raw_data_analysis

2022-09-21

check working directory is correctly set-up

```
getwd()
```

```
## [1] "/Users/cgu3/Library/CloudStorage/OneDrive-InsideMDAnderson/proteomics/missing protien project ma
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.5
## v tibble  3.2.1     v dplyr   1.1.2
## v tidyr   1.2.1     v stringr 1.4.1
## v readr   2.1.3     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# install.packages("devtools")
if (!require("protools", quietly = TRUE))
   devtools::install_github("https://github.com/FDUguchunhui/protools")
library('protools')
```

Import data

```
# import gastric primary cell nsaf data that filtered with Spectral count (SpC) >=2
gastric_nsaf <- as.matrix(read.csv('proteomics-data/gastric_primary_cell_nsaf_2.csv', row.names = 1))
# # import gastric primary cell SpC data
gastric_count <- read_csv('proteomics-data/gastric_primary_cell_count.csv', show_col_types = FALSE)
```

```
## New names:
## * `` -> `...1`
```

```
# rename the index column as "accession"
colnames(gastric_count)[1] <- 'accession'
# import information for mapping between accession, gene symbol, and ensembl ID
omics_metadata <- read_csv('support-data/omics_metadata.csv', show_col_types = FALSE)
missing_protein_df <- readxl::read_xlsx('support-data/PE2-5.xlsx')
```

Create regular protein set and missing protein set

create a character vector of missing protein accession only use P2-4 proteins from nextprot the regular protein is not the just a complementary of missing protein, need to filter P1-5 proteins

```
# missing_proteins is a character vector of MP gene symbols
# similar P2_5_proteins are protein gene symbols under P2-5 category in nextprot
# P2_5_proteins will be used for negative filtering to get regular proteins
missing_proteins <- missing_protein_df %>% filter(PE != 'Uncertain') %>% select(`gene name(s)`) %>% pull
P2_5_proteins <- missing_protein_df  %>% select(`gene name(s)`) %>% pull

# missing_proteins is a character vector of MP accession ID
# similar P2_5_proteins are protein accesion ID under P2-5 category in nextprot
missing_proteins_accession <- missing_protein_df %>% filter(PE != 'Uncertain') %>% select(Accession) %>%
P2_5_proteins_accession <- missing_protein_df %>% select(Accession) %>% pull
```

check the number of SpC for each protein for both missing and regular proteins

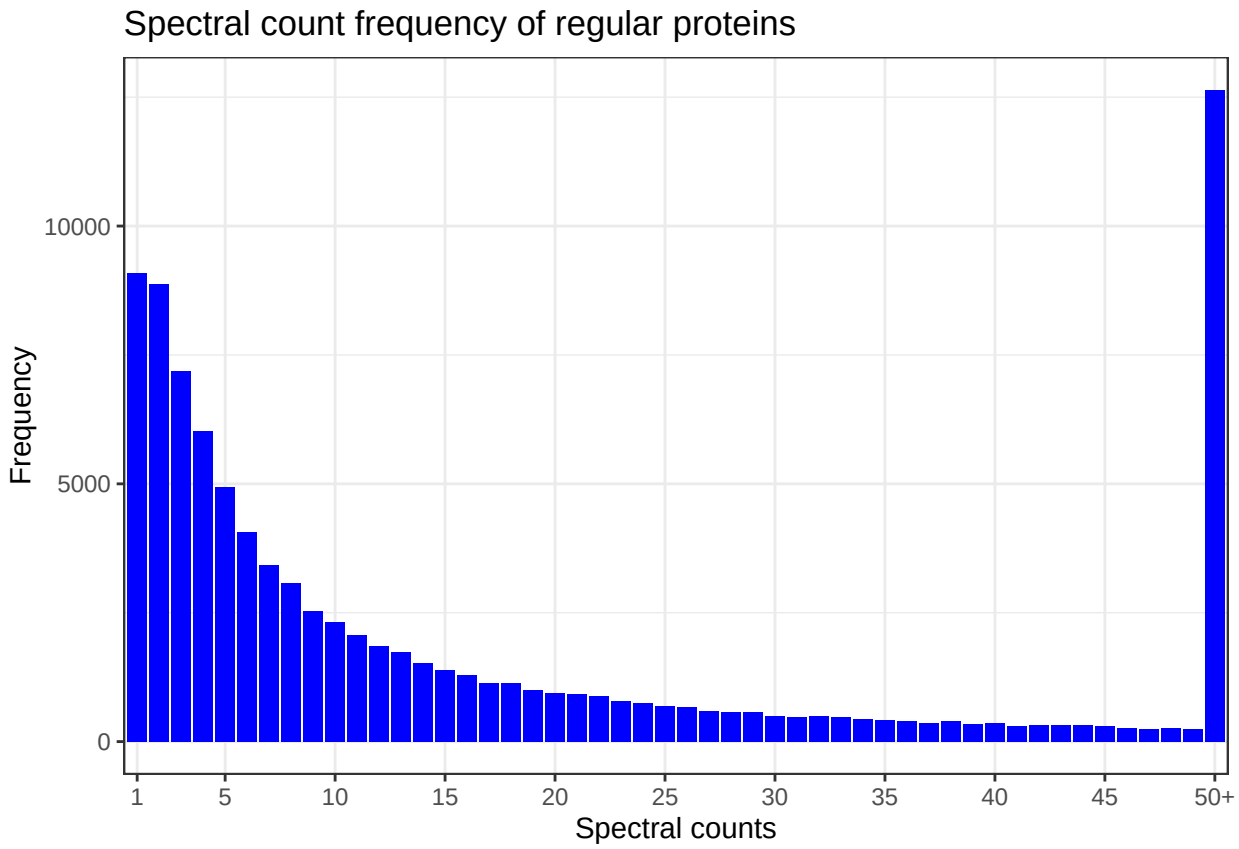
```
# IP0981_1701 IP0982_1701 IP0993_1701 IP0995_1701 IP0999_1701 IP7100_1701 IP7103_1701 IP7105_1701 sampl
# rename sample to Sample1-8
colnames(gastric_count) <- c('accession', paste('Sample', 1:8, sep = ' '))

# create long format used for barplot
MP_counts <- gastric_count %>% filter(accession %in% missing_proteins_accession)
MP_counts_long <- MP_counts %>% pivot_longer(names_to = 'IPAS', values_to = 'count', cols = c(-accessio
MP_counts_long$type <- 'Missing'

RP_counts <-  gastric_count %>% filter(!(accession %in% P2_5_proteins_accession))
RP_counts_long <- RP_counts %>% pivot_longer(names_to = 'IPAS', values_to = 'count', cols = c(-accessio
RP_counts_long$type <- 'Regular'

all_counts_long <- rbind(MP_counts_long, RP_counts_long)
```

