Visualizing Bile Acids Proteomics results

```
library(clusterProfiler)
library(org.Hs.eg.db)
library(tidyverse)
library(readxl)
library(ggprism)
library(patchwork)
library(ggtext)
library(glue)
library(umap)
```

Set up parameters - like colors

```
div_pal <- rcartocolor::carto_pal(7, "Geyser")
```

```
mq <- read_tsv(here::here("normalyzer_limma_de_results_annotated.tsv"))
```

```
## Rows: 5705 Columns: 35
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr  (4): Majority protein IDs, Protein IDs, Protein names, Gene names
## dbl (31): Cc-Lc_PValue, Cm-Lm_PValue, Cc-Lc_AdjPVal, Cm-Lm_AdjPVal, Cc-Lc_lo...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
meta <- read_tsv(here::here("norm_meta.tsv"))
```

```
## Rows: 24 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (2): sample, group
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

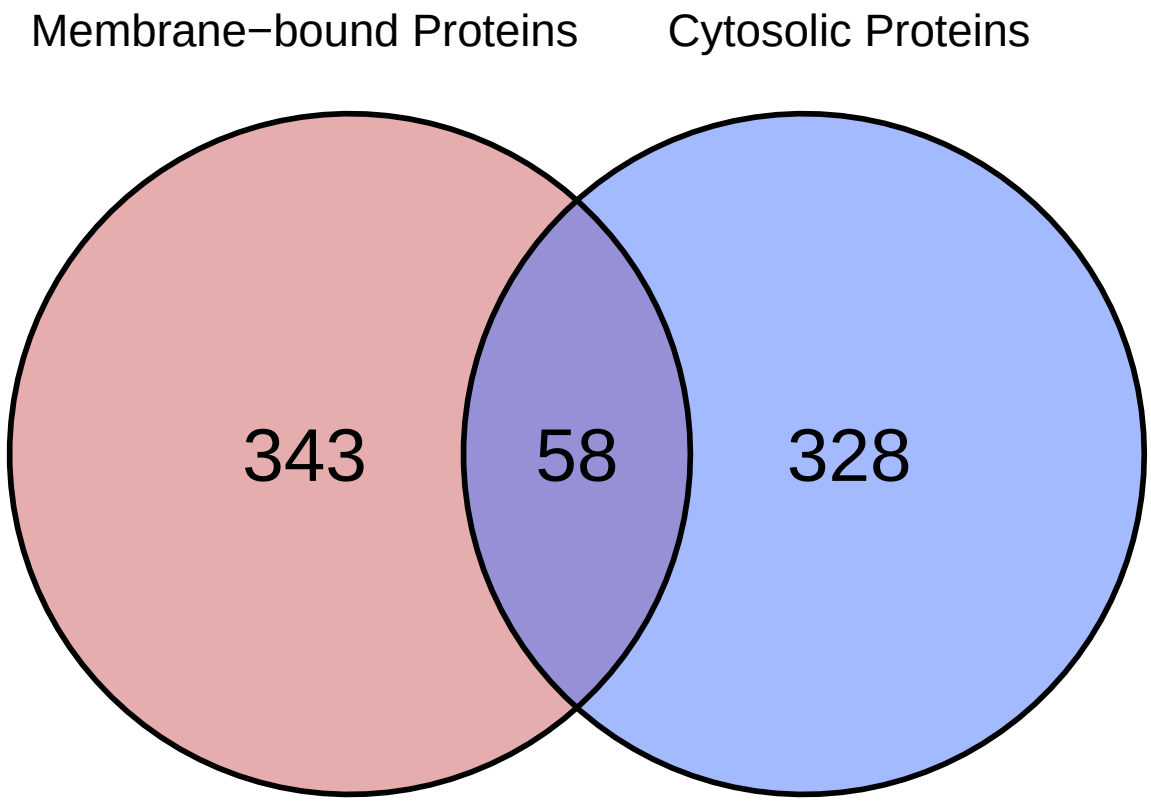
```
mq_res <- mq %>%
  select(contains("Log"), contains("Adj"), contains("PValue"), contains("Majority"), contains("Gene"))
mq_mat <- mq %>%
  select(`Gene names`, `Majority protein IDs`, matches("(C|L)[1-6](c|m)"))
#Convert the results table into a list and unify the columns names.
#Invert the foldChanges, so that comparisons are L-C instead of C-L.
#Reduce Gene names to just one gene for analysis
mq_res_list <- list(select(mq_res, -contains("Cc-Lc")),
```

```
                    select(mq_res, -contains("Cm-Lm"))) %>%
  set_names(nm = c("Membrane", "Cyto")) %>%
  map(rename_with, ~str_remove(.,"C(c|m)-L(c|m)_") ,matches("C(c|m)-L(c|m)_")) %>%
  map(mutate, across(contains("log2"), ~ -.x)) %>%
  map(separate, `Gene names`, into = letters [1:2], extra = "drop", sep = ";") %>%
  map(mutate, Genes = case_when(
    str_detect(a, "orf") & !is.na(b) & str_detect(b, "orf", negate = T) ~ b,
    TRUE ~ a
  ), Gene_rows = Genes)
```

```
## Warning: Expected 2 pieces. Missing pieces filled with ‘NA‘ in 5473 rows [3, 4, 6, 7, 8, 10, 11, 12,
## Expected 2 pieces. Missing pieces filled with ‘NA‘ in 5473 rows [3, 4, 6, 7, 8, 10, 11, 12, 13, 14,
```

Venn diagram

```
#Filter results to only include DA proteins
da_list <- mq_res_list %>%
  map(filter, AdjPVal < 0.05, abs(log2FoldChange) > 1)
#Pull just the Protein IDs for DA proteins
da_genes <- da_list %>% map(pull, `Majority protein IDs`) %>%
  set_names(nm = c("Membrane-bound Proteins", "Cytosolic Proteins"))
ggvenn::ggvenn(da_genes, text_size = 10, set_name_size = 6,
               fill_color = c("indianred", "royalblue1"), show_percentage = F)
```



```
#ggsave(here::here("Euler diagram DA proteins.png"), units = "in", dpi = 720, width = 8, height = 6)
```

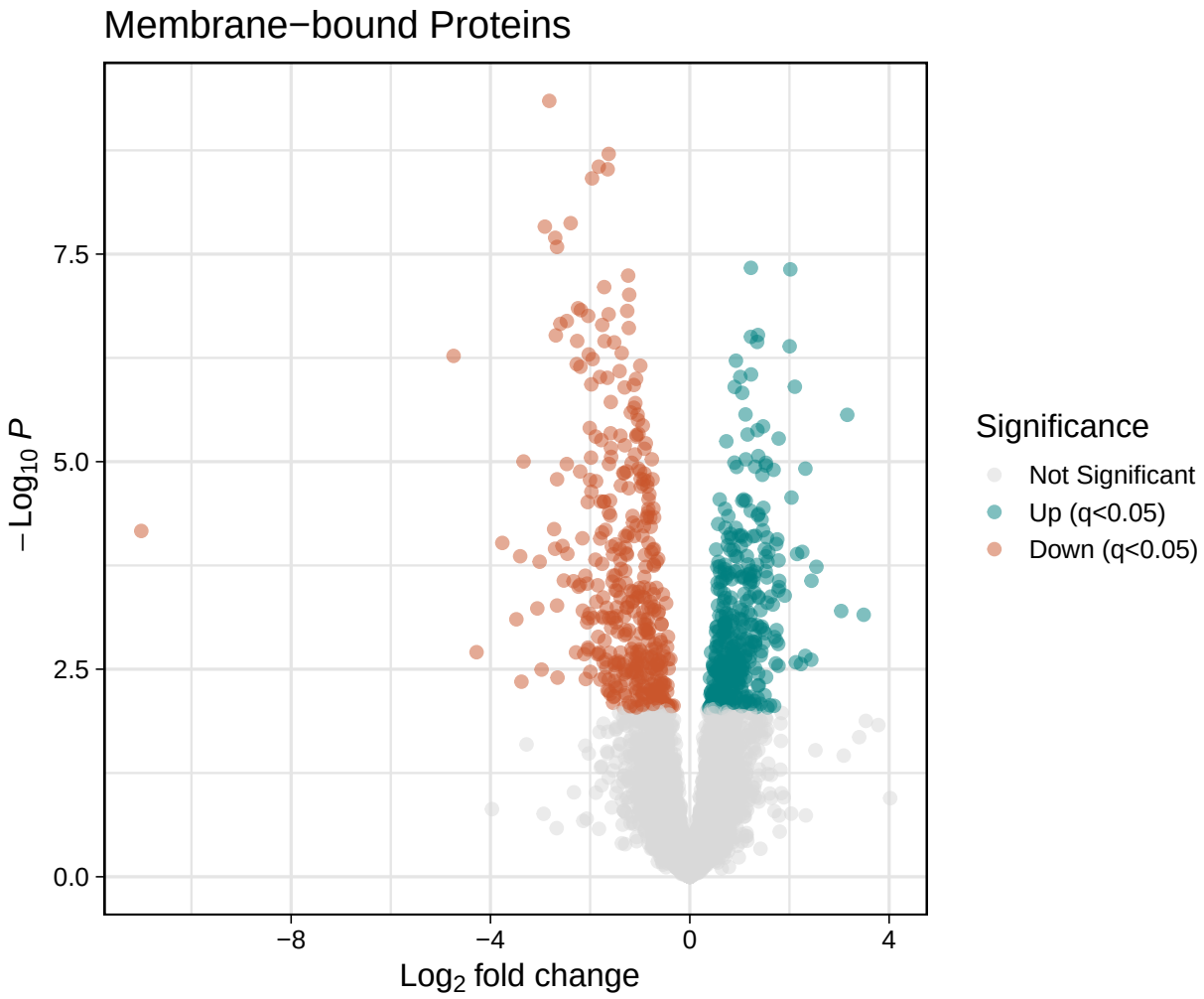
Get identities of common DE proteins

