#Visualize difference with PCA
#Principal component 1 and 2
autoplot(pca_all <- prcomp(data_lipid[, 9:ncol(data_lipid)],
                           center=TRUE, scale.=TRUE),
         data=data_lipid, colour="Site",
         x=1, y=2, label=TRUE, label.size=3) +
    ggtitle("data_lipid PC 1 and 2")
```



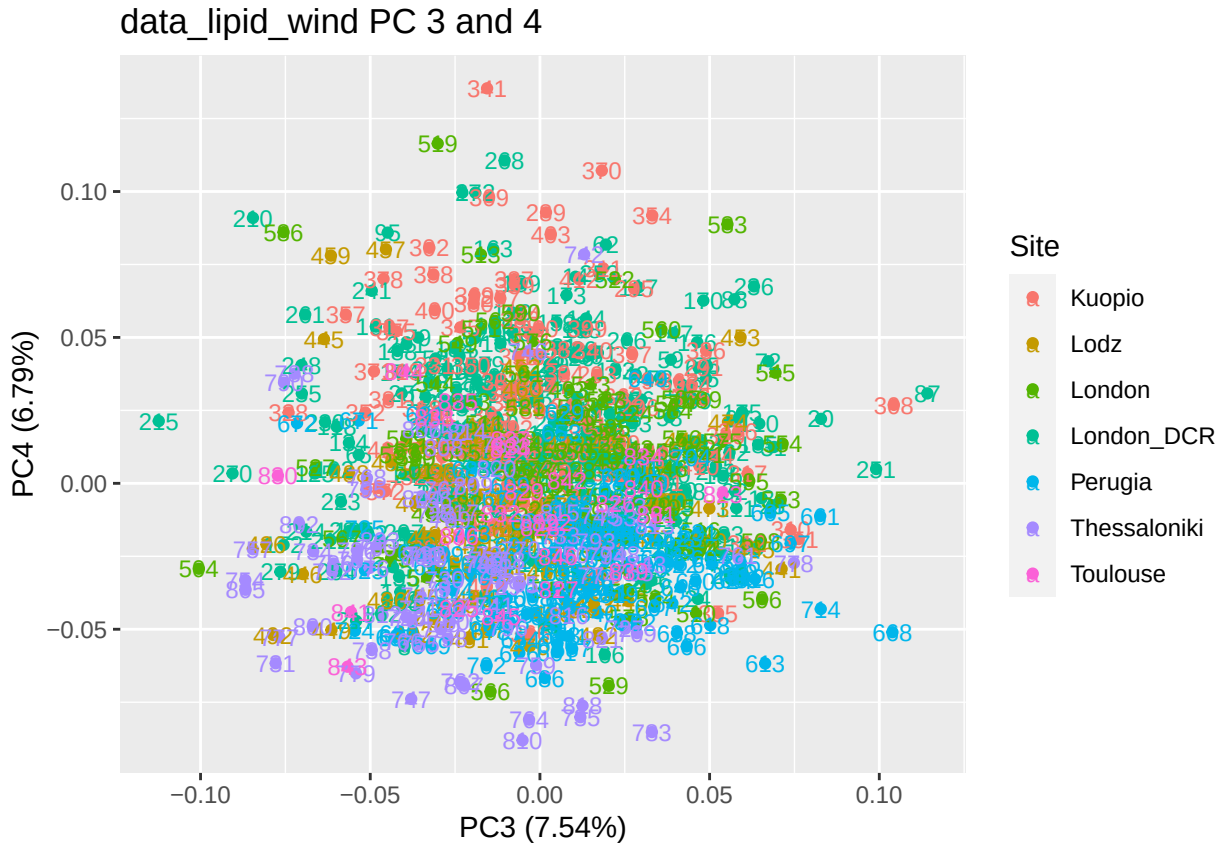

```
autoplot(pca_all <- prcomp(data_lipid_wind[, 9:ncol(data_lipid_wind)],
                           center=TRUE, scale.=TRUE),
         data=data_lipid_wind, colour="Site",
         x=1, y=2, label=TRUE, label.size=3) +
    ggtitle("data_lipid_wind PC 1 and 2")
```




```
#Principal component 3 and 4
autoplot(pca_all <- prcomp(data_lipid[, 9:ncol(data_lipid)],
                           center=TRUE, scale.=TRUE),
         data=data_lipid, colour="Site",
         x=3, y=4, label=TRUE, label.size=3)+
    ggtitle("data_lipid PC 3 and 4")
```