Figure S4 The peptide-spectral-matching (PSM) count distribution for regular proteins (Left blue) and missing proteins (Right red)

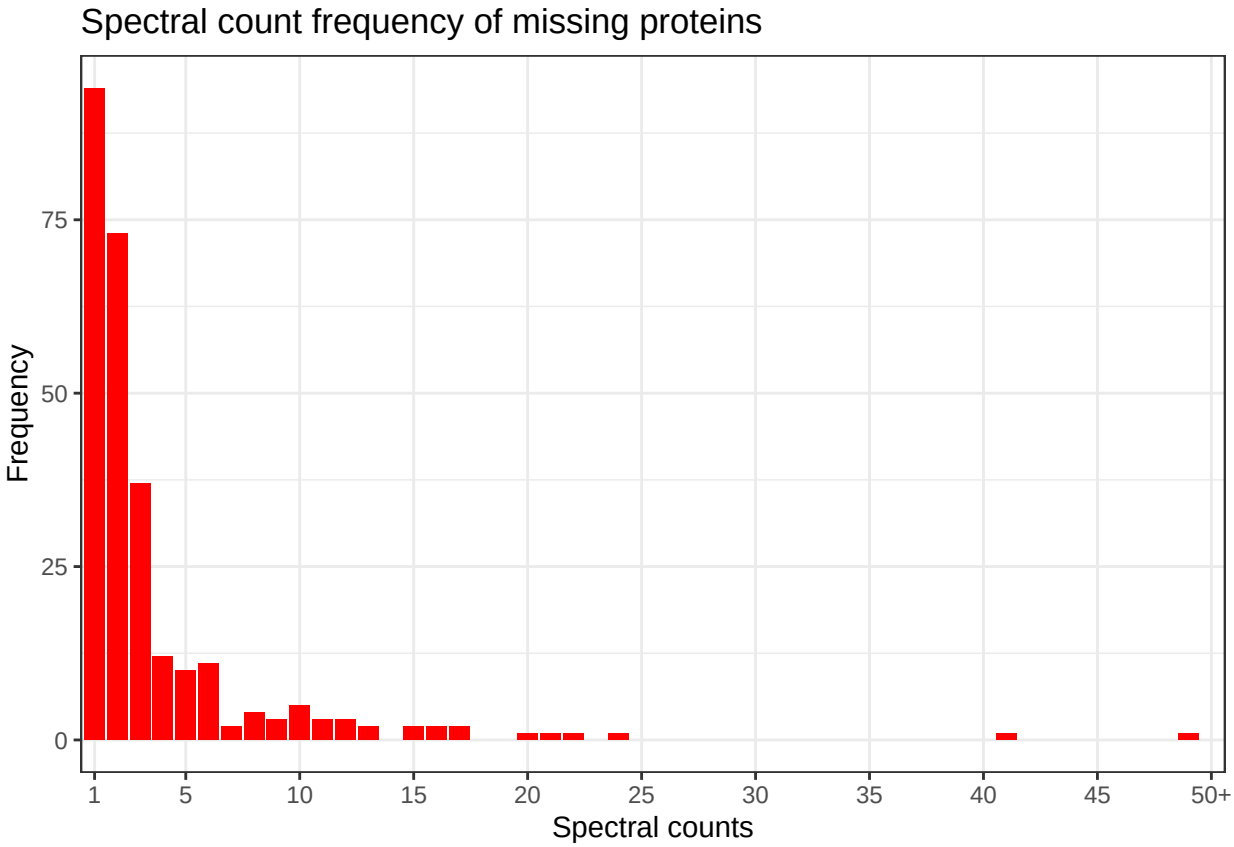
Frequency of Spectral count for regular proteins

```
# collapse SpC >= 50 into one single category "50+"
RP_barplot_data <-  RP_counts_long %>% mutate(count_discrete=ifelse(count >= 50, '50+', count))
RP_barplot_data$count_discrete <- factor(RP_barplot_data$count_discrete, levels=c(as.character(1:49), '
RP_barplot_data %>%
  ggplot(aes(x=count_discrete)) +
  geom_bar(position = 'identity', fill='blue') +
theme_bw() +
  xlab('Spectral counts') +
  ylab('Frequency') +
```

```
  ggtitle('Spectral count frequency of regular proteins') +
  scale_x_discrete(breaks=c('1', seq(5, 50, 5), '50+'))
```




```
MP_barplot_data <-  MP_counts_long %>% mutate(count_discrete=ifelse(count >= 50, '50+', count))
MP_barplot_data$count_discrete <- factor(MP_barplot_data$count_discrete, levels=c(as.character(1:49), '5
MP_barplot_data %>%
  ggplot(aes(x=count_discrete)) +
  geom_bar(position = 'identity', fill='red') +
theme_bw() +
  xlab('Spectral counts') +
  ylab('Frequency') +
  ggtitle('Spectral count frequency of missing proteins') +
  scale_x_discrete(breaks=c('1', seq(5, 50, 5), '50+'), drop=FALSE)
```



Aggregate proteins expression into gene-level

The oirginal proteomics data is in accession level and the original transcription data is in gene level. To match the two data, we need to aggregate protein expression at the gene level. Typically, one gene can be transcripted and translated into multiple proteins and a protein can be translated only from a single gene. So the mapping from accession to gene symbol is unique. After mapping, there can be rows with the same gene symbol ID, then those row with the same gene symbol ID were summed into a single row to represent the expression of proteins translated from that one gene.

create a dictionary used for mapping ensembl_gene_id to gene symbol

When a same ensembl_gene_id can be map to multiple gene symbol (not common) use only the first one (ordered by alphabeta)

```
dict_accession_to_gene_symbol <- omics_metadata %>% dplyr::select(accession_id, gene_symbol) %>% unique
```