```
common_top <- da_genes %>%
  reduce(intersect)
mq_intersect_top <- mq_res %>%
  filter(`Majority protein IDs` %in% common_top)
#write_csv(mq_intersect_top, here::here("Shared DE proteins.csv"))
```

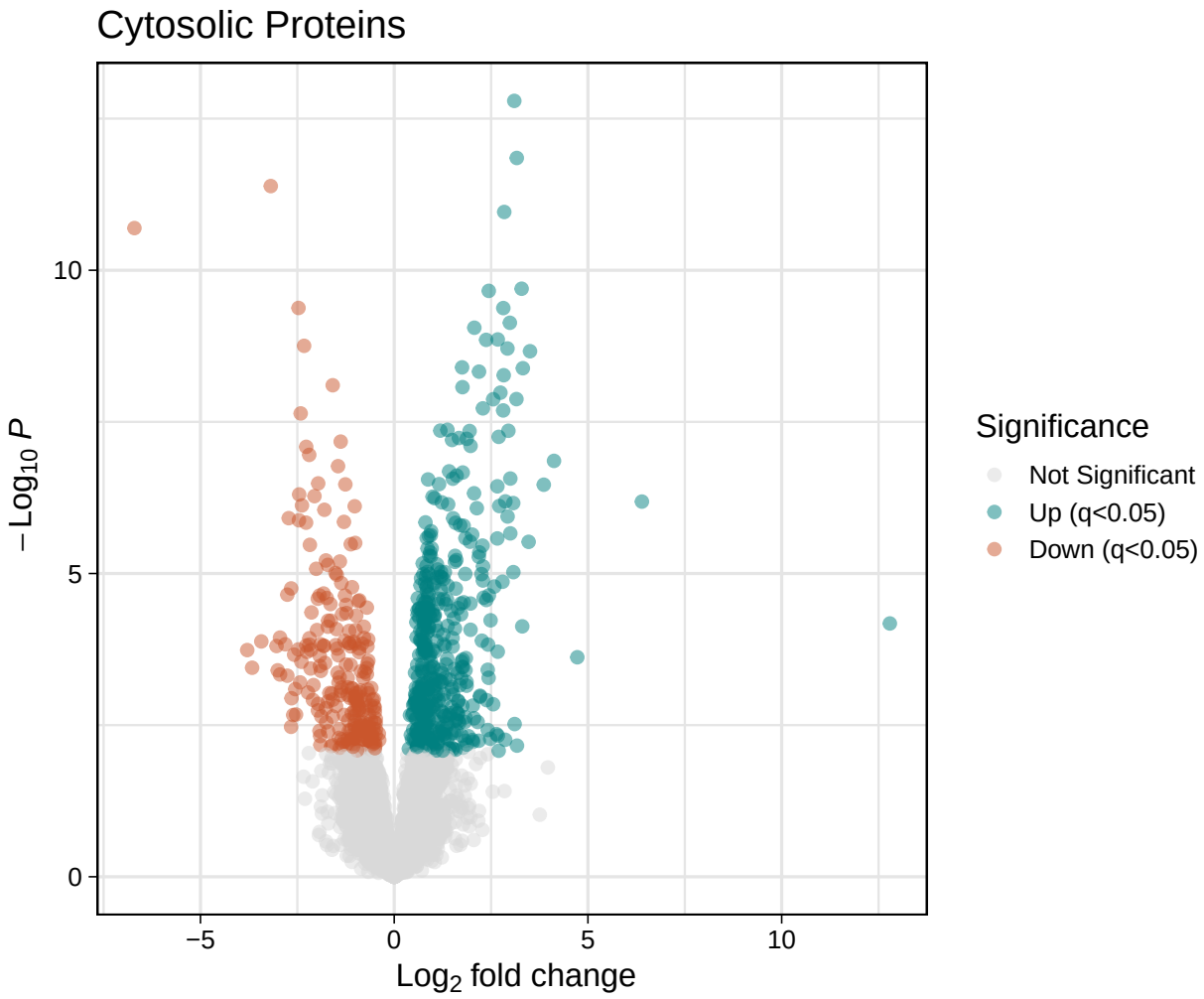
Volcano plots

```
#Add categories for signifcance
mq_res_volc_list <- mq_res_list %>%
  map(mutate,
      Significance = case_when(
        AdjPVal >= 0.05 | is.na(AdjPVal) ~ "Not Significant",
        AdjPVal < 0.05 & log2FoldChange > 0 ~ "Up (q<0.05)",
        AdjPVal < 0.05 & log2FoldChange < 0 ~ "Down (q<0.05)",
      )) %>%
  map(mutate,
      Significance = fct_relevel(Significance,"Not Significant","Up (q<0.05)","Down (q<0.05)"))

volcano_list <- mq_res_volc_list %>%
  map2(c("Membrane-bound Proteins", "Cytosolic Proteins"),
       ~ ggplot(.x, aes(x = log2FoldChange, y = -log10(PValue))) +
         geom_point(aes(fill = Significance, color = Significance), shape = 21, alpha = 0.5, size = 2.5)
         scale_color_manual(values = c("grey85",div_pal [c(1,7)])) +
         scale_fill_manual(values = c("grey85",div_pal [c(1,7)])) +
         labs(x = bquote(~Log[2] ~ "fold change"), y = bquote(~-Log[10] ~ italic(P)), title = .y) +
         theme_linedraw(base_size = 15) +
         theme(panel.grid = element_line(color =  "grey90"),
               panel.grid.major = element_line(linewidth = 0.7),
               panel.grid.minor = element_line(linewidth = 0.5),
               panel.border = element_rect(fill = NA, color = "black", linewidth = 0.9)))
walk(volcano_list, print)
```

```
## Warning: Removed 1708 rows containing missing values (‘geom_point()‘).
```




```
## Warning: Removed 1708 rows containing missing values (‘geom_point()‘).
```




```
#walk2(volcano_list, names(volcano_list),
 #    ~ggsave(here::here(str_c("final results/volcano_plot_",.y,".png")), plot = .x, width = 8, height
```

Functional enrichment

```
set.seed(711)
#Get the whole human genome as universe
human_genom <- org.Hs.egSYMBOL
mapped_genes <- mappedRkeys(human_genom)
#Try to get symbols from mq_res as universe
universe_df <- mq_res %>%
  select(Gene = `Gene names`) %>%
  deframe() %>%
  map(str_split, ";") %>%
  flatten() %>%
  flatten_chr()
h <- read.gmt(list.files(path = here::here("."),pattern = "h.all"))
c2_cp <- read.gmt(list.files(path = here::here("."),pattern = "c2.cp"))
```

```
c2_kegg <- c2_cp %>%
  filter(str_detect(term, "KEGG"))
c2_react <- c2_cp %>%
  filter(str_detect(term,"REACTOME"))
#Set up lists of genes for analysis
prots_list <- vector(mode = "list", length = 4) %>%
  set_names(map_chr(cross2(c("Membrane-bound","Cytoplasmic"),c("Up","Down")), reduce, str_c, sep = "_"))
#Select fold change cut-off and run the loop
cutoff <- 1
for(i in seq_along(prots_list)) {
  if (str_detect(names(prots_list) [i], "Membrane")) {
    res_dummy <- da_list [["Membrane"]]
    if (str_detect(names(prots_list) [i], "Up")) {
      prots_list [[i]] <- pull(filter(res_dummy, AdjPVal < 0.05, log2FoldChange > cutoff), var = "Genes")
    } else {
      prots_list [[i]] <- pull(filter(res_dummy, AdjPVal < 0.05, log2FoldChange < -cutoff), var = "Genes")
    }
  } else {
    res_dummy <- da_list [["Cyto"]]
    if (str_detect(names(prots_list) [i], "Up")) {
      prots_list [[i]] <- pull(filter(res_dummy, AdjPVal < 0.05, log2FoldChange > cutoff), var = "Genes")
    } else {
      prots_list [[i]] <- pull(filter(res_dummy, AdjPVal < 0.05, log2FoldChange < -cutoff), var = "Genes")
    }
  }
}


enrich_h_list <- map(prots_list, ~enricher(.,
                     TERM2GENE = h,
                     universe = universe_df,
                     pAdjustMethod = "BH",
                     pvalueCutoff = 0.05,
                     qvalueCutoff = 0.05))
ego_bp <- map(prots_list, ~enrichGO(gene = .,
                OrgDb         = org.Hs.eg.db,
                universe = universe_df,
                keyType       = 'SYMBOL',
                ont           = "BP",
                pAdjustMethod = "BH",
                pvalueCutoff  = 0.05,
                qvalueCutoff  = 0.05))
ego_bp_suc <- map(keep(ego_bp, ~nrow(.) > 0),enrichplot::pairwise_termsim)
ego_bp2 <- ego_bp_suc %>%
  map(~clusterProfiler::simplify(., cutoff=0.7, by="p.adjust", select_fun=min)) %>%
  set_names(str_c(names(.), "GOBP", sep = "__"))

