Computational Data Analysis of Atypical K6-linked Ubiquitylation Marks Toxic RNA-protein Crosslinks Induced by Formaldehyde

Christian J. Blum

29/Feb/2023

Packages and functions

```
#load all necessary packages ----
library(DT)
library(plotly)
library(htmlwidgets)
library(openxlsx)
library(ggpubr)
library(ggrepel)
library(limma)
library(ggvenn)
library(ggrepel)
library(GGally)
library(biomaRt)
library(matrixTests)
library(ggnetwork)
library(network)
library(igraph)
library(clusterProfiler)
# define organism against the search/enrichment will be performed
org <- "org.Hs.eg.db"
library(org, character.only = TRUE)
library(RColorBrewer)
library(ReactomePA)
library(enrichplot)
library(ggraph)
library(tidygraph)
library(tidyverse)


### functions ----
`%!in%` = Negate(`%in%`)
#calculate significance with limma; define limma_function
limma_function = function(ratios, max_q = 0.05, min_count = 2, prefix = ''){
  result = data.frame(
    count = ncol(ratios) - rowSums(is.na(ratios)),
    mean = rowMeans(ratios, na.rm = T)
  )
```

```
  #fit = statistical testing using eBayes and
  #lmFit to generate p and q (=FDR) values
  fit = as.data.frame(eBayes(lmFit(ratios[result$count >= min_count, ])))

  #adds the according p-value if the count value is more
  #or equal the preset min_count
  result[result$count >= min_count, 'pvalue'] = fit$p.value
  #adds the accoridng q-value if the count value is more or equal the preset
  #min_count; calculates the adjusted p-value
  #with p.adjust using the fdr method
  #(here fdr is the same as BH = Benjamini & Hochberg correction;
  #others can be chosen)
  result[result$count >= min_count, 'qvalue'] = p.adjust(fit$p.value,
                                                         method = 'fdr')
  #up or douwn-regulation of a protein are computed
  #giving a TRUE or FALSE statement
  results = mutate(
    result,
    up = count >= min_count & mean > 0 & qvalue <= max_q,
    down = count >= min_count & mean < 0 & qvalue <= max_q
  )
  names(results) = paste0(prefix, names(results))
  return(results)
}
```

Preprocessing of proteomics data

```
### PREPROCESSING ----
#set working directory
setwd(dir ="working_directory")

#read data of replicate 1-3
pg_gg <- read.delim("GlyGly (K)Sites.txt", stringsAsFactors = FALSE)
raw <- pg_gg

#set working directory to analysis folder
setwd(dir ="working_directory/analysis")

#reduce regular Gene.names strings
pg_gg$Gene.names <- sub(";.*$", "", pg_gg$Gene.names)

#define subset in proteinGroups; filtering for reverse positive, contaminant,
#and diGly-modification localization probability >= 0.95
no_reverse <-pg_gg [,"Reverse"] != "+"
no_contaminant <- pg_gg [,"Potential.contaminant"] != "+"
low_localization_prob <- pg_gg [, "Localization.prob"] >= 0.95

#filter to subset
pg_gg <- subset(pg_gg, no_reverse & no_contaminant & low_localization_prob)
```

```
rm(no_contaminant,no_reverse, low_localization_prob)

#log2 transformation and define ratios
# # log2 transformation of MaxQuant-normalized ratios and Label switch (SILAC)
pg_gg$log2.Treatment1.UT.1 <- log2(pg_gg$Ratio.H.L.normalized.1)
pg_gg$log2.Treatment1.UT.2 <- log2(1/pg_gg$Ratio.M.L.normalized.2)
pg_gg$log2.Treatment1.UT.3 <- log2(pg_gg$Ratio.H.L.normalized.3)


pg_gg$log2.Treatment2.UT.1 <- log2(pg_gg$Ratio.M.L.normalized.1)
pg_gg$log2.Treatment2.UT.2 <- log2(pg_gg$Ratio.H.M.normalized.2)
pg_gg$log2.Treatment2.UT.3 <- log2(pg_gg$Ratio.M.L.normalized.3)


pg_gg$log2.Treatment1.Treatment2.1 <- log2(pg_gg$Ratio.H.M.normalized.1)
pg_gg$log2.Treatment1.Treatment2.2 <- log2(1/pg_gg$Ratio.H.L.normalized.2)