Aggregate proteins into gene-level by summing rows with the same row indices

```
gastric_nsaf_gene_level <- gastric_nsaf
rownames(gastric_nsaf_gene_level) <- mapping(rownames(gastric_nsaf), dict_accession_to_gene_symbol)

# gastric_nsaf_gene_level <- rownames_mapping(gastric_nsaf, omics_metadata, map_from = 'accession_id', m
```

```
# use sum or average? ?? sum seems to make more sense
temp <- as_tibble(gastric_nsaf_gene_level, rownames='symbol') %>%  filter(!is.na(symbol))  %>% group_by
gastric_nsaf_gene_level <- as.matrix(temp[-1])
rownames(gastric_nsaf_gene_level) <- temp$symbol

# remove rows with NA index
gastric_nsaf_gene_level <- gastric_nsaf_gene_level[!is.na(rownames(gastric_nsaf_gene_level)), ]

head(gastric_nsaf_gene_level)
```

```
##         IP0981_1701 IP0982_1701 IP0993_1701 IP0995_1701 IP0999_1701 IP7100_1701
## A1BG        0.00000    370.7916    67.06595      0.0000      0.0000      0.0000
## A1CF        0.00000      0.0000     0.00000      0.0000      0.0000      0.0000
## A2M      1430.79079   7434.2917  4527.20623      0.0000    798.0217   2428.3120
## A3GALT2    96.60582      0.0000     0.00000      0.0000      0.0000      0.0000
## AAAS        0.00000    740.3128   891.04499    707.4861    280.7749    184.7237
## AACS        0.00000      0.0000   260.14970    939.9579      0.0000    289.7341
##         IP7103_1701 IP7105_1701
## A1BG         0.0000    355.9040
## A1CF         0.0000    113.4642
## A2M         79.8145    625.6049
## A3GALT2      0.0000      0.0000
## AAAS       464.1019    887.1259
## AACS         0.0000    626.7537
```

```
nrow(gastric_nsaf_gene_level)
```

```
## [1] 10665
```

Transform RNA expression data into gene symbol level

The original RNA data is indexed by ensembl_gene_id (ENSG), to used it with proteomics data at gene symbol level, we need to mapping ensembl_gene_id to gene symbol.

load transcription data

! avoid use load in future

```
#
load('cache/count_TPM.rda')
```

create a dictionary used for mapping ensembl_gene_id to gene symbol

When a same ensembl_gene_id can be map to multiple gene symbol (not common) use only the first one (ordered by alphabeta)

```
dict_ensembl_to_gen_symbol <- omics_metadata %>% select(ensembl_gene_id, gene_symbol) %>% unique()
dict_ensembl_to_gen_symbol <- dict_ensembl_to_gen_symbol %>% dplyr::group_by(ensembl_gene_id) %>%
      dplyr::arrange(gene_symbol) %>%
      dplyr::summarise(map_to=first(gene_symbol))
```

mapping ensembl_gene_id to gene symbol and pre-process data

1. Remove row with 0 in all cells
2. Only keep the 8 gastric primary cell samples
3. aggregate row with the same index by summing across row axis

```
count_TPM_gene_symbol <- count_TPM_mat
rownames(count_TPM_gene_symbol) <- mapping(rownames(count_TPM_mat), dictionary = dict_ensembl_to_gen_sy
# count_TPM_gene_symbol

# count_TPM_gene_symbol <- rownames_mapping(count_TPM_mat, omics_metadata, map_from = 'ensembl_gene_id'
count_TPM_gene_symbol <- count_TPM_gene_symbol[rowSums(count_TPM_gene_symbol > 0) > 0, ]

temp <- as_tibble(count_TPM_gene_symbol, rownames='symbol') %>% filter(!is.na(symbol)) %>% group_by(sym
count_TPM_gene_symbol <- as.matrix(temp[-1])
rownames(count_TPM_gene_symbol) <- temp$symbol
# change the column name the same as in spectral count matrix
colnames(count_TPM_gene_symbol) <- c('IP7103_1701', 'IP0972_1701', 'IP0995_1701', 'IP0982_1701', 'IP099
count_TPM_gene_symbol <- count_TPM_gene_symbol[, colnames(gastric_nsaf_gene_level)]

# remove row with NA index
count_TPM_gene_symbol <- count_TPM_gene_symbol[!is.na(rownames(count_TPM_gene_symbol)), ]

head(count_TPM_gene_symbol)
```

```
##          IP0981_1701 IP0982_1701 IP0993_1701  IP0995_1701 IP0999_1701
## A1BG      0.12136630   0.9714666   1.8962540   8.06389727   0.8891093
## A1BG-AS1  0.69555945   1.3701390   8.9431089 137.05080510   5.6369555
## A1CF      3.60652496   5.0558246   7.3000066   0.00000000   6.4181129
## A2M       6.77333151   3.2430162 605.7338518   6.07730643  34.4754551
## A2M-AS1   0.85851339   3.4359501  10.6812031   2.84116724  18.0294027
## A2ML1     0.02089757   0.0000000   0.2448813   0.04787897   0.4248308
##           IP7100_1701 IP7103_1701  IP7105_1701
## A1BG       2.00332441   2.5125801   3.39680657
## A1BG-AS1  44.92645048   7.1998934   7.00430202
## A1CF       1.54458435   0.0000000   6.29969386
## A2M      676.87508306   0.2867578 524.78950976
## A2M-AS1   24.64520330  27.6474070   2.16469545
## A2ML1      0.01124819   0.1081578   0.02371145
```

```
nrow(count_TPM_gene_symbol)
```

```
## [1] 31742
```

```
# write.csv(as.data.frame(count_TPM_gene_symbol), file='TPM_gastric_cancer_primary_cell.csv')
```

rename to sample 1- 8

```
# colnames(gastric_nsaf_gene_level) <- paste('Sample', 1:8, sep = ' ')
# colnames(count_TPM_gene_symbol) <- paste('Sample', 1:8, sep = ' ')
```

check the overall covarage of RNA and protein for each other

```
# percentage of gene that can be found to have protein product
mean(rownames(count_TPM_gene_symbol) %in% rownames(gastric_nsaf_gene_level))
```

```
## [1] 0.3141264
```

```
# percentage of proteins that have transcription level evidence
mean(rownames(gastric_nsaf_gene_level) %in% rownames(count_TPM_gene_symbol))
```

```
## [1] 0.9349273
```

combine and match RNA and proteomics data into long format

Each row is a protein identified by gene symbol and sample ID primary key: (accession, IPAS) in term of SQL TPM is the RNA expression of the gene that coding those proteins NSAF is the combined protein expressions for proteomics translated from that gene

```
A <- as_tibble(log(count_TPM_gene_symbol+1), rownames='accession') %>% pivot_longer(names_to = 'IPAS', 
B <- as_tibble(log(gastric_nsaf_gene_level+1), rownames='accession') %>% pivot_longer(names_to = 'IPAS'
combined_long <- full_join(A, B, by=c('accession', 'IPAS')) %>% replace(is.na(.), 0) %>% filter(!(TPM ==
head(combined_long)
```

```
## # A tibble: 6 x 4
##   accession IPAS          TPM  NSAF
##   <chr>     <chr>       <dbl> <dbl>
## 1 A1BG      IP0981_1701 0.115  0
## 2 A1BG      IP0982_1701 0.679  5.92
## 3 A1BG      IP0993_1701 1.06   4.22
## 4 A1BG      IP0995_1701 2.20   0
## 5 A1BG      IP0999_1701 0.636  0
## 6 A1BG      IP7100_1701 1.10   0
```