#Create a list for different C2 databases
prots_c2_list <- cross2(prots_list, list("KEGG" = c2_kegg, "REACTOME" = c2_react)) %>%
  set_names(map_chr(cross2(names(prots_list),c("KEGG","REACTOME")), reduce, str_c, sep = "__"))

enrich_c2_list <- map(prots_c2_list, ~enricher(.x [[1]],
                     TERM2GENE = .x [[2]],
```

```
                    universe = universe_df,
                    pAdjustMethod = "BH",
                    pvalueCutoff = 0.05,
                    qvalueCutoff = 0.05))
```

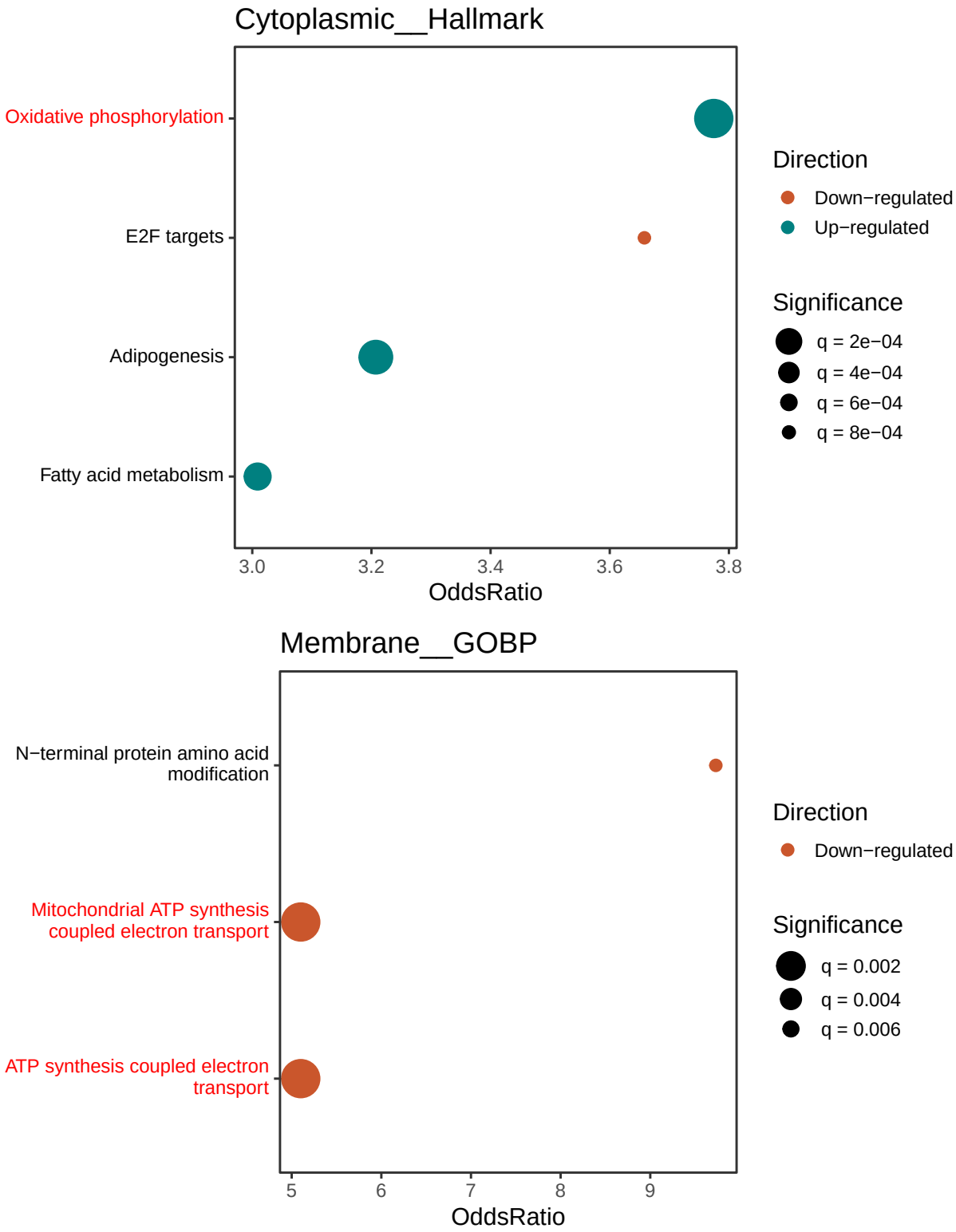
Plot the functional enrichment results. Combine the up and down regulated pathways from the same database. Calculate OddsRatio as described here

```
#Keep only groups with significant results
enrich_h_suc_list <- keep(enrich_h_list, ~nrow(.) > 0) %>%
  set_names(str_c(names(.),"Hallmark",  sep = "__"))

#Process the dataframes
enrich_df_list <- prepend(ego_bp2,enrich_h_suc_list) %>%
  prepend(keep(enrich_c2_list, ~nrow(.) > 0)) %>%
  map(as.data.frame) %>%
  map(mutate,
      num_gene_ratio = sapply(GeneRatio, function(x) eval(parse(text=x))),
      num_bg_ratio = sapply(BgRatio, function(x) eval(parse(text=x)))) %>%
  map(mutate, OddsRatio = num_gene_ratio/num_bg_ratio)
enrich_df_list <- enrich_df_list %>%
  map2(names(enrich_df_list),
       ~mutate(.x, Direction = if_else(str_detect(.y,"Up"), "Up-regulated","Down-regulated")))
#Combine the dataframes for up and down regulation
enrich_df_comb_list <- vector(mode = "list", length = 1)
names(enrich_df_comb_list) <- "placeholder"
j <- 1
#Create a loop to combine all the dataframes
for (i in seq_along(enrich_df_list)) {
  #Extract the db and fraction name from the list name
  df_name <- names(enrich_df_list) [i]
  fract_name <- str_remove(str_extract(df_name, "[:alpha:]+[-_]"),"[-_]")
  db_name <- str_remove(str_extract(df_name, "__.+"),"__")
  index <- intersect(str_which(names(enrich_df_list), fract_name),str_which(names(enrich_df_list), db_na
  enrich_df_comb_list [[j]] <- reduce(enrich_df_list [index], bind_rows)
  names(enrich_df_comb_list) [j] <- str_c(fract_name, db_name, sep = "__")
  j <- j+1
}
#Remove duplicate entries and clean up names
enrich_df_comb_unq_list <- keep(enrich_df_comb_list, !base::duplicated(enrich_df_comb_list)) %>%
  map(mutate,
      clean_ID = str_remove(Description, "(KEGG)|(REACTOME)|(HALLMARK)"),
      clean_ID = str_replace_all(clean_ID, "_", " "),
      clean_ID = str_to_sentence(clean_ID),
      clean_ID = str_replace_all(clean_ID, "[Rr]na", "RNA"),
      clean_ID = str_replace_all(clean_ID, "RRNA", "rRNA"),
      clean_ID = str_replace_all(clean_ID, "E2f", "E2F"),
      clean_ID = str_replace_all(clean_ID, "[Aa]tp", "ATP"),
      clean_ID = str_replace_all(clean_ID, " i ", " I "),
      clean_ID = str_replace_all(clean_ID, " ii", " II"),
      clean_ID = str_replace_all(clean_ID, " IIi", " III"),
      clean_ID = str_replace_all(clean_ID, "[Jj]ak", "JAK"),
      clean_ID = str_replace_all(clean_ID, " [Ss]tat", " STAT"),
```

