analysis_5.R

```
library(tidyverse)
library(limma)
library(ggbeeswarm)
library(ggpubr)

file_names <- list.files('gpr/', pattern='.gpr')
file_paths <- paste('gpr/', file_names, sep="")


# All columns from STARTSCRIPT.R
gpr_columns <- c("Block", "Column", "Row", "Name", "ID", "X", "Y", "Dia.",
                 "F635 Median", "F635 Mean", "F635 SD", "F635 CV", "B635",
                 "B635 Median", "B635 Mean", "B635 SD", "B635 CV",
                 "% > B635+1SD", "% > B635+2SD", "F635 % Sat.", "F532 Median",
                 "F532 Mean", "F532 SD", "F532 CV", "B532", "B532 Median",
                 "B532 Mean", "B532 SD", "B532 CV", "% > B532+1SD",
                 "% > B532+2SD", "F532 % Sat.", "Ratio of Medians (635/532)",
                 "Ratio of Means (635/532)", "Median of Ratios (635/532)",
                 "Mean of Ratios (635/532)", "Ratios SD (635/532)",
                 "Rgn Ratio (635/532)", "Rgn R2 (635/532)", "F Pixels",
                 "B Pixels", "Circularity", "Sum of Medians (635/532)",
                 "Sum of Means (635/532)", "Log Ratio (635/532)",
                 "F635 Median - B635", "F532 Median - B532", "F635 Mean - B635",
                 "F532 Mean - B532", "F635 Total Intensity",
                 "F532 Total Intensity", "SNR 635", "SNR 532", "Flags",
                 "Normalize", "Autoflag")

# My subset of columns
gpr_subset <- c("Name", "ID",
                "F635 Median", "F635 Mean", "F635 SD",
                "B635 Median", "B635 Mean", "B635 SD",
                "F532 Median", "F532 Mean", "F532 SD",
                "B532 Median", "B532 Mean", "B532 SD",
                "F Pixels", "B Pixels", "Circularity",
                "F635 Median - B635", "F532 Median - B532",
                "Flags")

# Easy variable names for subset of columns (snake case)
gpr_colnames <- c("name", "id",
                  "f635_median", "f635_mean", "f635_sd",
                  "b635_median", "b635_mean", "b635_sd",
                  "f532_median", "f532_mean", "f532_sd",
                  "b532_median", "b532_mean", "b532_sd",
                  "f_pixels", "b_pixels", "circularity",
                  "fb635_median", "fb532_median",
                  "flags")

## DATA IMPORT
```

```
gpr = tibble()
pools = paste("pool", 1:4, sep="_")
sets = paste("set", 1:2, sep="_")

for(i in 1:8) {
  slide <- read.maimages(file_paths[i], source="genepix.median",
                         other.columns=gpr_subset)$other
  slide <- as.data.frame(slide)
  colnames(slide) = gpr_colnames

  slide <- as_tibble(slide) %>%
    mutate(pool = pools[ceiling(i/2)],
           set = sets[2-i%%2])

  gpr <- rbind(gpr, slide)
}

## FILTERING AND DB%
db <- read_tsv("data/fragment_rsd.txt") %>%
  select(PrEST, `%DB`) %>%
  rename(name = PrEST, db = `%DB`)

d_clean <- gpr %>%
  # Incorporate %DB
  left_join(db, by = "name") %>%
  # Filter rows that are flagged, non-HPPR, too small or too weak
  filter(flags == 0,
         str_starts(name, "HPRR"),
         f_pixels >= 30,
         f532_median - b532_median > 10 * b532_sd) %>%
  # If there are replicates, only keep the strongest one
  arrange(name, pool, desc(f532_median)) %>%
  distinct(name, pool, .keep_all = T)


## TRANSFORM AND DEFINE SIGNAL
calc_stats <- function(d){
  mfi <- d$fb635_median
  mn <- mean(mfi)
  md <- median(mfi)
  sd <- sd(mfi)
  # log2_mfi <- log2(mfi)
  # log2_mn <- mean(log2_mfi)
  # log2_md <- median(log2_mfi)
  # log2_sd <- sd(log2_mfi)
  d_new <- d %>%
    mutate(mfi = mfi,
           mns = (mfi-mn) / mn,
           mds = (mfi-mn) / md,
           sds = (mfi-mn) / sd)
           # log2_mfi = log2(mfi),
           # log2_mns = (log2_mfi - log2_mn) / log2_sd,
```

```r
          # log2_mds = (log2_mfi - log2_mn) / log2_sd,
          # log2_sds = (log2_mfi - log2_mn) / log2_sd)
  return(d_new)}