Create a helper function for calculate the number of proteins-at-gene-level has non-zero RNA expression

```
#' @title calculate proportion of protein products supported with RNA products
#' @description It product the proportion of each condition based on different settings
#' @param tbl a tibble object, see details in "dplyr"
#' @param missing_protein whether to only include missing proteins
#' @param group_by_IPAS whether to stratified by IPAS samples
#' @param TPM_cutoff float, the cutoff used for RNA TPM value to consider a RNA product to be existing
#' @param NSAF_cutoff float, the cutoff used for RNA TPM value to consider a protein product to be exis
#' @param cases a vector of integer, used to decide whether cases to included for calculation
summarize_by_case <- function(tbl, proteins_subset, negate=FALSE, remove_IPAS=NULL, group_by_IPAS=FALSE

  if(negate) {
    tbl <- tbl %>% filter(!(accession %in% proteins_subset))
  } else{
    tbl <- tbl %>% filter(accession %in% proteins_subset)
  }


  if (!is.null(remove_IPAS)) {
    tbl <- tbl %>% filter(!(IPAS %in% remove_IPAS))
  }

  tbl <- tbl %>% rowwise() %>%
    mutate(case=if_else(TPM > TPM_cutoff & NSAF > NSAF_cutoff, 1,
                        if_else(TPM <= TPM_cutoff & NSAF > NSAF_cutoff, 2,
                                if_else(TPM > TPM_cutoff & NSAF <= NSAF_cutoff, 3, 4)
                        )
    )
    ) %>%
    filter(case %in% cases) %>%
    group_by(IPAS, case) %>%
    summarise(count=n(), ) %>%
    ungroup(IPAS, case)

  if (group_by_IPAS) {
    tbl <- tbl %>% group_by(IPAS)
  }

  tbl <- tbl %>%
    group_by(case, .add=TRUE) %>%
    summarise(count= sum(count)) %>%
    mutate(countT=sum(count)) %>%
    mutate(per=paste0(round(100*count/countT,2),'%'))

  return(tbl)
}
```

the proportion of PE2-5 in the missing protein dataset

```
table(missing_protein_df$PE)
```

```
##
## Evidence at transcript level       Inferred from homology
##                         1135                          195
##                    Predicted                    Uncertain
##                           13                          609
```

check missing proteins with corresponding RNA expression

```
# check the protein and RNA expression for the detected missing proteins
full_table_detected_MP <- combined_long %>% filter(accession %in% missing_proteins) %>% filter(NSAF > 0

# write_csv(full_table_detected_MP, 'output/protein-list/detected_119_MP_products_with_RNA.csv')
```

association analysis

Table 2 The proportions of protein products with mRNA products for missing proteins and regular proteins

Table 2 missing protein part

```
# the coverage rate of RNA product in protein product for missing protein
(a <- summarize_by_case(combined_long, proteins_subset=missing_proteins, group_by_IPAS = TRUE, cases = c
```

```
## `summarise()` has grouped output by 'IPAS'. You can override using the
## `.groups` argument.
## `summarise()` has grouped output by 'IPAS'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 16 x 5
## # Groups:   IPAS [8]
##    IPAS         case count countT per
##    <chr>       <dbl> <int>  <int> <chr>
##  1 IP0981_1701     1     5     10 50%
##  2 IP0981_1701     2     5     10 50%
##  3 IP0982_1701     1    34     49 69.39%
##  4 IP0982_1701     2    15     49 30.61%
##  5 IP0993_1701     1    18     31 58.06%
##  6 IP0993_1701     2    13     31 41.94%
##  7 IP0995_1701     1    14     28 50%
##  8 IP0995_1701     2    14     28 50%
##  9 IP0999_1701     1    14     23 60.87%
```

```
## 10 IP0999_1701     2     9     23 39.13%
## 11 IP7100_1701     1    14     19 73.68%
## 12 IP7100_1701     2     5     19 26.32%
## 13 IP7103_1701     1     3     17 17.65%
## 14 IP7103_1701     2    14     17 82.35%
## 15 IP7105_1701     1    17     27 62.96%
## 16 IP7105_1701     2    10     27 37.04%
```

```
# write_csv(a, file='output/sample_MP_stats.csv')

## overall without removing the outlier
(b <- summarize_by_case(combined_long, proteins_subset=missing_proteins, group_by_IPAS = FALSE, cases =
```

```
## `summarise()` has grouped output by 'IPAS'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 2 x 4
##    case count countT per
##   <dbl> <int>  <int> <chr>
## 1     1   119    204 58.33%
## 2     2    85    204 41.67%
```

```
## overall with removing the outlier
(c <- summarize_by_case(combined_long, proteins_subset=missing_proteins, group_by_IPAS = FALSE, cases =
```

```
## `summarise()` has grouped output by 'IPAS'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 2 x 4
##    case count countT per
##   <dbl> <int>  <int> <chr>
## 1     1   111    177 62.71%
## 2     2    66    177 37.29%
```

```
# the coverage rate of protein product in RNA product for missing protein
summarize_by_case(combined_long, proteins_subset=missing_proteins, group_by_IPAS = TRUE, cases = c(1, 3
```

```
## `summarise()` has grouped output by 'IPAS'. You can override using the
## `.groups` argument.
## `summarise()` has grouped output by 'IPAS'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 16 x 5
## # Groups:   IPAS [8]
##    IPAS         case count countT per
##    <chr>       <dbl> <int>  <int> <chr>
##  1 IP0981_1701     1     5    418 1.2%
##  2 IP0981_1701     3   413    418 98.8%
##  3 IP0982_1701     1    34    422 8.06%
##  4 IP0982_1701     3   388    422 91.94%
##  5 IP0993_1701     1    18    473 3.81%
```

```
##  6 IP0993_1701     3   455    473 96.19%
##  7 IP0995_1701     1    14    388 3.61%
##  8 IP0995_1701     3   374    388 96.39%
##  9 IP0999_1701     1    14    527 2.66%
## 10 IP0999_1701     3   513    527 97.34%
## 11 IP7100_1701     1    14    497 2.82%
## 12 IP7100_1701     3   483    497 97.18%
## 13 IP7103_1701     1     3    298 1.01%
## 14 IP7103_1701     3   295    298 98.99%
## 15 IP7105_1701     1    17    469 3.62%
## 16 IP7105_1701     3   452    469 96.38%
```

```
## overall without removing the outlier
summarize_by_case(combined_long, proteins_subset=missing_proteins, group_by_IPAS = FALSE, cases = c(1, 3
```

```
## `summarise()` has grouped output by 'IPAS'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 2 x 4
##    case count countT per
##   <dbl> <int>  <int> <chr>
## 1     1   119   3492 3.41%
## 2     3  3373   3492 96.59%
```

```
## overall with removing the outlier
summarize_by_case(combined_long, proteins_subset=missing_proteins, group_by_IPAS = FALSE,  cases = c(1,
```

```
## `summarise()` has grouped output by 'IPAS'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 2 x 4
##    case count countT per
##   <dbl> <int>  <int> <chr>
## 1     1   111   2776 4%
## 2     3  2665   2776 96%
```