```
      clean_ID = str_replace_all(clean_ID, "[Tt]ca", "TCA"),
      clean_ID = str_replace_all(clean_ID, "[Aa]pc c", "APC C"),
      clean_ID = str_replace_all(clean_ID, "cdh", "CDH"),
      clean_ID = str_replace_all(clean_ID, "cdc", "CDC"),
      clean_ID = str_replace_all(clean_ID, "3s", "3S"),
      clean_ID = str_replace_all(clean_ID, "k11", "K11"),
      clean_ID = str_wrap(clean_ID, width = 30))
#Print the results into csv
#walk2(enrich_df_comb_unq_list, names(enrich_df_comb_unq_list),
 #     ~write_csv(.x, file = here::here(str_c("final results/enrichment_results/",.y,".csv"))))
```

Make the dotplots of the results

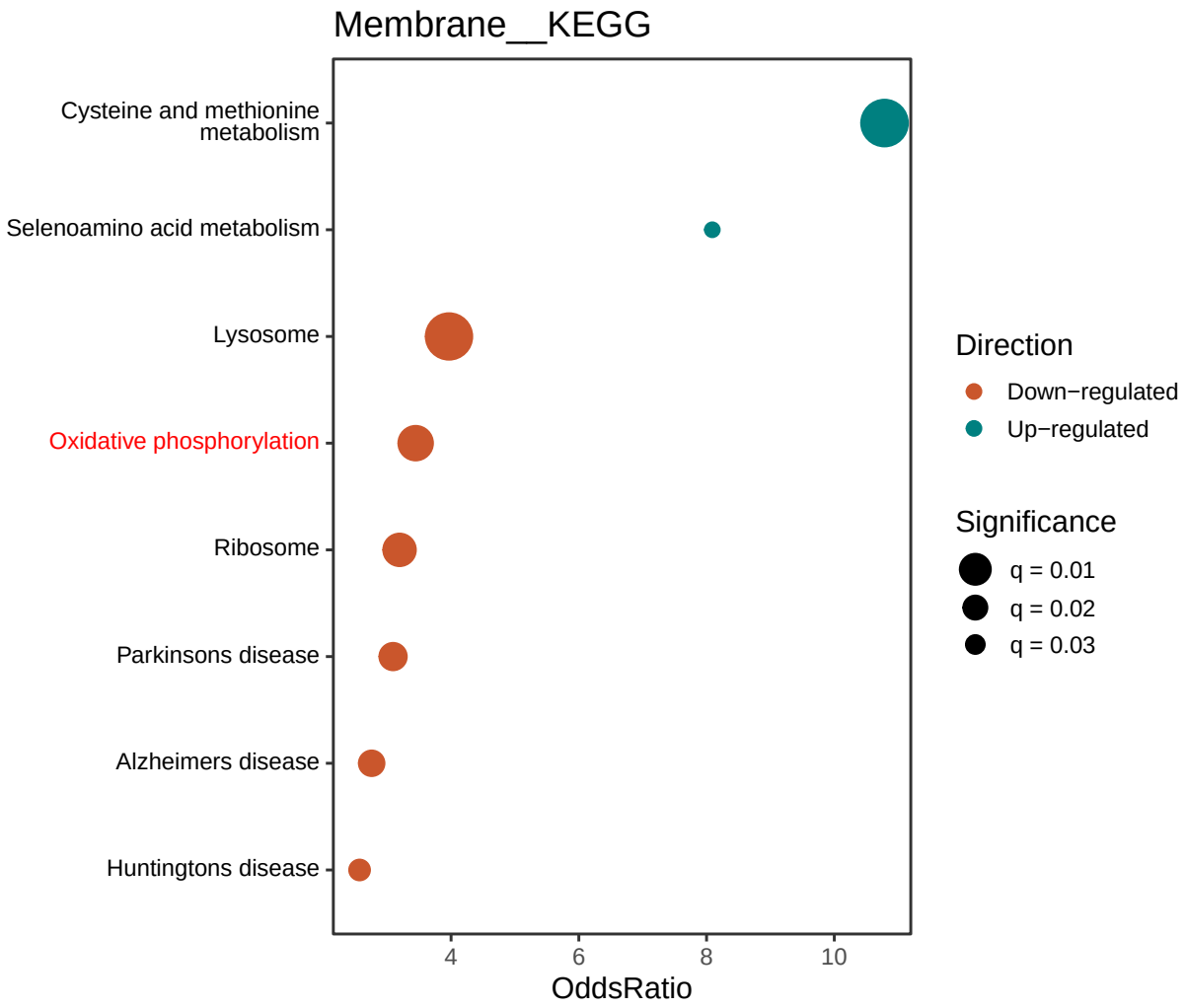
```
#Finally can plot the results
#Plot some plots separately
same_dim_dots <- enrich_df_comb_unq_list %>%
  map(mutate,
      clean_ID = fct_reorder(clean_ID, OddsRatio),
      Mito = "No")
#Prep keywords to identify mitochondrial categories
mito_key <- c("Mitochondria", "NADH", "Energy", "Respiratory", "ATP", "Oxidative", "Complex",  "Respir"
              "TCA")
#Loop through the keywords to identify the categories of relevance
same_dim_dots <- same_dim_dots %>%
  map(mutate,
      Mito = map_lgl(clean_ID, ~any(str_detect(str_to_lower(.x),str_to_lower(mito_key)))))
#Create labels
same_dim_dots_labs_list <- same_dim_dots %>%
  map(~glue_data(.x, "<span style='color: {if_else(Mito == 'TRUE', 'red', 'black')}'>{str_replace_all(cl
  map2(same_dim_dots,~set_names(.x, nm = pull(.y, var = "clean_ID")))
#Select only plots with low category number to plot
small_index <- c(5,7)
#pdf(here::here("final results/enrichment results low cat number.pdf"), width = 8, height = 5)
pwalk(list(same_dim_dots [small_index], same_dim_dots_labs_list [small_index], names(same_dim_dots) [sma
      ~print(ggplot(..1, aes(x = OddsRatio, y = clean_ID )) +
               geom_point(aes(color = Direction, size = qvalue)) +
               labs(y = element_blank(), title =..3, size = "Significance") +
               scale_color_manual(values = div_pal [c(7,1)]) +
               scale_size(range = c(10,3), labels = function(x) {str_c("q = ", x)}) +
               scale_y_discrete(labels = ..2) +
               guides(color = guide_legend(override.aes = list(size = 3))) +
               coord_cartesian(clip = "off") +
               theme_bw(base_size = 14) +
               theme(panel.grid = element_blank(),
                     axis.text.y = element_markdown(lineheight = 1.1))))
```

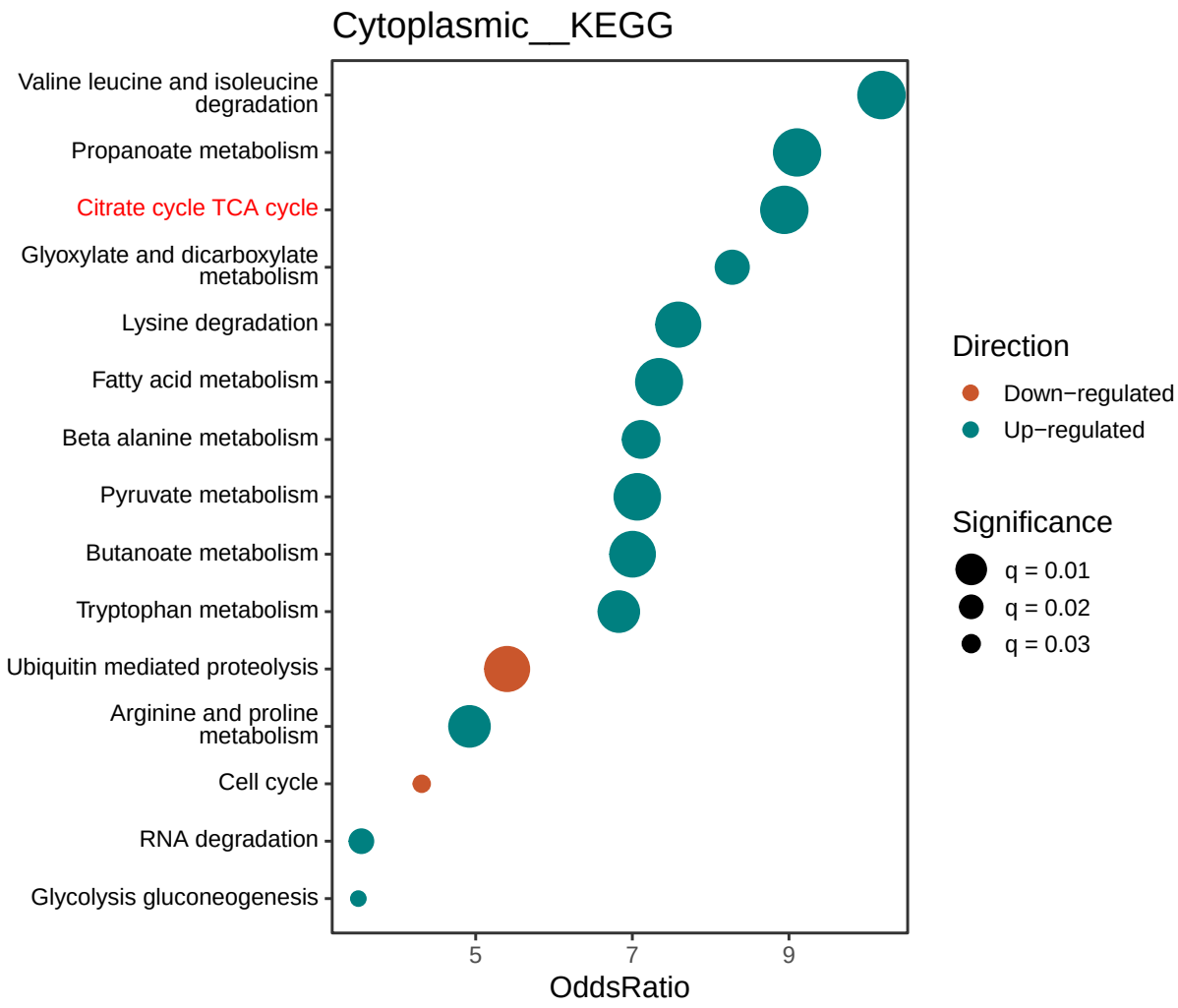


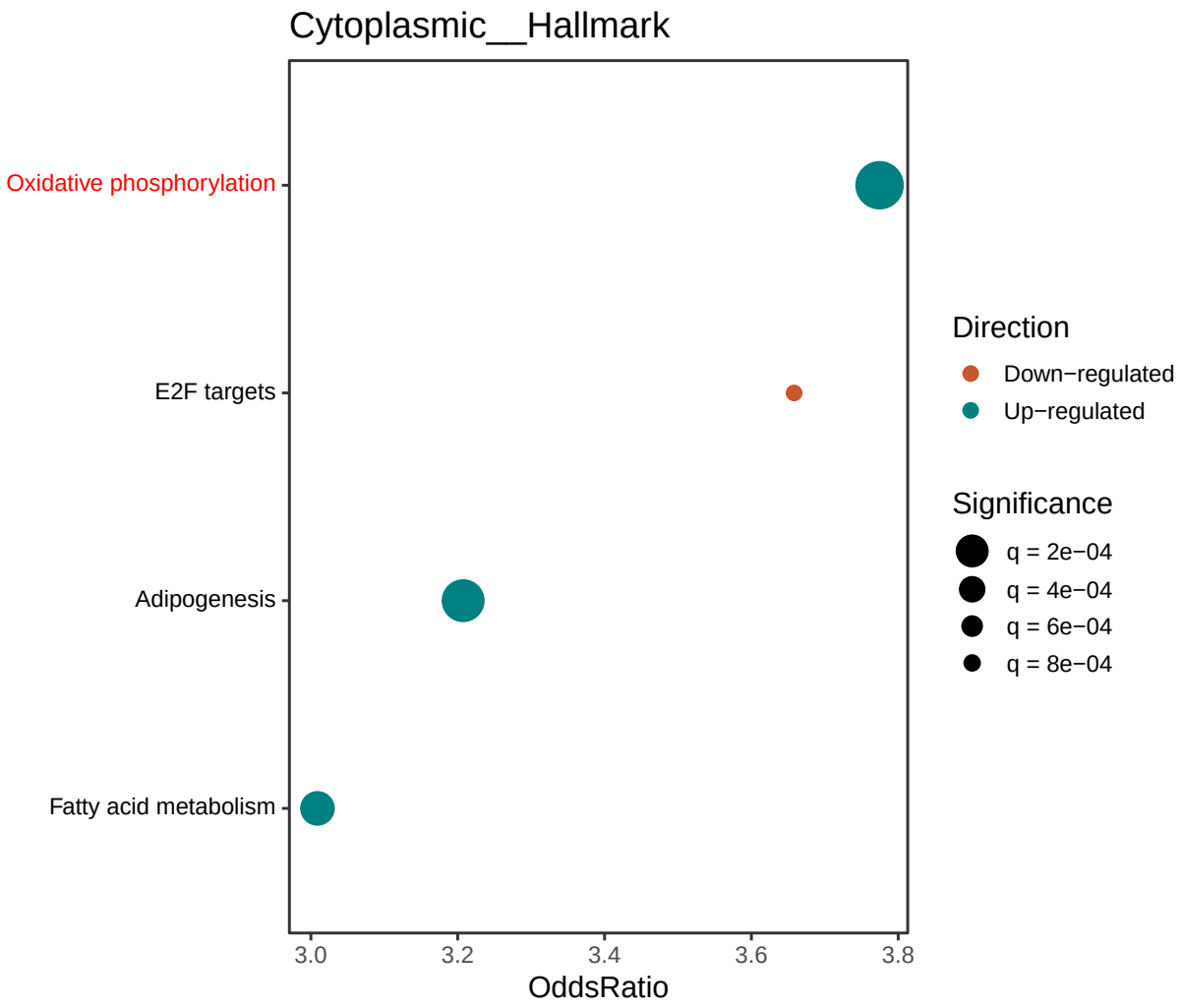