pg_gg$log2.Treatment1.Treatment2.3 <- log2(pg_gg$Ratio.H.M.normalized.3)


#add missing gene names from FASTA header
missing_gene_names <- which(pg_gg$Gene.names == "")
for (i in missing_gene_names){
  #adding gene.names
  pg_gg$Gene.names[i] <- sub(" PE=.*", "", pg_gg$Fasta.headers[i])
  pg_gg$Gene.names[i] <- sub(".*GN=","",pg_gg$Gene.names[i])
  #adding protein names/description
  pg_gg$Protein.names[i] <- sub(" OS=.*", "", pg_gg$Fasta.headers[i])
  pg_gg$Protein.names[i] <- sub(".*HUMAN ","",pg_gg$Protein.names[i])

  if (startsWith(x = pg_gg$Gene.names[i], prefix = ">")){
    pg_gg$Gene.names[i] <- pg_gg$Protein.names[i]
  }
}

# reducing dataframe
pg_gg_filtered <- pg_gg
pg_gg <- pg_gg_filtered[ ,c(5:6, 8:13, 27:29, 32:33, 265, 282:290)]
```

Statistical analysis

```
## LIMMA for normalized Log2- transformed ratios

# run the limma_function defined in Packages_and_functions.R
#define experiment names
Exp_name <- c("log2.Treatment1.UT.1",
              "log2.Treatment1.UT.2",
              "log2.Treatment1.UT.3")
#calculate statistics for diGly-sites that were
```

```
#at least 2 times quantified via MS
hl_statistics = limma_function(pg_gg[ ,Exp_name], min_count = 2)
#add statistical analysis to working df
pg_gg = bind_cols(pg_gg, hl_statistics)
pg_gg$Signature <- paste0(pg_gg$Gene.names, "_",
                          pg_gg$Amino.acid, pg_gg$Position)
#selecting relevant columns and creating output df
output <- pg_gg[ ,c(1:2, 30, 3:17, 24:29)]
#writing output file
write.table(x = output, file = "limma_Treatment1vsUT.txt", sep = "\t",
            row.names = FALSE, col.names = TRUE, quote = FALSE)
```

Volcano-plot

```
# Volcano Plot
# subsetting for diGly sites with at least 2 quantifications
pg_gg <- subset(pg_gg, pg_gg$count >= 2)

#dot color and labeling according to significance cutoffs
#qvalue <= 0.05 & log2(FC) >= 2)
for (i in 1: nrow(pg_gg)){
  if (pg_gg$mean[i] >= 2 & pg_gg$qvalue[i] <= 0.05)
  {pg_gg$threshold[i] <- "#cb181d"
  pg_gg$label[i] <- 1}
  else if (pg_gg$mean[i] >= 1 & pg_gg$mean[i] <= 2 & pg_gg$qvalue[i] <= 0.05)
  {pg_gg$threshold[i] <- "#fc9272"
  pg_gg$label[i] <- 2}
  else if (pg_gg$mean[i] <= -2 & pg_gg$qvalue[i] <= 0.05)
  {pg_gg$threshold[i] <- "#0073C2FF"
  pg_gg$label[i] <- 3}
  else if (pg_gg$mean[i] <= -1 & pg_gg$mean[i] >= -2 & pg_gg$qvalue[i] <= 0.05)
  {pg_gg$threshold[i] <- "#9ecae1"
  pg_gg$label[i] <- 4}
  else {pg_gg$threshold[i] <- "#bdbdbd"
  pg_gg$label[i] <- 0}
}

#Plotting

#plot with ggplot
volcano <-
  ggplot(data=pg_gg,
    aes(x = mean, y = -log10(qvalue),
        text = paste("Protein:", Gene.names, "\n",
                     "GlyGly..K..Probabilities", GlyGly..K..Probabilities,
                     "\n",
                     "fold-change:", ifelse(mean > 0,
                                            round(2^abs(mean),digits = 3),
                                            -round(2^abs(mean), digits = 3)),
```

```
                    "\n",
                    "-log10(q-value):", round(-log10(qvalue), digits = 3),
                    "\n",
                    "Description:", Protein.names,
                    "\n",
                    "K-Position:", Position,
                    "\n",
                    sep = " "))) +
  scale_x_continuous(name="log2(Treatment1 / UT)",
                     expand = c(0,0),
                     limits = c(min(pg_gg$mean)-0.5, max(pg_gg$mean)+0.5),
                     breaks = waiver(),
                     n.breaks = 5)+
  scale_y_continuous(name="-log10(q-value)",
                     expand = c(0,0),
                     limits = c(0, -log10(min(pg_gg$qvalue))+0.5),