```
autoplot(pca_all <- prcomp(data_lipid_wind[, 9:ncol(data_lipid_wind)],
                           center=TRUE, scale.=TRUE),
         data=data_lipid_wind, colour="Site",
         x=3, y=4, label=TRUE, label.size=3)+
    ggtitle("data_lipid_wind PC 3 and 4")
```




```
#Truncation of outlier lipid levels reduced impact of outlier individuals

# data_lipid %>%
#       ggplot(aes(x = Cer.d42.1._A)) +
#       geom_histogram()

rm(tmp_outlier_range, pca_all)
```

```
library(gridExtra)

#Vector of all lipids to iterate through
Metabolite_names <- data_lipid %>%
    select(where(is.double)) %>%
    colnames()

data_lipid <- data.frame(data_lipid)

#Visualize histograms raw vs. winsorized
p <- list()
for (j in Metabolite_names) {
   p[[paste0(j, " - Raw")]] <- ggplot(data_lipid, aes(x=data_lipid[,j])) +
     geom_histogram(aes_string(x=data_lipid[,j]), bins = 30) +
       xlab(paste0(names(data_lipid[j]), " - Raw")) +
     theme(axis.text=element_text(size=5),
           axis.title=element_text(size=5),
           legend.text=element_text(size=5),
```

```
          legend.title=element_text(size=5))

   p[[paste0(j, " - Windsorized")]] <- ggplot(data_lipid_wind, aes(x=data_lipid_wind[,j])) +
     geom_histogram(aes_string(x=data_lipid_wind[,j]), bins = 30) +
       xlab(paste0(names(data_lipid_wind[j]), " - Windsorized")) +
     theme(axis.text=element_text(size=5),
           axis.title=element_text(size=5),
           legend.text=element_text(size=5),
           legend.title=element_text(size=5))}

# #Write pdf with all figures
# ggsave("Histograms_raw_vs_Windsorized.pdf",
#        marrangeGrob(grobs=p, nrow=2, ncol=2),
#        path = here("figures"))
# dev.off()

#Apply windsorizing
data_lipid_nowind <- data_lipid
data_lipid <- data_lipid_wind

#Clean
rm(j, p, Metabolite_names, data_lipid_wind)
```

```
library(readxl)
```

```
## Warning: pakke 'readxl' blev bygget under R version 4.3.1
```

```
#Read in clinical data
data_clinical <- read_xlsx(here("data-raw/basic_data_demo.xlsx"))

#Read in running order data (LCMS instrument)
order <- read_xlsx(here("data-raw/Running_order.xlsx"))


#Merge data
data <- tibble(data_clinical) %>%
    inner_join(x = ., y = tibble(order),
               by = c("entity_id" = "ID_SAD_VISIT")) %>%
    inner_join(x = data_lipid, y = .,
               by = c("ID_SAD_VISIT" = "entity_id"))

#Note there are 2 individuals with no demo data - we can retain them but even if they have sex info (fr


#Non unique ID_SAD_VISIT introduced in merging
data$ID_SAD_VISIT[duplicated(data$ID_SAD_VISIT)]
```

```
## [1] "LDZCTL018_2" "PRGCTL025_2" "PRGCTL051_1"
```

```
data[data$ID_SAD_VISIT %in% data$ID_SAD_VISIT[duplicated(data$ID_SAD_VISIT)],]$Label
```

```
## [1] 634 634 508 508 847 847
```

```r
#These are just full duplicates and one of each can be droped
data <- data[!duplicated(data$ID_SAD_VISIT),]


#Reorganize
data <- data %>%
    relocate(entity:NEW_ORDER, .after = Site) %>%
    select(-c(Sample.ID, Sample.ID_EU, Sample.ID_SAD, entity, id)) %>%
    relocate(c(ID_SAD_VISIT, Site, Visit, Date, NEW_ORDER),
             .before = Label) %>%
    rename(Order = NEW_ORDER) %>%
    rename(ID = ID_SAD_VISIT)