```
#dev.off()
```

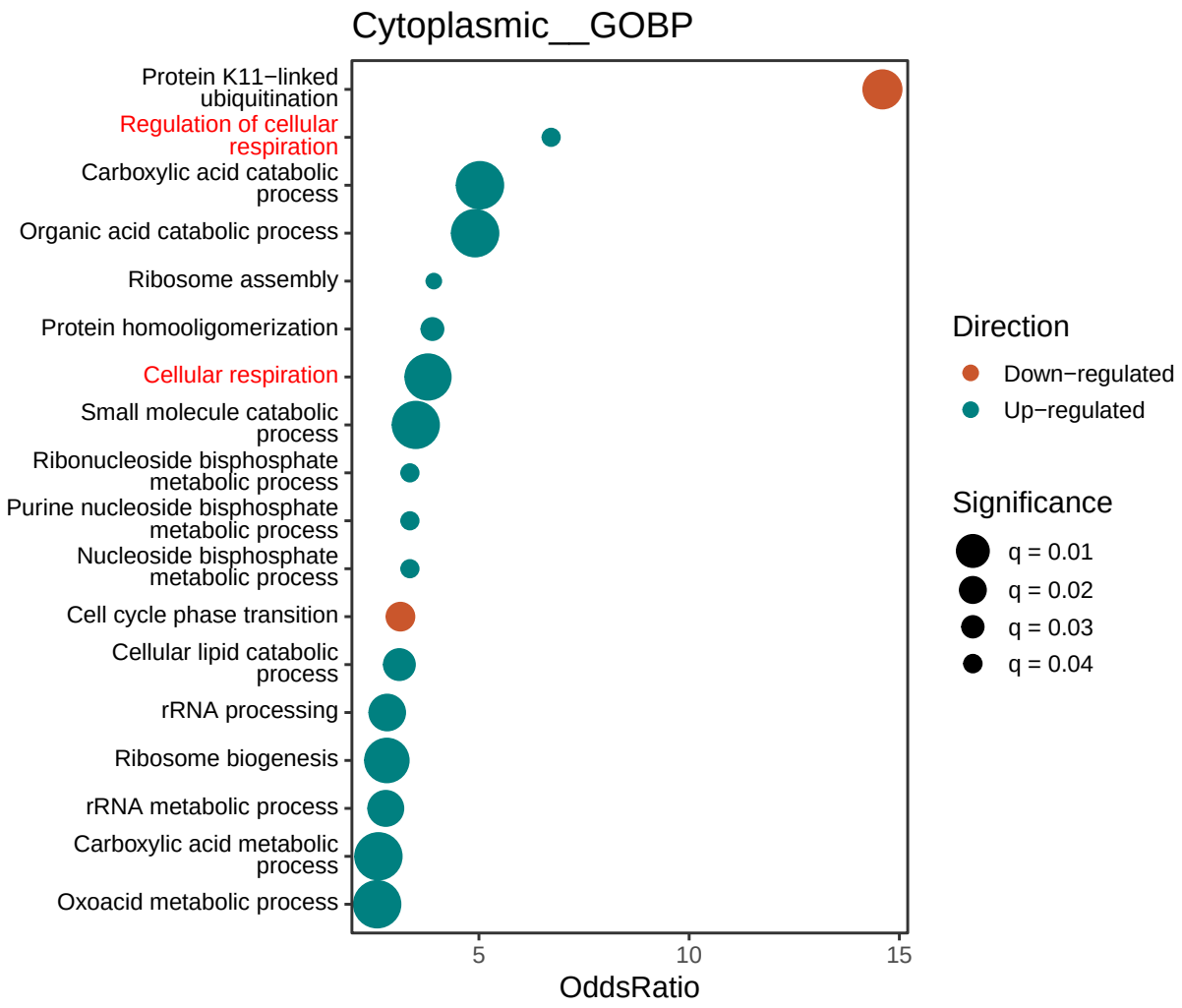
Now plot graphs of intermediate category number

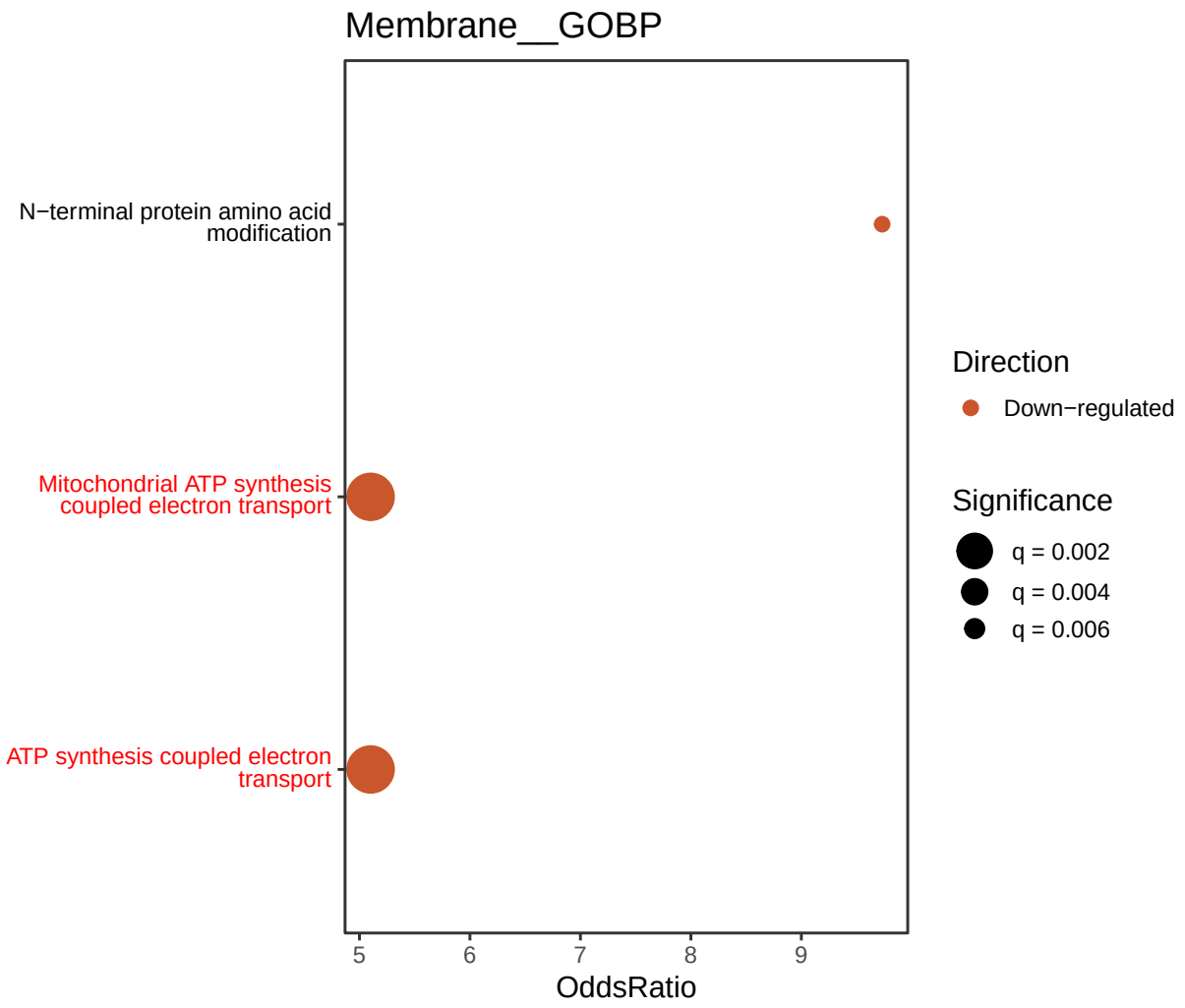
```
inter_index <- c(-3,-4)
#pdf(here::here("final results/enrichment results medium cat number.pdf"), width = 8, height = 6.66)
pwalk(list(same_dim_dots [inter_index], same_dim_dots_labs_list [inter_index], names(same_dim_dots) [in
      ~print(ggplot(..1, aes(x = OddsRatio, y = clean_ID )) +
               geom_point(aes(color = Direction, size = qvalue)) +
               labs(y = element_blank(), title =..3, size = "Significance") +
               scale_color_manual(values = div_pal [c(7,1)]) +
               scale_size(range = c(10,3), labels = function(x) {str_c("q = ", x)}) +
               scale_y_discrete(labels = ..2) +
               guides(color = guide_legend(override.aes = list(size = 3))) +
               coord_cartesian(clip = "off") +
               theme_bw(base_size = 14) +
               theme(panel.grid = element_blank(),
                     axis.text.y = element_markdown(lineheight = 0.7))))
```




Cytoplasmic__KEGG
Valine leucine and isoleucine degradation
Propanoate metabolism
Citrate cycle TCA cycle
Glyoxylate and dicarboxylate metabolism
Lysine degradation
Fatty acid metabolism