# 1) Use global stats
# d_rank <- calc_stats(d_clean) %>%
#   select(name, pool, set, all_of(stats)) %>%
#   arrange(desc(sds))

# 2) Use per pool stats
# d_rank = tibble()
# for(p in pools){
#   pool <- calc_stats(filter(d_clean, pool == p))
#   d_rank <- rbind(d_rank, pool)}
# d_rank <- d_rank %>%
#   select(name, pool, set, all_of(stats)) %>%
#   arrange(desc(sds))

# 3) Use per glass stats
d_rank = tibble()
for(p in pools){
  for(s in sets){
    glass <- calc_stats(filter(d_clean, pool == p, set == s))
    d_rank <- rbind(d_rank, glass)}}
d_rank <- d_rank %>%
  select(name, db, pool, set, sds, everything()) %>%
  arrange(desc(sds))

## Incorporate gene info
d_42k <- as_tibble(read.delim("data/42k_array.txt")) %>%
  rename(name = PrEST)

d_id <- d_rank %>%
  left_join(d_42k, by = "name")
d_id <- d_id %>%
  arrange(pool, desc(sds)) %>%
  mutate(prank = c(1:nrow(filter(d_id, pool == "pool_1")),
                   1:nrow(filter(d_id, pool == "pool_2")),
                   1:nrow(filter(d_id, pool == "pool_3")),
                   1:nrow(filter(d_id, pool == "pool_4")))) %>%
  # Remove rows with no Uniprot ID
  filter(Uniprot != "") %>%
  # Remover rows with multiple Uniprot ID
  # filter(str_count(Uniprot, ";")+1 == 1)
  arrange(desc(sds)) %>%
  # Keep all except antigen names
  select(name, db, pool, set, sds, prank, Gene, Gene.desc, Uniprot,
         everything(), -Ag.name)

## Write Uniprot IDs
# to_write <- filter(d_id, sds > 4)
# write_csv(as.data.frame(to_write$Uniprot), col_names = F,
```

```
#           file = paste("data/uniprot_id_",
#                        format(Sys.time(),"%y%m%d_%H%M%S"),
#                        ".txt", sep = ""))
# -> then manually obtain GO-list from Uniprot and save as uniprot_GO.tsv

## Incorporate downloaded GO and flag keywords
go <- read_tsv("data/uniprot_GO.tsv") %>%
  rename(Uniprot = Entry)

# Create regexp filters based on keywords
kw_bioprocess <- paste(
  read_delim("keywords/keywords_bioprocess.txt", "\t", col_names=F)$X1,
  collapse = "|")
kw_cellcomp <- paste(
  read_delim("keywords/keywords_cellcomp.txt", "\t", col_names=F)$X1,
  collapse = "|")

d_go <- d_id %>%
  left_join(go, by = "Uniprot") %>%
  arrange(desc(sds)) %>%
  # kw1: immunological function
  # kw2: extracellular/membrane localization
  mutate(kw1 = str_detect(`Gene ontology (biological process)`, kw_bioprocess),
         kw2 = str_detect(`Gene ontology (cellular component)`, kw_cellcomp)) %>%
  select(name, db, sds, prank, pool, kw1, kw2, Gene,
         Uniprot, `Protein names`, names(go),
         everything(), -Gene.desc)

# Subset all >4 sd
d_sub <- filter(d_go, sds>4)

# Subset kw
d_kw_and <- filter(d_sub, kw1 == TRUE & kw2 == TRUE)
d_kw_or <- filter(d_sub,
                  !(kw1 == TRUE & kw2 == TRUE) & (kw1 == TRUE | kw2 == TRUE))

## Write results
# write_csv(d_sub, file = paste("results",format(Sys.time(),"%y%m%d_%H%M%S"),
#                               ".txt", sep = ""))

################################################################################

## Write de-duplicated results
d_dedup <- d_sub %>%
  add_count(name) %>%
  distinct(name, .keep_all = TRUE)
# write_tsv(d_dedup, file = paste("dedup_results",
#                                 format(Sys.time(),"%y%m%d_%H%M%S"),".txt",
#                                 sep = ""))

## Write gene list for antigen collection
# gene_list <- str_split(d_sub$Gene, ";")
```

```
# gene_vector <- unlist(gene_list)
# write(gene_vector, "42k_genes.txt")

## OVERLAP WITH LITERATURE GENES
# 24 genes from literature represented by 55 prests in assay
# lit <- paste(read_tsv("../Antigenpanel/lit_genes.txt")$Gene,
#              collapse = "|") %>%
#   str_to_upper()
# lit_prest <- filter(d_42k, str_detect(Gene, lit))$name
# lit_go <- filter(d_go, str_detect(Gene, lit))
# count(lit_go, pool)
```

plot.R