#Fix lipid names
#First .
colnames(data) <- sub("\\.", "(", colnames(data))
#Second .
colnames(data) <- sub("\\.", ":", colnames(data))
#Third .
colnames(data) <- sub("\\.", ")", colnames(data))
#4
colnames(data) <- sub("\\.", ")", colnames(data))
#5
colnames(data) <- sub("\\.", "/", colnames(data))
#6
colnames(data) <- sub("\\.", "(", colnames(data))
#7
colnames(data) <- sub("\\.", "-", colnames(data))
#8
colnames(data) <- sub("\\.", ":", colnames(data))
#9
colnames(data) <- sub("\\.", ")", colnames(data))

#Other name fixes
colnames(data) <- sub("O:", "O-", colnames(data))
colnames(data) <- sub("P:", "P-", colnames(data))
colnames(data)[grepl("-", colnames(data))] <- sub(")", ":", colnames(data)[grepl("-", colnames(data))])
#colnames(data) <- sub(")_.", ")", colnames(data))

#specific lipid name fix
colnames(data)[colnames(data) %in% "PE(O-36:2)/(PE-P-36)1."] <- "PE(O-36:2)/PE(P-36:1)"

#Are all column names unique
length(unique(colnames(data))) == length(colnames(data))
```

```
## [1] TRUE
```

```r
#Fix dates
data <- data %>%
    mutate(Date = if_else(nchar(Date) == 5,
                          format(as.Date(x = as.integer(Date),
                                         origin = "1899-12-30"),
```

```
                                c("%d/%m/%Y")), Date)) %>%
    mutate(DOB = if_else(nchar(DOB) == 5 |
                         nchar(DOB) == 4,
                          format(as.Date(x = as.integer(DOB),
                                         origin = "1899-12-30"),
                                 c("%d/%m/%Y")), DOB))
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `Date = if_else(...)`.
## Caused by warning in `as.Date()`:
## ! NAs introduced by coercion
```

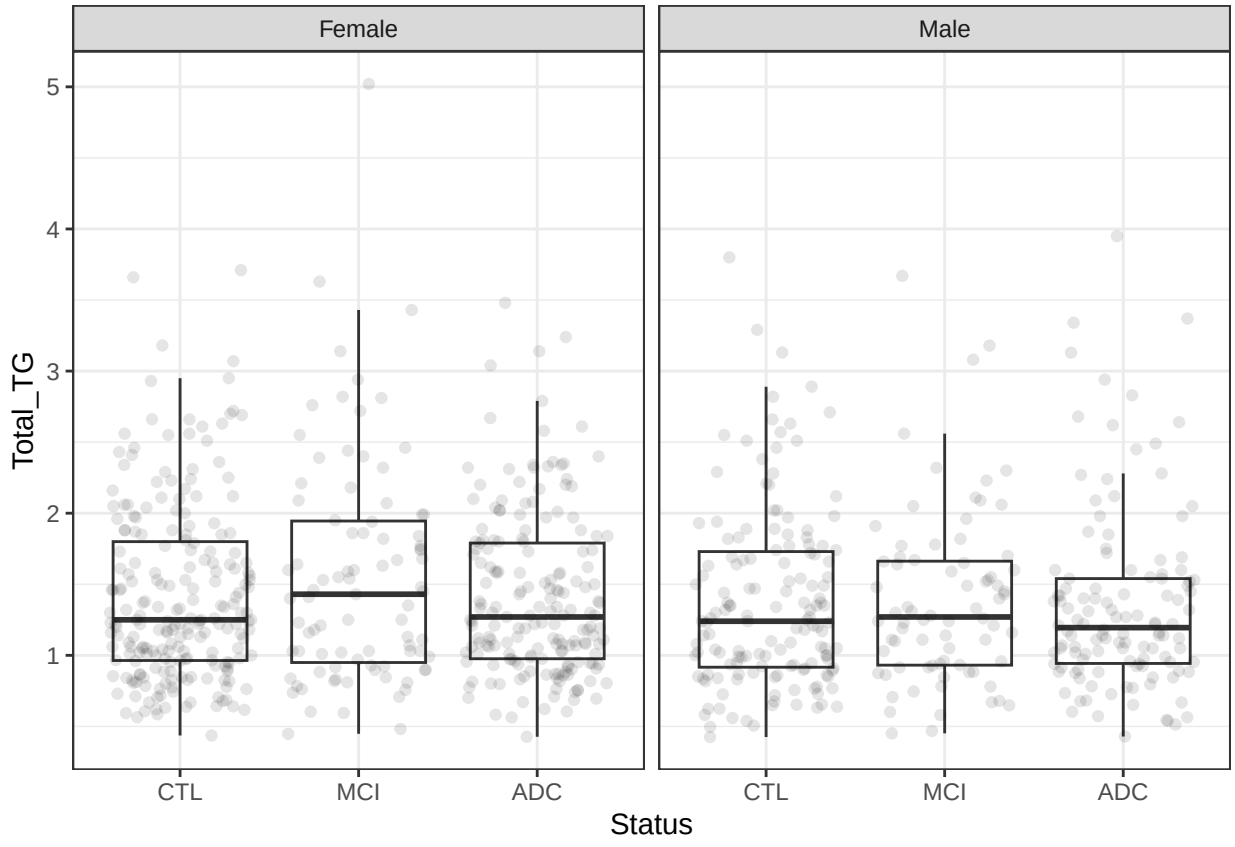
```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `DOB = if_else(...)`.
## Caused by warning in `as.Date()`:
## ! NAs introduced by coercion
```