Beta alanine metabolism
Pyruvate metabolism
Butanoate metabolism
Tryptophan metabolism
Ubiquitin mediated proteolysis
Arginine and proline metabolism
Cell cycle
RNA degradation
Glycolysis gluconeogenesis
5
7
9
OddsRatio
Direction
Down−regulated
Up−regulated
Significance
q = 0.01
q = 0.02
q = 0.03


Cytoplasmic__Hallmark
Oxidative phosphorylation
E2F targets
Adipogenesis
Fatty acid metabolism
3.0
3.2
3.4
3.6
3.8
OddsRatio
Direction
Down−regulated
Up−regulated
Significance
q = 2e−04
q = 4e−04
q = 6e−04
q = 8e−04


Cytoplasmic__GOBP
Protein K11−linked ubiquitination
Regulation of cellular respiration
Carboxylic acid catabolic process
Organic acid catabolic process
Ribosome assembly
Protein homooligomerization
Cellular respiration
Small molecule catabolic process
Ribonucleoside bisphosphate metabolic process
Purine nucleoside bisphosphate metabolic process
Nucleoside bisphosphate metabolic process
Cell cycle phase transition
Cellular lipid catabolic process
rRNA processing
Ribosome biogenesis
rRNA metabolic process
Carboxylic acid metabolic process
Oxoacid metabolic process
5
10
15
OddsRatio
Direction
Down−regulated
Up−regulated
Significance
q = 0.01
q = 0.02
q = 0.03
q = 0.04

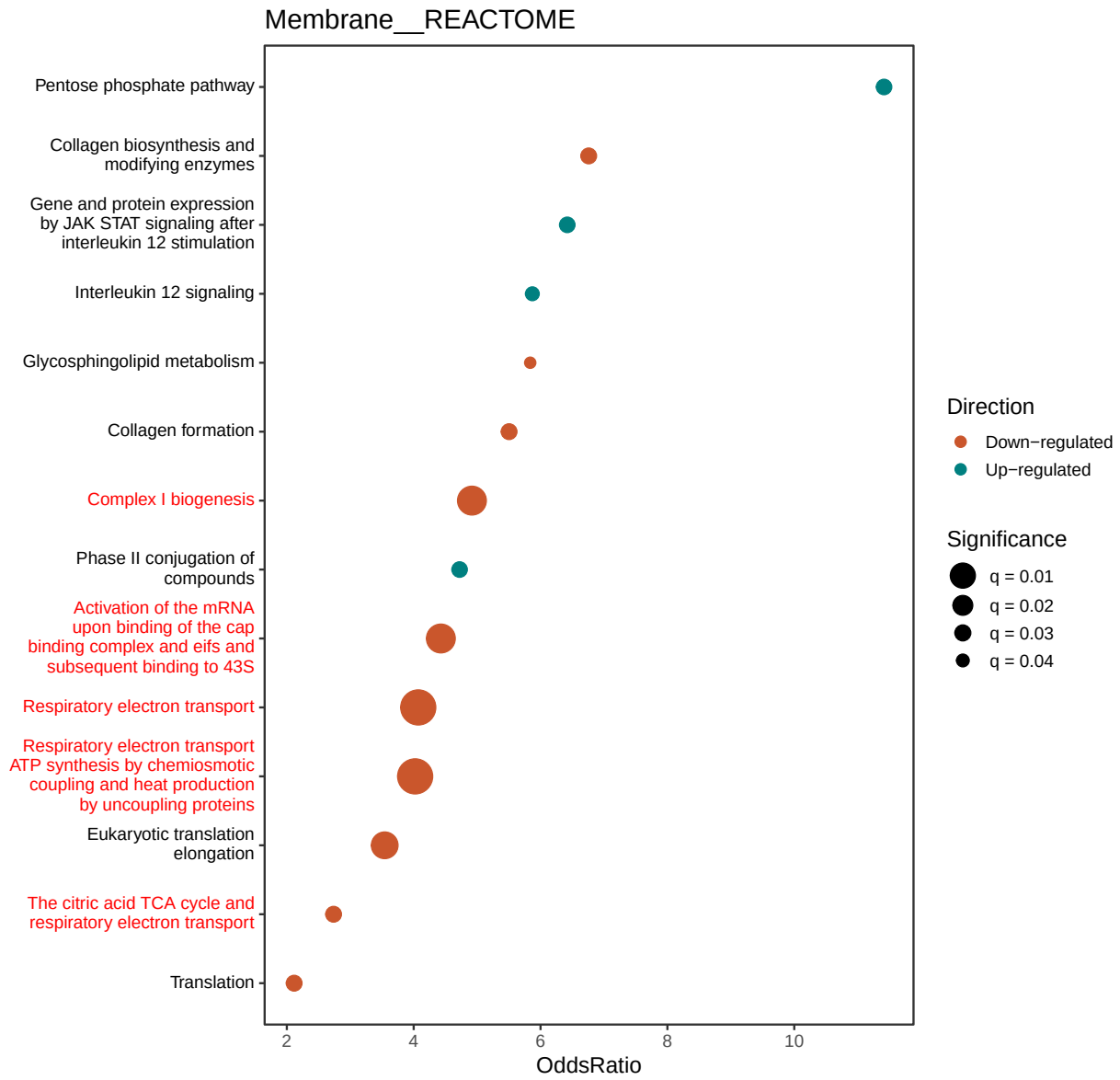


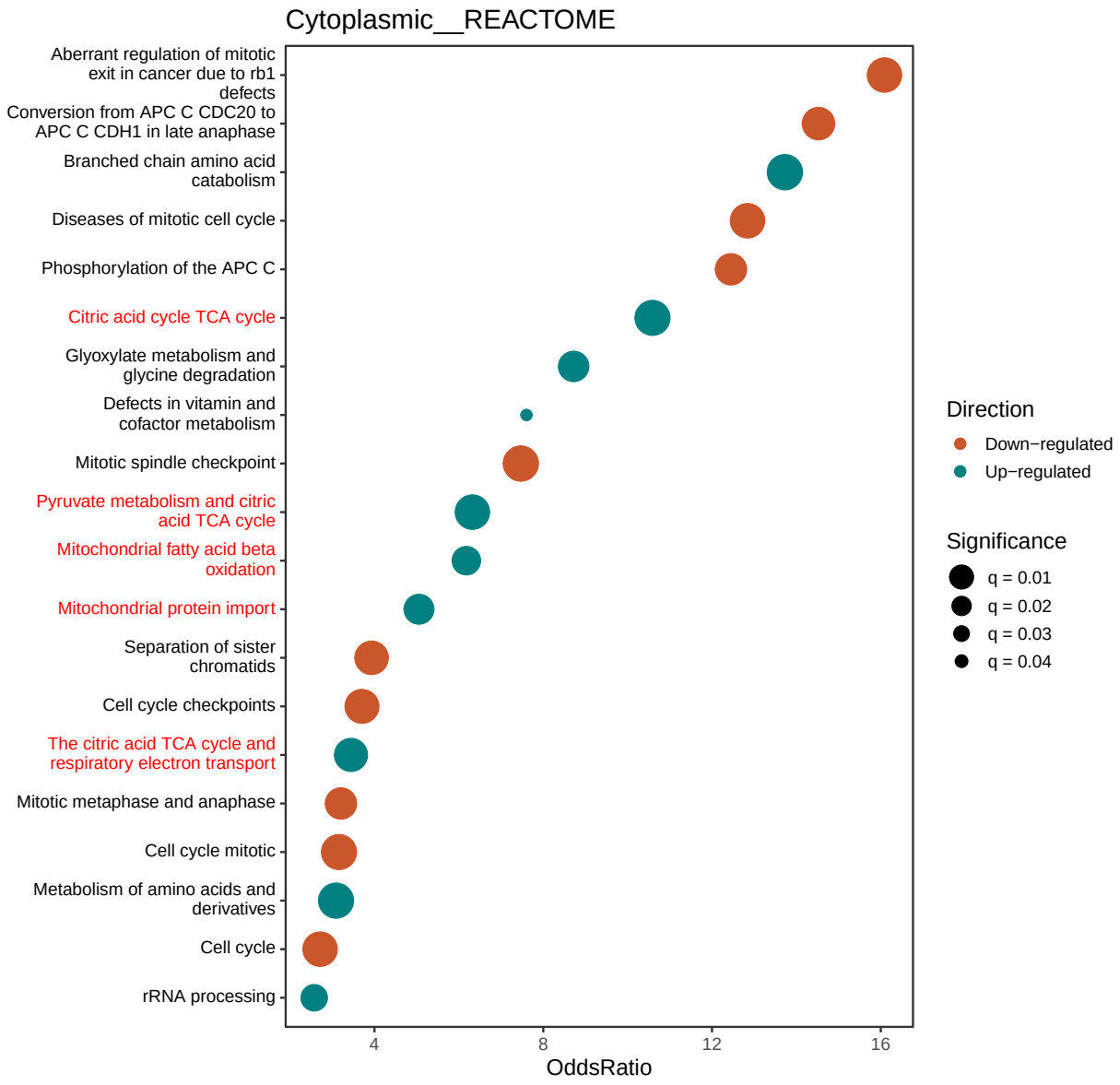