                     breaks = waiver(),
                     n.breaks = 8)+
  geom_hline(yintercept=-log10(0.05),
             linetype="dashed",
             color = "darkgrey",
             size = 0.7,
             alpha = 0.7) +
  geom_vline(xintercept=2,
             linetype="dashed",
             color = "darkgrey",
             size = 0.7,
             alpha = 0.7) +
  geom_vline(xintercept=1,
             linetype="dashed",
             color = "lightgrey",
             size = 0.7) +
  geom_vline(xintercept=-2,
             linetype="dashed",
             color = "darkgrey",
             size = 0.7,
             alpha = 0.7) +
  geom_vline(xintercept=-1,
             linetype="dashed",
             color = "lightgrey",
             size = 0.7) +
  geom_point(aes(alpha=1,
                 colour=threshold),
             size = 3) +
  geom_point(data =filter(pg_gg, label == 1),
             shape = 1,
             size = 3,
             colour = "black")+
  geom_point(data =filter(pg_gg, label == 3),
             shape = 1,
             size = 3,
             colour = "black")+
  geom_label_repel(data=filter(pg_gg, label == 1),
```

```
                    aes(label=paste0(Gene.names, " (", Amino.acid,
                                     Position, ")")),
                    na.rm = TRUE,
                    max.overlaps = 500,
                    max.time = 2,
                    size = 4,
                    segment.colour = 'black',
                    segment.alpha = 0.5,
                    # hjust = -0.1,
                    segment.curvature = 0.1,
                    segment.inflect = TRUE,
                    segment.square = FALSE,
                    xlim = c(0, NA)) +
    scale_color_identity()+
    theme(axis.line = element_line(colour = "black", size = 1),
          axis.ticks = element_line(colour = "black", size = 1),
          axis.text = element_text(size = 14, face = "bold",
                                   colour = "black"),
          axis.title = element_text(size = 14, face = "bold",
                                    colour = "black"),
          panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          panel.border = element_rect(linetype = "solid", fill = NA,
                                      colour = "black", size = 1),
          legend.position="none",
          panel.background = element_blank(),) +
    theme(plot.title = element_text(hjust = 0.5))

#creating output pdf file
pdf("working_directory/volcanoplot_Treatment1vsUT.pdf",
    height = 13, width = 14)
volcano
dev.off()
```

GO-enrichment analysis

```
### GO-Enrichment using clusterProfiler 4.0 ----
# load df of Treamtnet1 vs UT with statistical analysis
df <- read.delim("limma_df_Treatment1_UT.txt", stringsAsFactors = F)

# filter for signifcantly upregulated proteins
temp_df <- df
temp_df <- subset(temp_df, temp_df$count >= 2 & temp_df$mean >= log2(2))
temp_df <- temp_df$Gene.names

# load total df as background to compare to
background <- unique(df$Gene.names)
universe <- background
```

```
# biomaRt (ENSEMBL)
listMarts()
ensembl <- useMart("ensembl")
datasets <- listDatasets(ensembl)
  # View(datasets)
ensembl105 <- useDataset(dataset = "hsapiens_gene_ensembl", mart = ensembl)

# ENSEMBL annotation of upregulated proteins
annotation_df <- getBM(attributes = c("entrezgene_id",
                                      "description",
                                      "external_gene_name"),
                       filters = c("external_gene_name"),
                       values = temp_df,