```
#NMR data
data_NMR <- read.csv(here("data-raw/Metabolics_QC.csv"))

data_NMR <- data_NMR %>%
    tibble() %>%
    select(entity_id, Total.C, Total.TG, HDL.C, LDL.C, ApoB) %>%
    rename(ID = entity_id)

colnames(data_NMR) <- gsub("\\.", "_", colnames(data_NMR))

# #Not used due to large amount of missingness
# #Amyloid and tau
# data_ATN <- read.csv(here("data-raw/addneuromed_blood_biomarkers_200416_Petra.csv"))
#
# data_ATN <- data_ATN %>%
#     tibble() %>%
#     select(!Short.ID:Date.of.Visit) %>%
#     rename(ID = Long.ID) %>%
#     select(-Sadman.ID)
#
# colnames(data_ATN) <- gsub("\\.", "_", colnames(data_ATN))

# #Not used due to large amount of missingness
# #Waist circumference and blood pressure
# data_phys <- read.csv(here("data-raw/PhysicalMeasurement.csv"))
#
# data_phys <- data_phys %>%
#     tibble() %>%
#     select(entity_id, Waist_Circumference, Systolic, Diastolic)  %>%
#     rename(ID = entity_id)


#MMSE
data_MMSE <- read.csv(here("data-raw/MMSE_data.csv"))

data_MMSE <- data_MMSE %>%
```

```
    tibble() %>%
    select(entity_id, MMSE_Total)  %>%
    rename(ID = entity_id)

#Merge
data <- data %>%
    left_join(., data_NMR, by = "ID") %>%
    #left_join(., data_ATN, by = "ID") %>%
    #left_join(., data_phys, by = "ID") %>%
    left_join(., data_MMSE, by = "ID") %>%
    relocate(Total_C:MMSE_Total, .after = Disease_Duration)

#visualize new variables
data %>%
    mutate(Status = factor(Status, levels = c("CTL", "MCI", "ADC"))) %>%
    ggplot(aes(y = Total_TG, x = Status)) +
    geom_boxplot(outlier.shape = NA)+
    geom_jitter(alpha = 0.1)+
    facet_grid(.~ Sex)+
    theme_bw()
```

```
## Warning: Removed 13 rows containing non-finite values (`stat_boxplot()`).
```

```
## Warning: Removed 13 rows containing missing values (`geom_point()`).
```



```r
rm(data_NMR, data_ATN, data_MMSE, data_phys)
```

```
## Warning in rm(data_NMR, data_ATN, data_MMSE, data_phys): objekt 'data_ATN' blev
## ikke fundet
```

```
## Warning in rm(data_NMR, data_ATN, data_MMSE, data_phys): objekt 'data_phys'
## blev ikke fundet
```

```r
library(vroom)
```

```
## Warning: pakke 'vroom' blev bygget under R version 4.3.1
```

```
##
## Vedhæfter pakke: 'vroom'
```

```
## De følgende objekter er maskerede fra 'package:readr':
##
##     as.col_spec, col_character, col_date, col_datetime, col_double,
##     col_factor, col_guess, col_integer, col_logical, col_number,
##     col_skip, col_time, cols, cols_condense, cols_only, date_names,
##     date_names_lang, date_names_langs, default_locale, fwf_cols,
##     fwf_empty, fwf_positions, fwf_widths, locale, output_column,
##     problems, spec
```

```r
# #Export preprocessed data to data folder
# vroom_write(data, here("data/ANM_Lipid_Preprocessed_v4.csv"))
```