```
#dev.off()
```

Finally plot graphs of large category number

```
large_index <- c(3,4)
same_dim_dots <- same_dim_dots %>%
  map_at(large_index, slice_head, n=20)

#pdf(here::here("final results/enrichment results large cat number.pdf"), width = 10, height = 9.5)
pwalk(list(same_dim_dots [large_index], same_dim_dots_labs_list [large_index], names(same_dim_dots) [la
      ~print(ggplot(..1, aes(x = OddsRatio, y = clean_ID )) +
               geom_point(aes(color = Direction, size = qvalue)) +
               labs(y = element_blank(), title =..3, size = "Significance") +
               scale_color_manual(values = div_pal [c(7,1)]) +
               scale_size(range = c(10,3), labels = function(x) {str_c("q = ", x)}) +
               scale_y_discrete(labels = ..2) +
               guides(color = guide_legend(override.aes = list(size = 3))) +
               coord_cartesian(clip = "off") +
               theme_bw(base_size = 14) +
               theme(panel.grid = element_blank(),
```

```
    axis.text.y = element_markdown(lineheight = 1.1))))
```





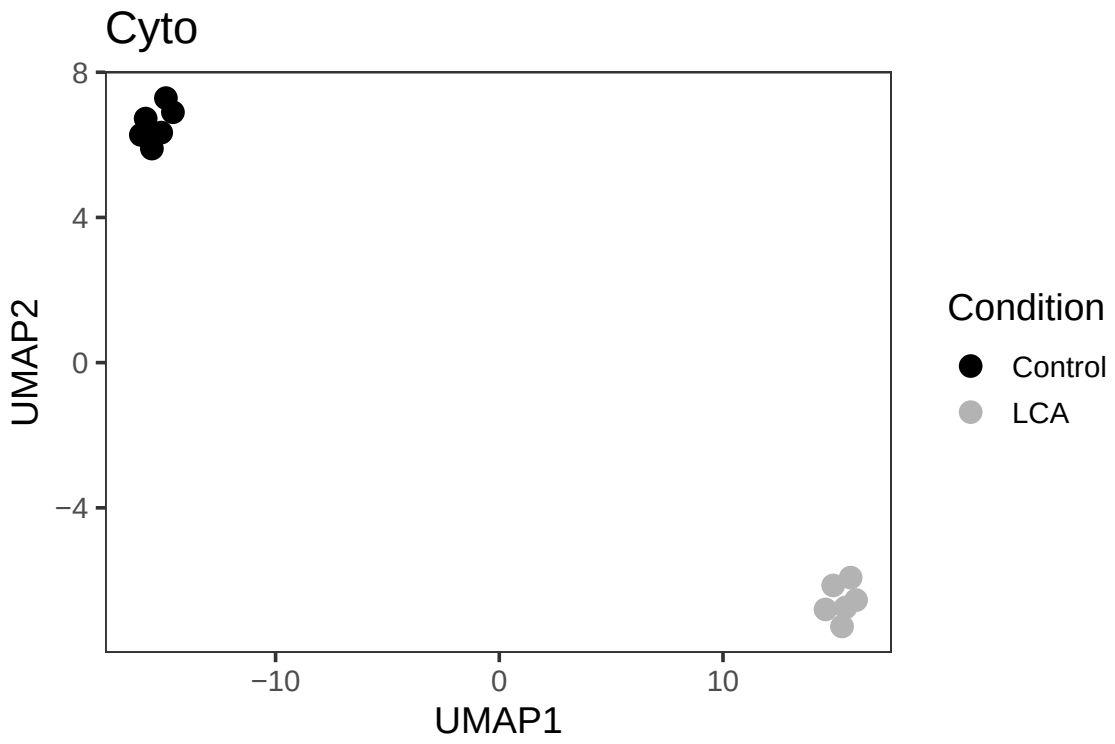

```
#dev.off()
```

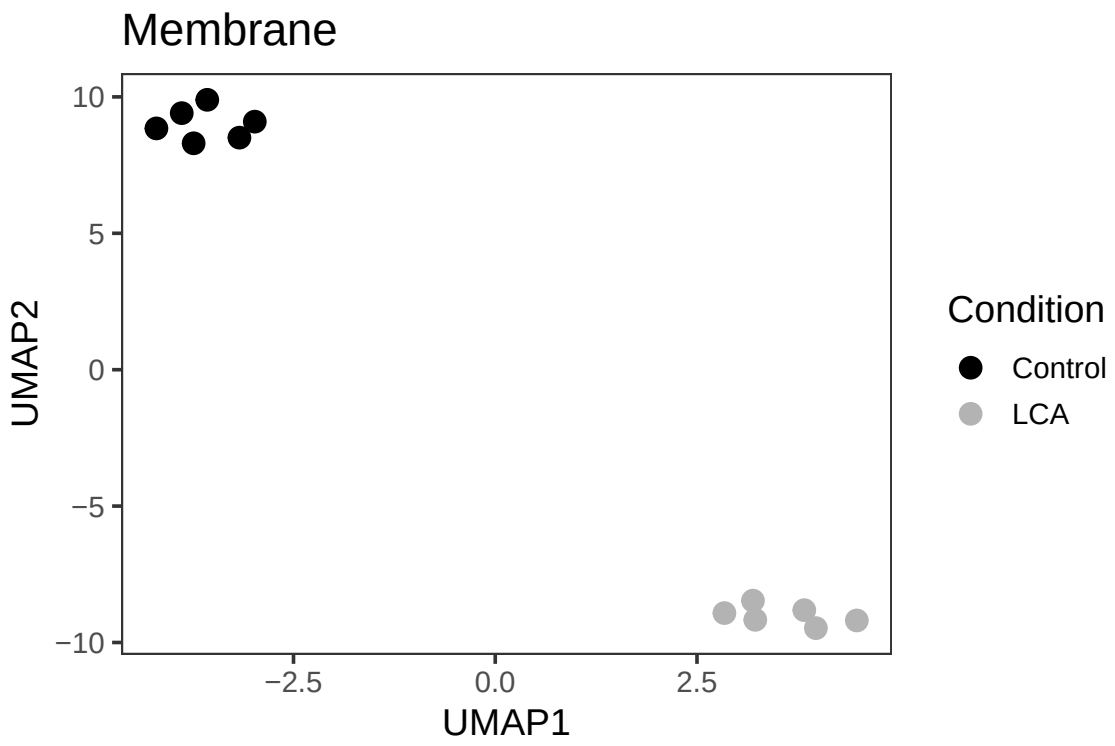
Last step is to create a UMAP graph of the data

```
#Import vsn-normalized counts
vsn_mq <- read_tsv(here::here(here::here("normalyzer/test_run/VSN-normalized.txt")))
```

```
## Rows: 5705 Columns: 25
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr  (1): Majority protein IDs
## dbl (24): C1c, C1m, C2c, C2m, C3c, C3m, C4c, C4m, C5c, C5m, C6c, C6m, L1c, L...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