                       mart = ensembl105)

# ENSEMBL annotation of all quantified proteins
annotation_universe <- getBM(attributes = c("entrezgene_id",
                                            "description",
                                            "external_gene_name"),
                             filters = c("external_gene_name"),
                             values = universe,
                             mart = ensembl105)


############################
### GO term annotation ###
############################

# join both datasets, use the first ENSEMBL name for double annotated genes
#for high confidence hits
annotated_df <- temp_df
temp <- as.data.frame(matrix(nrow = length(temp_df), ncol=ncol(annotation_df)))
colnames(temp) <- colnames(annotation_df)
annotated_df <- cbind(annotated_df, temp)
colnames(annotated_df) <- c("Gene.names", colnames(temp))
annotated_df <- subset(annotated_df,
                       annotated_df$Gene.names
                       %in%
                         annotation_df$external_gene_name)

for (i in 1:nrow(annotated_df)){
  temp <- subset(annotation_df,
                 annotation_df$external_gene_name == annotated_df$Gene.names[i])
  if (nrow(temp) > 1){
    annotated_df$entrezgene_id[i] <- temp$entrezgene_id[1]
    annotated_df$description[i] <- temp$description[1]
    annotated_df$external_gene_name[i] <- temp$external_gene_name[1]
  }else{
    annotated_df$entrezgene_id[i] <- temp$entrezgene_id[1]
    annotated_df$description[i] <- temp$description[1]
    annotated_df$external_gene_name[i] <- temp$external_gene_name[1]
  }
}
```

```
# for all proteins
annotated_df_uni <- universe
temp <- as.data.frame(matrix(nrow = length(universe),
                             ncol=ncol(annotation_universe)))
colnames(temp) <- colnames(annotation_universe)
annotated_df_uni <- cbind(annotated_df_uni, temp)
colnames(annotated_df_uni) <- c("Gene.names", colnames(temp))
annotated_df_uni <- subset(annotated_df_uni,
                           annotated_df_uni$Gene.names
                           %in%
                             annotation_universe$external_gene_name)

for (i in 1:nrow(annotated_df_uni)){
  temp <- subset(annotation_universe,
                 annotation_universe$external_gene_name ==
                   annotated_df_uni$Gene.names[i])
  if (nrow(temp) > 1){
    annotated_df_uni$entrezgene_id[i] <- temp$entrezgene_id[1]
    annotated_df_uni$description[i] <- temp$description[1]
    annotated_df_uni$external_gene_name[i] <- temp$external_gene_name[1]
  }else{
    annotated_df_uni$entrezgene_id[i] <- temp$entrezgene_id[1]
    annotated_df_uni$description[i] <- temp$description[1]
    annotated_df_uni$external_gene_name[i] <- temp$external_gene_name[1]
  }
}

# transform entrez_gene_id column into a vector and remove NA values
#for high confidence hits
ent_gene <- annotated_df$entrezgene_id
ent_gene <- ent_gene[which(!is.na(ent_gene))]
ent_gene <- as.character(ent_gene)
#for universe
ent_gene_uni <- annotated_df_uni$entrezgene_id
ent_gene_uni <- ent_gene_uni[which(!is.na(ent_gene_uni))]
ent_gene_uni <- as.character(ent_gene_uni)

# clusterProfiler GO enrichment - Molecular Function
ego_MF <- enrichGO(gene = ent_gene, OrgDb = org,
                   ont = "MF", universe = ent_gene_uni,
                   pvalueCutoff = 0.05, pAdjustMethod = "BH", readable = TRUE)
write.table(x = ego_MF@results, file = "ego_MF.txt", sep = "\t")

# clusterProfiler GO enrichment - Biological Processes
ego_BP <- enrichGO(gene = ent_gene, OrgDb = org,
                   ont = "BP", universe = ent_gene_uni,