```
## PLOTS
# Which data structure will be used for plotting?
###############
d_plot <- d_sub
###############

pool_index <- c(1:nrow(filter(d_plot, pool == "pool_1")),
                1:nrow(filter(d_plot, pool == "pool_2")),
                1:nrow(filter(d_plot, pool == "pool_3")),
                1:nrow(filter(d_plot, pool == "pool_4")))
(count_df <- count(d_plot, pool, set))

# Plot fall-off for each pool
ggplot(arrange(d_plot, pool, desc(sds))) +
  geom_point(aes(y = sds,
                 x = pool_index,
                 color = pool,
                 shape = set),
             size = 1.5) +
  geom_hline(yintercept = 4)

# Boxplot of pools and sets
ggplot(d_plot, aes(x = pool, y = sds)) +
  geom_boxplot(outlier.shape = NA, aes(fill = set)) +
  geom_quasirandom(dodge.width=0.8, size = 0.9, aes(group = set)) +
  geom_text(inherit.aes = F,
            stat="count",
            aes(x = pool,label=..count.., group = set, color = set),
            y = 2.5,
            position = position_dodge(width = 0.8))

# Wilcox difference between sets for every pool
ggplot(d_plot, aes(x = set, y = sds, fill = set)) +
  geom_boxplot() +
  facet_wrap(~ pool, ncol = 2) +
  geom_quasirandom(dodge.width=0.8) +
  stat_compare_means(label.y = 75)

# Plot xy-correlation
ggplot(d_plot, aes(x=sds, y=f635_median, color = pool, shape = set)) +
  geom_point(size=2)
```

wrangle.R

```
library(tidyverse)

## Prerequisite step
# LIMS: 1) Input list of genes from 42k results and literature studies
#       2) Export df containing HPRR and HPRA, save as lims_genes.xls

# Here we merge those gene lists, remove rows without HPRA and de-duplicate HPRR
lims_42k_hpra <- read_tsv("lims_42k_genes.xls") %>%
  mutate(origin = "42k_genes")
lims_lit_genes <- read_tsv("lims_lit_genes.xls") %>%
  mutate(origin = "lit_genes")

hprr2collect <- rbind(lims_42k_hpra, lims_lit_genes) %>%
  select(-X4) %>%
  filter(`Ag name` != "") %>%
  distinct(PrEST, .keep_all = T)


# This goes to position-finding script
write_tsv(hprr2collect, "AK_covid.tsv")
```

pos_2.R

```
library(tidyverse)

# Import hprr2gene
hprr2gene <- as_tibble(read.delim("../42k/data/42k_array.txt")) %>%
  select(PrEST, Gene, Gene.desc)

# Make position df with gene info
plock <- read_tsv("AK_covid_pos.txt") %>%
  # Remove mostly empty columns occupying namespace
  select(-Gene, -PrEST.1) %>%
  left_join(hprr2gene, by = "PrEST") %>%
  select(PrEST, Ag.name, Gene, Gene.desc, everything())

# Make position df without gene info
hprr2pos <- plock %>%
  select(-Gene, -Gene.desc)

pos_info <- names(hprr2pos)[2:length(hprr2pos)]


# 1) 42k hpra
d_42k <- read_tsv("42k_dedup.txt") %>%
  # Rename "name" -> "PrEST
  rename(PrEST = name)
# This df has HPRA as unique key, meaning all other columns may be copied for some rows
d_42k_pos <- d_42k %>%
  inner_join(hprr2pos, by = "PrEST") %>%
  distinct(Ag.name, .keep_all = T) %>%
  mutate(prio = "1_42k") %>%
  select(prio, PrEST, db, sds, Gene, pos_info)

# 2) Literature genes
d_lit <- read_tsv("lims_lit_genes.xls") %>%
  filter(`Ag name` != "") %>%
  select(-X4, -`Ag name`)
d_lit_pos <- d_lit %>%
  inner_join(hprr2pos, by = "PrEST") %>%
  mutate(sds = NA, db = NA) %>%
  mutate(prio = "1_lit") %>%
  select(prio, PrEST, db, sds, Gene, pos_info)

# 3) Fill upp remainder of panel on HPRR that:
# - were not reactive themselves
# - represents genes that showed other reactive HPRR
# - represents genes that were immunologically relevant and extracellularly accessible
# - are sorted in descending signal strength
bonus_genes <- unique(filter(d_42k, kw1 == T & kw2 == T)$Gene)
bonus_genes_sds <- d_42k %>%
  filter(kw1 == T & kw2 == T) %>%
  select(Gene, sds) %>%
```

```
  distinct(Gene, .keep_all=T)
bl <- str_split(plock$Gene,";") %in% bonus_genes
d_bonus_pos <- plock[bl,] %>%
  left_join(bonus_genes_sds, by = "Gene") %>%
  mutate(db = NA) %>%
  mutate(prio = "2_bonus") %>%
  select(prio, PrEST, db, sds, Gene, pos_info) %>%
  arrange(desc(sds))

final <- rbind(d_42k_pos, d_lit_pos, d_bonus_pos) %>%
  arrange(prio,
          ProjBox.4,
          ProjBox.3,
          ProjBox.2,
          ProjBox.1)


## WRITE RESULTS
# write_tsv(d_lit_pos, file = paste("plock_lit_",format(Sys.time(),"%y%m%d_%H%M%S"),".xls", s
ep = ""), na = "")
```