p-Values on PD Output using imputeLCMD

Alexander Ondrus

13/09/2020

Libraries and Selector Functions

Note that `gmm`, one of the dependencies of `imputeLCMD`, uses Fortran code. If you are running this on macOS, you will need to have GNU Fortran 8.2 installed.

All of the selector functions below assume that the column names follow the Proteome Discover output conventions.

```
library(tidyverse)
library(readxl)
library(imputeLCMD)
library(ggthemes)

select_abundance_cols <- function(df){
  # Selects all of the abundance columns of df
  return(select(df , starts_with("Abundance: F")))
}

select_dmso_cols <- function(abundance_cols){
  # Selects all of the DMSO columns _from the abundance columns_ of df,
  # i.e. use select_abundance_cols first
  return(select(abundance_cols, contains("DMSO")))
}

select_probe_cols <- function(abundance_cols){
  # Selects all of the probe columns _from the abundance columns_ of df,
  # i.e. use select_abundance_cols first
  return(select(abundance_cols, contains("XM,")))
}

select_competitor_plus_probe_cols <- function(abundance_cols){
  # Selects all of the XM_GZ columns _from the abundance columns_ of df,
  # i.e. use select_abundance_cols first
  return(select(abundance_cols, contains("XM_GZ")))
}

num_missing <- function(df){
  # Returns a vector of the number of missing entries from each row of df
  # generic, not PD output specific
  return(apply(df, 1, (function(x) sum(is.na(x)))))
}

filter_no_probe_rows <- function(df){
```

```
  # Filters out all of the rows that are missing all of the entries in the
  # probe columns (PD output specific)
  abundance_cols <- select_abundance_cols(df)
  probe_cols <- select_probe_cols(abundance_cols)
  num_of_probe_cols <- dim(probe_cols)[2]
  num_missing_probes <- num_missing(probe_cols)
  return(filter(df, num_missing_probes < num_of_probe_cols))
}

gen_pvals <- function(log_difference_df){
  # Assumes that log_difference_df is the log-difference between two
  # data frames that you want to compare, i.e. apply log2 to each df
  # and then take the difference between the two before applying this
  # function.
  # Returns: vector of p-values for one-sample, two-tailed t-test with
  # H_0: mu = 0, H_a: mu \neq 0
  return(
    apply(log_difference_df,
          1,
          (function(x) t.test(x)$p.value))
  )
}

xm_dmso_pvals <- function(path){
  # Takes in the path for a PD output data frame and filters out
  # all rows that have no probe measurements

  # Returns a data frame with the following columns:
  #   Gene_Symbol
  #   log2_fold_change
  #   p_val
  #   num_imputed_xm
  #   num_imputed_dmso


  PD_output <- read_excel(path) %>%
    filter_no_probe_rows()

  log_abundance_cols <- select_abundance_cols(PD_output) %>%
    log2()

  log_xm <- select_probe_cols(log_abundance_cols)
  num_imputed_xm <- num_missing(log_xm)
  imputed_xm <- impute.MinProb(log_xm)

  log_dmso <- select_dmso_cols(log_abundance_cols)
  num_imputed_dmso <- num_missing(log_dmso)
  imputed_dmso <- impute.MinProb(log_dmso)

  log_diff_df <- imputed_xm - imputed_dmso
  p_val <- gen_pvals(log_diff_df)
  log2_fold_change <- apply(log_diff_df, 1, mean)
```

```
  return_df <- data.frame(Gene_Symbol = PD_output$`Gene Symbol`,
                          log2_fold_change,
                          p_val,
                          num_imputed_xm,
                          num_imputed_dmso) %>%
    cbind(imputed_xm, imputed_dmso) %>%
    filter(!is.na(Gene_Symbol))

  return(return_df)
}

xm_compete_pvals <- function(path){
  # Takes in the path for a PD output data frame and filters out
  # all rows that have no probe measurements

  # Returns a data frame with the following columns:
  #   Gene_Symbol
  #   log2_fold_change
  #   p_val
  #   num_imputed_xm
  #   num_imputed_compete


  PD_output <- read_excel(path) %>%
    filter_no_probe_rows()

  log_abundance_cols <- select_abundance_cols(PD_output) %>%
    log2()

  log_xm <- select_probe_cols(log_abundance_cols)
  num_imputed_xm <- num_missing(log_xm)
  imputed_xm <- impute.MinProb(log_xm)

  log_compete <- select_competitor_plus_probe_cols(log_abundance_cols)
  num_imputed_compete <- num_missing(log_compete)
  imputed_compete <- impute.MinProb(log_compete)

  log_diff_df <- imputed_xm - imputed_compete
  p_val <- gen_pvals(log_diff_df)
  log2_fold_change <- apply(log_diff_df, 1, mean)

  return_df <- data.frame(Gene_Symbol = PD_output$`Gene Symbol`,
                          log2_fold_change,
                          p_val,
                          num_imputed_xm,
                          num_imputed_compete) %>%
    cbind(imputed_xm, imputed_compete) %>%
    filter(!is.na(Gene_Symbol))

  return(return_df)
}
```

Applying to PD Output

The commands below apply the imputation functions to the mass spec data.

```
Jul13_Aug11_XM_DMSO_Sept13 <- xm_dmso_pvals("../data/Yu-Shiuan_TMT6_13July20_11Aug20_RTS-3hr_2_Nested.x
```

```
## [1] 0.624015
## [1] 0.8193359
```

```
Jul13_Aug11_XM_GZ_Sept13 <- xm_compete_pvals("../data/Yu-Shiuan_TMT6_13July20_11Aug20_RTS-3hr_2_Nested.
```

```
## [1] 0.624015
## [1] 0.5796569
```

```
Mar13_XM_DMSO_Dec9 <- xm_dmso_pvals("../data/20201209_Yu-Shiuan_13Mar20_Sequest_Reviewed_ProteinAbundac
```

```
## [1] 0.4528488
## [1] 0.5591684
```

```
Aug20_XM_DMSO_Dec9 <- xm_dmso_pvals("../data/20201209_Yu-Shiuan_TMT10_20August20_RTS-3hr_Nested.xlsx")
```

```
## [1] 1.053789
## [1] 0.6764546
```

```
Aug20_XM_GZ_Dec9 <- xm_compete_pvals("../data/20201209_Yu-Shiuan_TMT10_20August20_RTS-3hr_Nested.xlsx")
```

```
## [1] 1.053789
## [1] 1.158842
```

Plotting the Data

This function generates a volcano plot for the output of the above commands.

```
gen_volcano_plot <- function(pvals_df, type = ""){
  pvals_df$Total_Imputed <- select(pvals_df, starts_with("num_")) %>%
    rowSums()
  pvals_df$Imputed <- (pvals_df$Total_Imputed >0)
  p <- ggplot(pvals_df,
              aes(x = log2_fold_change,
                  y = -log10(p_val),
                  colour = Imputed)) +
    geom_point(alpha = 0.5) +
    labs(title = paste("Volcano Plot for", type),
         subtitle = "Using Minimal Probabilistic Imputation",
         x = "Log2-fold Change",
         y = "-log10 of p-Value") +
    scale_colour_manual(name = NULL,
                        labels = c("Imputation Not Required","Imputation Required"),
                        values = c("red", "blue")) +
    theme_fivethirtyeight()
return(p)
}
```