                   pvalueCutoff = 0.05, pAdjustMethod = "BH", readable = TRUE)
write.table(x = ego_BP@results, file = "ego_BP.txt", sep = "\t")

# clusterProfiler GO enrichment - Cellular Compartments
ego_CC <- enrichGO(gene = ent_gene, OrgDb = org,
                   ont = "CC", universe = ent_gene_uni,
                   pvalueCutoff = 0.05, pAdjustMethod = "BH", readable = TRUE)
```

```
write.table(x = ego_CC@results, file = "ego_CC.txt", sep = "\t")


#########################################################################
# reading in data to generate lollipop plots
# filter data for Benjamini-Hochberg adjusted p-value <= 0.05
MF <- read.delim("ego_MF.txt", stringsAsFactors = FALSE)
MF <- subset(MF, p.adjust <= 0.05)
MF$origin <- "MF"
BP <- read.delim("ego_BP.txt", stringsAsFactors = FALSE)
BP <- subset(BP, p.adjust <= 0.05)
BP$origin <- "BP"
CC <- read.delim("ego_CC.txt", stringsAsFactors = FALSE)
CC <- subset(CC, p.adjust <= 0.05)
CC$origin <- "CC"

# assemble data in one common df
GO_data <- rbind(MF, BP, CC)

# calculate fold enrichment score
for (i in 1: nrow(GO_data)){
  GO_data$total[i] <- as.numeric(str_split(GO_data$GeneRatio[i],
                                           pattern = "/")[[1]][2])
  GO_data$total_bg[i] <- as.numeric(str_split(GO_data$BgRatio[i],
                                              pattern = "/")[[1]][2])
  GO_data$count_bg[i] <- as.numeric(str_split(GO_data$BgRatio[i],
                                              pattern = "/")[[1]][1])

  GO_data$fold_enrichment[i] <-
    (GO_data$Count[i]/GO_data$total[i])/
    (GO_data$count_bg[i]/GO_data$total_bg[i])
}

# order df in decreasing order according to fold enrichment
GO_data <- GO_data[order(GO_data$fold_enrichment, decreasing = TRUE),]

#reduce to top 25 terms from GO_data
GO_data <- GO_data[1:25, ]

# Plotting
levels_MF_ordered <- subset(GO_data, GO_data$origin == "MF")$Description
levels_BP_ordered <- subset(GO_data, GO_data$origin == "BP")$Description
levels_CC_ordered <- subset(GO_data, GO_data$origin == "CC")$Description
test_levels <- c(levels_BP_ordered, levels_MF_ordered, levels_CC_ordered)

GO_data <- within(GO_data,
                  Description <- factor(Description,
                                        levels = rev(test_levels)))

## plot
ggplot(data = GO_data, aes(x = Description, y = fold_enrichment)) +
  scale_x_discrete(expand = c(0.05,0.05),
                   breaks = waiver())+
```

```
  scale_y_continuous(expand = c(0,0.05),
                     breaks = waiver())+
  geom_segment(aes(x=Description,
                   xend=Description,
                   y=0,
                   yend=fold_enrichment),
               color="grey",
               linetype = "dashed") +
  geom_point(aes(color = p.adjust, size = Count)) +
  scale_colour_gradient(low = "#CC1A1F", high = "#1472B9") +
  coord_flip() +
  theme(axis.line = element_line(colour = "black", size = 1),
        axis.ticks = element_line(colour = "black", size = 1),
        axis.text = element_text(size = 12, face = "bold", colour = "black"),
        axis.title = element_text(size = 12, face = "bold", colour = "black"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        legend.position = "right") +
  theme(plot.title = element_text(hjust = 0.5)) +
  ylab("Fold enrichment") +
  xlab("top25 GO-terms")
```

Technical Information

```
#R-session Information
sessionInfo()