histogram

```
b$IPAS = 'Combined'
c$IPAS = 'Combined w.o. outlier'
dat_hist <- rbind(a, b, c)


dat_hist$case <- factor(dat_hist$case, levels = c('1', '2'), labels = c('RNA (+)', 'RNA (-)'))
dat_hist$case <- relevel(dat_hist$case, 'RNA (+)')

dat_hist$IPAS <- factor(dat_hist$IPAS, levels = c("IP0981_1701", "IP0982_1701", "IP0993_1701", "IP0995_1
                                                  "IP7100_1701", "IP7103_1701", "IP7105_1701", "Combine


ggplot(data=dat_hist, aes(x=IPAS, y=count, group=case, fill=case)) +
```

```
  labs(title="Coverage rate for prodein product with RNA product in missing proteins", subtitle = '',
       x = "IPAS", y = 'count') +
  geom_bar( stat = 'identity',  position=position_dodge()) +
  geom_text(aes(label=per), color="black",size=3, vjust=-0.5, position = position_dodge(1)) +
  theme(
       axis.text.x = element_text(angle = 45, hjust = 1))
```

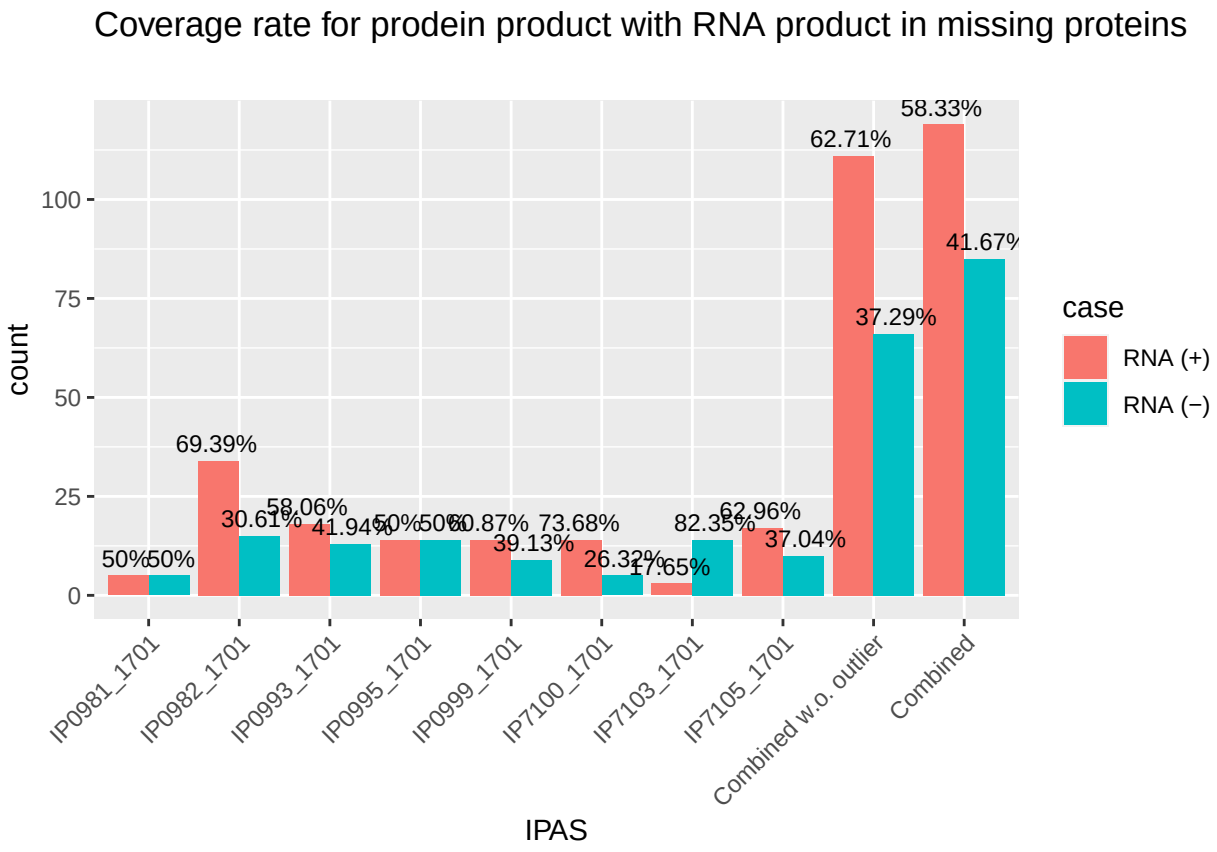


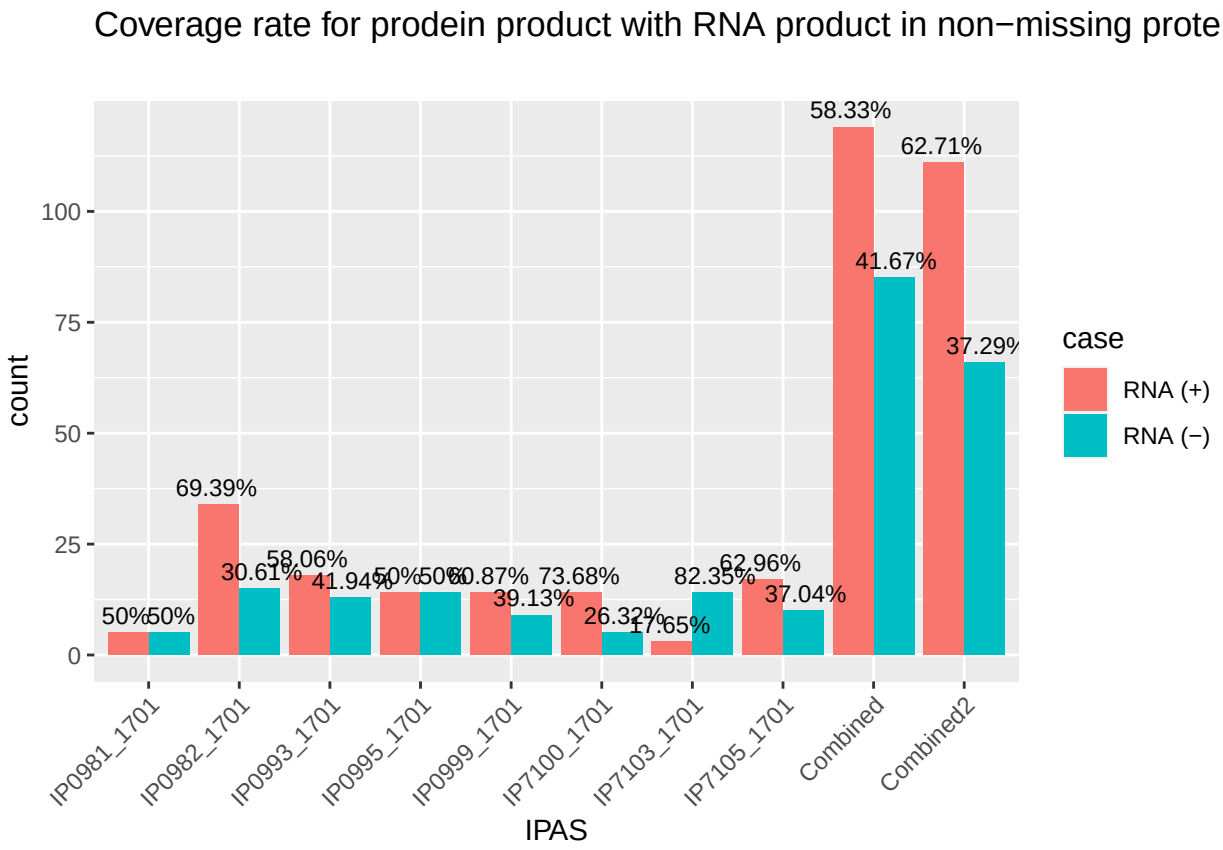
Table 2 non-missing proteins part

The execution of the following chuck of code could be slow for the size of non-missing proteins

barplot for non-missing proteins

```
b$IPAS = 'Combined'
c$IPAS = 'Combined2'
dat_hist <- rbind(a, b, c)

dat_hist$case <- factor(dat_hist$case, levels = c('1', '2'), labels = c('RNA (+)', 'RNA (-)'))
dat_hist$case <- relevel(dat_hist$case, 'RNA (+)')

dat_hist$IPAS <- factor(dat_hist$IPAS, levels = c("IP0981_1701", "IP0982_1701", "IP0993_1701", "IP0995_
                                                  "IP7100_1701", "IP7103_1701", "IP7105_1701", "Combine
ggplot(data=dat_hist, aes(x=IPAS, y=count, group=case, fill=case)) +
```

```
  labs(title="Coverage rate for prodein product with RNA product in non-missing proteins", subtitle = '
       x = "IPAS", y = 'count') +
  geom_bar( stat = 'identity',  position=position_dodge()) +
  geom_text(aes(label=per), color="black",size=3, vjust=-0.5, position = position_dodge(1)) +
  theme(
       axis.text.x = element_text(angle = 45, hjust = 1))
```



Data preparation for Figure 3: The scatter plot of protein-mRNA-product matched pairs.

generate data that can be used for plot the scatter plot

Four datasets were created: 1) glmm_dat_missing: paired RNA/protein expressions for missing protein 2) glmm_dat_missing_filtered: pair RNA/protein expressions for missing protein and only keep proteins with both valid (>0) RNA and protein expressions 3) glmm_dat_non_missing: paired RNA/protein expressions for regular protein 4) glmm_dat_non_missing_filtered: air RNA/protein expressions for regular proteins and only keep proteins with both valid (>0) RNA and protein expressions