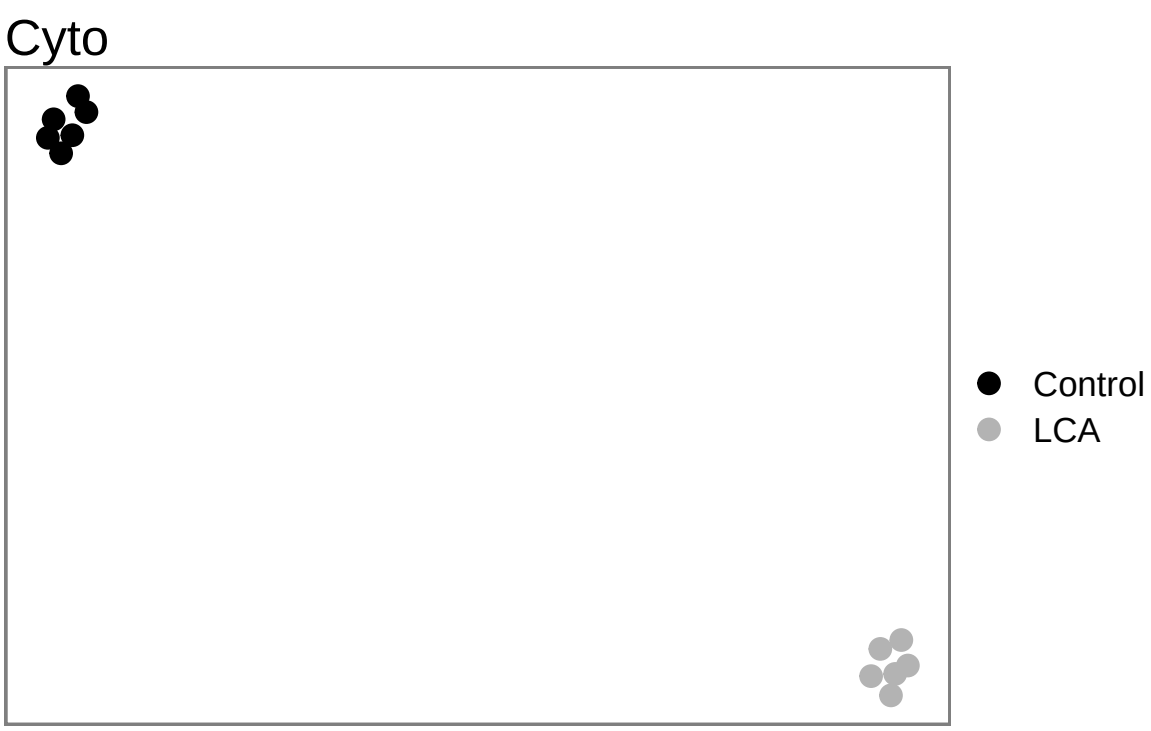
```r
vsn_mq_mat <- vsn_mq %>%
  column_to_rownames(var = "Majority protein IDs") %>%
  #vsn is akin to log-transform so replace missing values with 1s (log of 0 is 1)
  mutate(across(.fns = ~as.numeric(str_replace_na(.x, replacement = 1)))) %>%
  as.matrix() %>%
  t()
#Create list of fraction-based samples
vsn_mq_list <- map(set_names(c("c","m"), nm = c("Cyto","Membrane")),
                   ~ vsn_mq_mat [str_detect(rownames(vsn_mq_mat), .x),])
#Run umap
#Reduce number of neighbours
custom_sets <- umap.defaults
custom_sets$n_neighbors <- 4
umap_fit_list <- map(vsn_mq_list, umap, config = custom_sets)
#Extract coordinates
umap_df_list <- umap_fit_list %>%
  map(~.x$layout) %>%
  map(as.data.frame) %>%
  map(rename, UMAP1 = V1, UMAP2 = V2) %>%
  map(rownames_to_column, var = "sample") %>%
  map(mutate, Condition = if_else(str_detect(sample, "L"), "LCA", "Control"))

#Plot the results
#pdf(here::here("final results/UMAP plots.pdf"), width = 6, height = 4)
walk2(umap_df_list, names(umap_df_list),
      ~print(ggplot(.x, aes(UMAP1,UMAP2, color = Condition)) +
               geom_point(size = 3.5) +
               theme_bw(base_size = 14) +
               labs(title = .y) +
               scale_color_manual(values = c("black","grey70")) +
               theme(panel.grid = element_blank())
               ))
```






```
walk2(umap_df_list, names(umap_df_list),
      ~print(ggplot(.x, aes(UMAP1,UMAP2, color = Condition)) +
               geom_point(size = 3.5) +
               labs(title = .y, x = " ", y = " ") +
```

```
              theme_void(base_size = 16) +
              scale_color_manual(values = c("black","grey70")) +
              theme(panel.border = element_rect(linewidth = 1, color = "grey50", fill = NA),
                    panel.grid = element_blank(),
                    legend.title = element_blank(),
                    axis.title = element_text())
    ))
```




```
#dev.off()
```

Session Info

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19044)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_Canada.1252  LC_CTYPE=English_Canada.1252
## [3] LC_MONETARY=English_Canada.1252 LC_NUMERIC=C
## [5] LC_TIME=English_Canada.1252
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] umap_0.2.9.0         glue_1.6.2           ggtext_0.1.2
##  [4] patchwork_1.1.2      ggprism_1.0.3        readxl_1.4.1
##  [7] forcats_0.5.2        stringr_1.4.1        dplyr_1.0.9
```

```
## [10] purrr_0.3.4           readr_2.1.3           tidyr_1.2.1
## [13] tibble_3.1.7          ggplot2_3.4.0         tidyverse_1.3.2
## [16] org.Hs.eg.db_3.14.0   AnnotationDbi_1.56.2  IRanges_2.28.0
## [19] S4Vectors_0.32.4      Biobase_2.54.0        BiocGenerics_0.40.0
## [22] clusterProfiler_4.2.2
##
## loaded via a namespace (and not attached):
##   [1] shadowtext_0.1.2       backports_1.4.1        fastmatch_1.1-3
##   [4] plyr_1.8.7             igraph_1.3.5           lazyeval_0.2.2
##   [7] splines_4.1.2          BiocParallel_1.28.3    GenomeInfoDb_1.30.1
##  [10] digest_0.6.29          yulab.utils_0.0.5      htmltools_0.5.2
##  [13] GOSemSim_2.20.0        viridis_0.6.2          GO.db_3.14.0
##  [16] fansi_1.0.2            magrittr_2.0.3         memoise_2.0.1
##  [19] googlesheets4_1.0.1    tzdb_0.3.0             Biostrings_2.62.0
##  [22] graphlayouts_0.8.3     modelr_0.1.10          vroom_1.6.0
##  [25] askpass_1.1            timechange_0.1.1       enrichplot_1.14.2
##  [28] colorspace_2.0-3       blob_1.2.3             rvest_1.0.3
##  [31] ggrepel_0.9.1          haven_2.5.1            xfun_0.35
##  [34] crayon_1.5.2           RCurl_1.98-1.8         jsonlite_1.8.0
##  [37] scatterpie_0.1.8       ape_5.6-2              polyclip_1.10-0
##  [40] gtable_0.3.1           gargle_1.2.1           zlibbioc_1.40.0
##  [43] XVector_0.34.0         scales_1.2.1           DOSE_3.20.1
##  [46] DBI_1.1.3              Rcpp_1.0.9             gridtext_0.1.5
##  [49] viridisLite_0.4.1      reticulate_1.26        gridGraphics_0.5-1
##  [52] tidytree_0.4.1         bit_4.0.5              httr_1.4.4
##  [55] fgsea_1.20.0           RColorBrewer_1.1-3     ellipsis_0.3.2
##  [58] pkgconfig_2.0.3        farver_2.1.1           dbplyr_2.2.1
##  [61] here_1.0.1             utf8_1.2.2             labeling_0.4.2
##  [64] ggplotify_0.1.0        tidyselect_1.2.0       rlang_1.0.6
##  [67] reshape2_1.4.4         munsell_0.5.0          cellranger_1.1.0
##  [70] tools_4.1.2            cachem_1.0.6           downloader_0.4
##  [73] cli_3.4.1              generics_0.1.3         RSQLite_2.2.17
##  [76] broom_1.0.1            evaluate_0.18          fastmap_1.1.0
##  [79] yaml_2.2.2             ggtree_3.2.1           knitr_1.41
##  [82] bit64_4.0.5            fs_1.5.2               tidygraph_1.2.2
##  [85] KEGGREST_1.34.0        ggraph_2.1.0           nlme_3.1-153
##  [88] aplot_0.1.9            ggvenn_0.1.9           DO.db_2.9
##  [91] xml2_1.3.3             compiler_4.1.2         rstudioapi_0.14
##  [94] png_0.1-7              reprex_2.0.2           treeio_1.18.1
##  [97] tweenr_2.0.2           stringi_1.7.6          highr_0.9
## [100] RSpectra_0.16-1        lattice_0.20-45        Matrix_1.3-4
## [103] commonmark_1.8.1       markdown_1.4           vctrs_0.5.1
## [106] pillar_1.8.1           lifecycle_1.0.3        data.table_1.14.6
## [109] bitops_1.0-7           rcartocolor_2.0.0      qvalue_2.26.0
## [112] R6_2.5.1               gridExtra_2.3          MASS_7.3-54
## [115] assertthat_0.2.1       rprojroot_2.0.3        openssl_2.0.4
## [118] withr_2.5.0            GenomeInfoDbData_1.2.7 parallel_4.1.2
## [121] hms_1.1.2              grid_4.1.2             ggfun_0.0.9
## [124] rmarkdown_2.18         googledrive_2.0.0      ggforce_0.4.1
## [127] lubridate_1.9.0
```