# R version 4.1.1 (2021-08-10)
# Platform: x86_64-w64-mingw32/x64 (64-bit)
# Running under: Windows 10 x64 (build 19044)
#
# Matrix products: default
#
# locale:
# LC_COLLATE=German_Germany.1252
# LC_CTYPE=German_Germany.1252
# LC_MONETARY=German_Germany.1252 LC_NUMERIC=C
# LC_TIME=German_Germany.1252
#
# attached base packages:
# parallel  stats4    grid      stats     graphics
# grDevices utils     datasets  methods   base
#
# other attached packages:
# forcats_0.5.2         stringr_1.5.0
# purrr_0.3.5           readr_2.1.3
# tidyr_1.2.1           tibble_3.1.8
```

```
# tidyverse_1.3.2
# tidygraph_1.2.2       ggraph_2.1.0
# enrichplot_1.12.2     ReactomePA_1.36.0
# RColorBrewer_1.1-3    org.Hs.eg.db_3.13.0
# AnnotationDbi_1.54.1  IRanges_2.26.0
# S4Vectors_0.30.1      Biobase_2.52.0
# BiocGenerics_0.38.0   clusterProfiler_4.0.5
# igraph_1.3.5          network_1.18.0
# ggnetwork_0.5.10      matrixTests_0.1.9.1
# biomaRt_2.48.3        GGally_2.1.2
# ggvenn_0.1.9          dplyr_1.0.10
# limma_3.48.3          ggrepel_0.9.2
# ggpubr_0.5.0          openxlsx_4.2.5.1
# htmlwidgets_1.5.4     plotly_4.10.1
# ggplot2_3.4.0         DT_0.26
# tinytex_0.43
#
# loaded via a namespace (and not attached):
# utf8_1.2.2              tidyselect_1.2.0
# RSQLite_2.2.19          BiocParallel_1.26.2
# scatterpie_0.1.8        munsell_0.5.0
# withr_2.5.0             colorspace_2.0-3
# GOSemSim_2.18.1         filelock_1.0.2
# knitr_1.41              rstudioapi_0.14
# ggsignif_0.6.4          DOSE_3.18.2
# GenomeInfoDbData_1.2.6  polyclip_1.10-4
# bit64_4.0.5             farver_2.1.1
# pheatmap_1.0.12         downloader_0.4
# coda_0.19-4             vctrs_0.5.1
# treeio_1.16.2           generics_0.1.3
# xfun_0.35               timechange_0.1.1
# BiocFileCache_2.0.0     R6_2.5.1
# GenomeInfoDb_1.28.4     graphlayouts_0.8.4
# bitops_1.0-7            cachem_1.0.6
# reshape_0.8.9           fgsea_1.18.0
# gridGraphics_0.5-1      assertthat_0.2.1
# scales_1.2.1            googlesheets4_1.0.1
# gtable_0.3.1            rlang_1.0.6
# splines_4.1.1           rstatix_0.7.1
# lazyeval_0.2.2          gargle_1.2.1
# broom_1.0.1             checkmate_2.1.0
# yaml_2.3.6              reshape2_1.4.4
# abind_1.4-5             modelr_0.1.10
# backports_1.4.1         qvalue_2.24.0
# tools_4.1.1             ggplotify_0.1.0
# statnet.common_4.7.0    ellipsis_0.3.2
# Rcpp_1.0.9              plyr_1.8.8
# progress_1.2.2          zlibbioc_1.38.0
# RCurl_1.98-1.9          prettyunits_1.1.1
# viridis_0.6.2           cowplot_1.1.1
# haven_2.5.1             fs_1.5.2
# magrittr_2.0.3          data.table_1.14.6
# DO.db_2.9               reprex_2.0.2
```

```
# reactome.db_1.76.0      googledrive_2.0.0
# hms_1.1.2               patchwork_1.1.2
# evaluate_0.19           XML_3.99-0.13
# readxl_1.4.1            gridExtra_2.3
# compiler_4.1.1          crayon_1.5.2
# shadowtext_0.1.2        htmltools_0.5.4
# ggfun_0.0.9             tzdb_0.3.0
# aplot_0.1.9             lubridate_1.9.0
# DBI_1.1.3               tweenr_2.0.2
# dbplyr_2.2.1            MASS_7.3-58.1
# rappdirs_0.3.3          Matrix_1.5-3
# car_3.1-1cli_3.4.1      pkgconfig_2.0.3
# xml2_1.3.3              ggtree_3.0.4
# XVector_0.32.0          rvest_1.0.3
# yulab.utils_0.0.5       digest_0.6.31
# graph_1.70.0            Biostrings_2.60.2
# rmarkdown_2.18          cellranger_1.1.0
# fastmatch_1.1-3         tidytree_0.4.1
# curl_4.3.3              graphite_1.38.0
# lifecycle_1.0.3         nlme_3.1-160
# jsonlite_1.8.4          carData_3.0-5
# viridisLite_0.4.1       fansi_1.0.3
# pillar_1.8.1            lattice_0.20-45
# KEGGREST_1.32.0         fastmap_1.1.0
# httr_1.4.4              GO.db_3.13.0
# glue_1.6.2              zip_2.2.2
# png_0.1-8               bit_4.0.5
# ggforce_0.4.1           stringi_1.7.8
# blob_1.2.3              memoise_2.0.1
# ape_5.6-2
```