All TPM/NSAF values were log-transformed in all four datasets

```
count_nsaf_missing <-  gastric_nsaf_gene_level[rownames(gastric_nsaf_gene_level) %in% missing_proteins,
count_TPM_missing <- count_TPM_gene_symbol[rownames(count_TPM_gene_symbol) %in% missing_proteins, ]
count_nsaf_non_missing <-  gastric_nsaf_gene_level[!(rownames(gastric_nsaf_gene_level) %in% P2_5_protei
count_TPM_non_missing <- count_TPM_gene_symbol[!(rownames(count_TPM_gene_symbol) %in% P2_5_proteins), ]
```

```
A <- as_tibble(log(count_TPM_missing+1), rownames='accession') %>% pivot_longer(names_to = 'IPAS', value
B <- as_tibble(log(count_nsaf_missing+1), rownames='accession') %>% pivot_longer(names_to = 'IPAS', valu
glmm_dat_missing <- full_join(A, B, by=c('accession', 'IPAS')) %>% replace(is.na(.), 0) %>% filter(!(TPM
glmm_dat_missing_filtered <- glmm_dat_missing %>% filter(TPM > 0 & NSAF > 0)

A <- as_tibble(log(count_TPM_non_missing+1), rownames='accession') %>% pivot_longer(names_to = 'IPAS', v
B <- as_tibble(log(count_nsaf_non_missing+1), rownames='accession') %>% pivot_longer(names_to = 'IPAS',
glmm_dat_non_missing <- full_join(A, B, by=c('accession', 'IPAS')) %>% replace(is.na(.), 0) %>% filter(
glmm_dat_non_missing_filtered <- glmm_dat_non_missing %>% filter(TPM > 0 & NSAF > 0)
```

Checking Sample heterogeneity

Check whether RNA expression, protein expression, and protein-RNA-expression relationship were similar between samples

Generalization linear mixed model using RNA expression as random effects in different samples

check whether the correlation relationship is different in different samples

```
library(lme4)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
glmm_model <- lmer(NSAF ~ TPM + (0 + TPM | IPAS), data = glmm_dat_missing_filtered)
glm_model <- lm(NSAF ~ TPM, data = glmm_dat_missing_filtered)
summary(glmm_model)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: NSAF ~ TPM + (0 + TPM | IPAS)
##    Data: glmm_dat_missing_filtered
##
## REML criterion at convergence: 320.8
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.52903 -0.60143  0.06817  0.55968  2.42894
##
## Random effects:
##  Groups   Name Variance Std.Dev.
##  IPAS     TPM  0.0227   0.1507
```

```
##  Residual      0.7933   0.8907
## Number of obs: 119, groups:  IPAS, 8
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  4.58008    0.10771  42.522
## TPM          0.15109    0.08304   1.819
##
## Correlation of Fixed Effects:
##     (Intr)
## TPM -0.475
```

No significant evidence for support using RNA expression as random effects

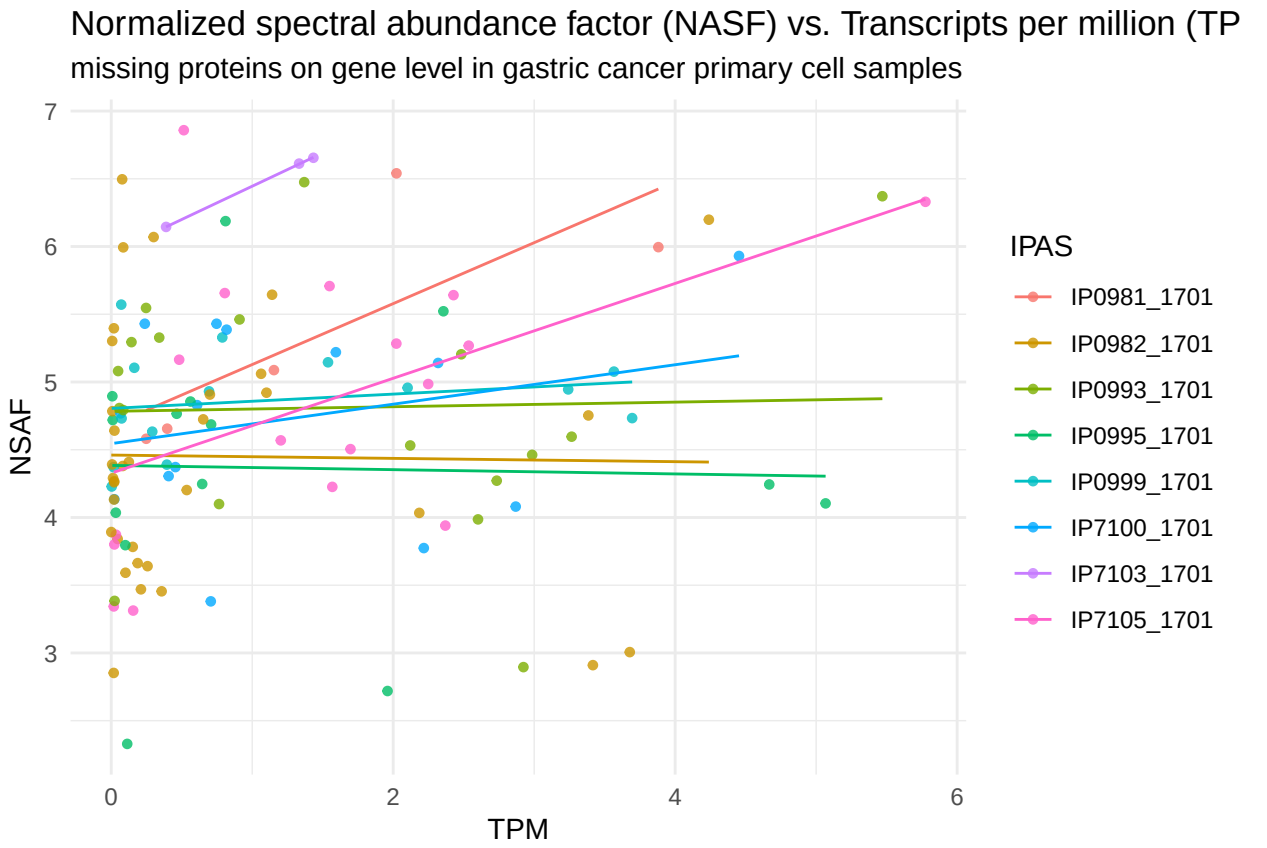
```
anova(glmm_model, glm_model)
```

```
## refitting model(s) with ML (instead of REML)

## Data: glmm_dat_missing_filtered
## Models:
## glm_model: NSAF ~ TPM
## glmm_model: NSAF ~ TPM + (0 + TPM | IPAS)
##            npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## glm_model     3 321.28 329.62 -157.64   315.29
## glmm_model    4 322.57 333.68 -157.28   314.57 0.7165  1     0.3973
```

The scatterplot with group-wise linear regression line validates the result

```
ggplot(data      = glmm_dat_missing_filtered,
       aes(x     = TPM,
           y     = NSAF,
           col   = IPAS,
           group = IPAS))+ #to add the colours for different classes
  geom_point(size     = 1.2,
             alpha    = .8,
             position = "jitter")+ #to add some random noise for plotting purposes
  theme_minimal() +
  geom_smooth(method = lm,
              se     = FALSE,
              size   = .5,
              alpha  = .8)+ # to add regression line
  labs(title    = "Normalized spectral abundance factor (NASF) vs. Transcripts per million (TPM)",
       subtitle = "missing proteins on gene level in gastric cancer primary cell samples",
       xlab = 'Log2(TPM) + 1',
       ylab = 'Log2(NSAF) + 1',
  )
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
model.lm <- lm(NSAF ~ TPM, data = glmm_dat_missing_filtered)
summary(model.lm)
```

```
##
## Call:
## lm(formula = NSAF ~ TPM, data = glmm_dat_missing_filtered)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2749 -0.5685  0.0478  0.6581  2.2099
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.59078    0.11072  41.462   <2e-16 ***
## TPM          0.11238    0.06085   1.847   0.0673 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9178 on 117 degrees of freedom
## Multiple R-squared:  0.02833,    Adjusted R-squared:  0.02003
## F-statistic: 3.411 on 1 and 117 DF,  p-value: 0.06727
```

correlation plot for pairwise correlation of missing protein expression

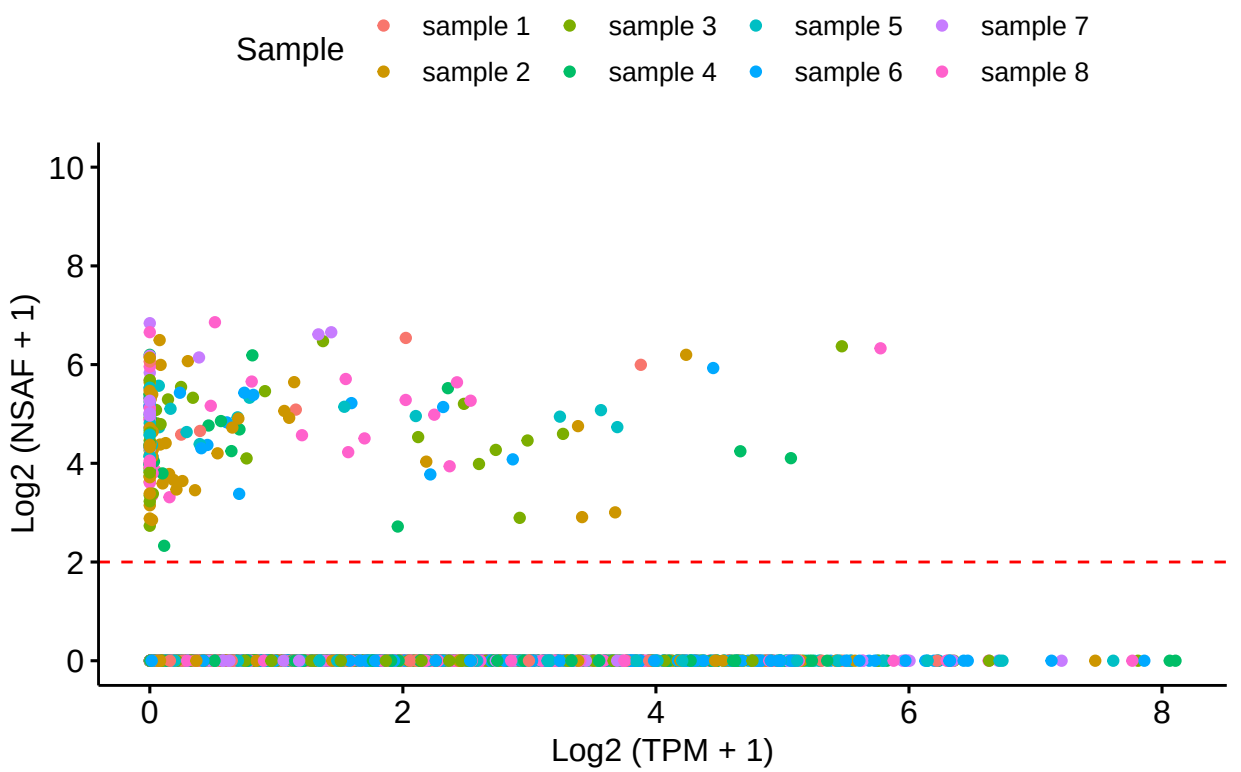
results of the two correlation plot can be found in the end of the this markdown file

correlation plot for pairwise correlation of missing protein gene expression

Figure 3 with spearman correlation for missing proteins

The scatter plot of protein-mRNA-product matched pairs. (a): All protein-mRNA pairs for missing proteins. (b): protein-mRNA pairs with TPM > 0 and NSAF > 0. (c): All protein-mRNA pairs for regular proteins. (d): All protein-mRNA pairs for regular proteins with TPM > 0 and NSAF > 0. Blue line: Simple linear regression. Spearman correlations and corresponding p-value were calculated for (b) and (d).

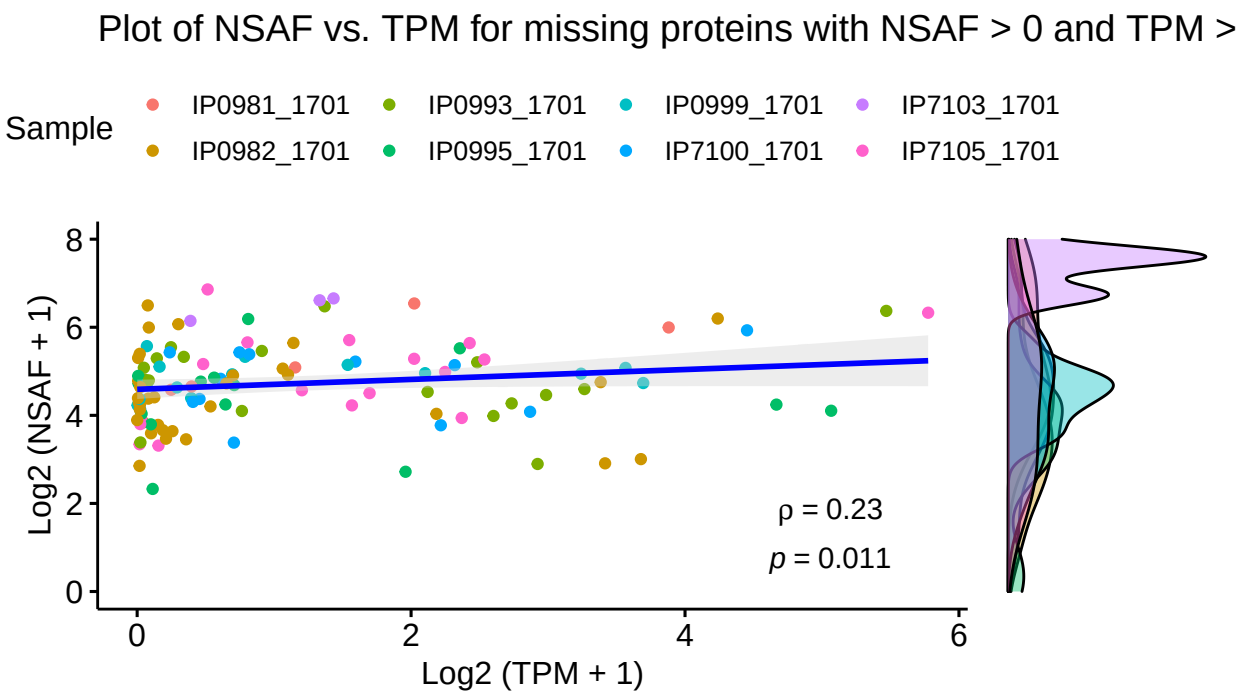
```
par(mfrow=c(2, 2))
ggpubr::ggscatter(glmm_dat_missing, x = "TPM", y = "NSAF",
   color = 'IPAS', size = 1.5, # Points color, shape and size
   title='Plot of NSAF vs. TPM for missing proteins',
   xlab = 'Log2 (TPM + 1)',
   ylab = 'Log2 (NSAF + 1)',
   ) +
  scale_color_discrete(labels=paste('sample', 1:8)) +
  scale_y_continuous(name='Log2 (NSAF + 1)', breaks=c(0, 2, 4, 6, 8, 10), limits = c(0, 10)) +
  # geom_vline(xintercept=0.1, linetype=2, color='red', size=0.5) +
  geom_hline(yintercept=2, linetype=2, color='red', size=0.5) +
  theme(legend.text = element_text(size=10)) +
  labs(color='Sample')
```



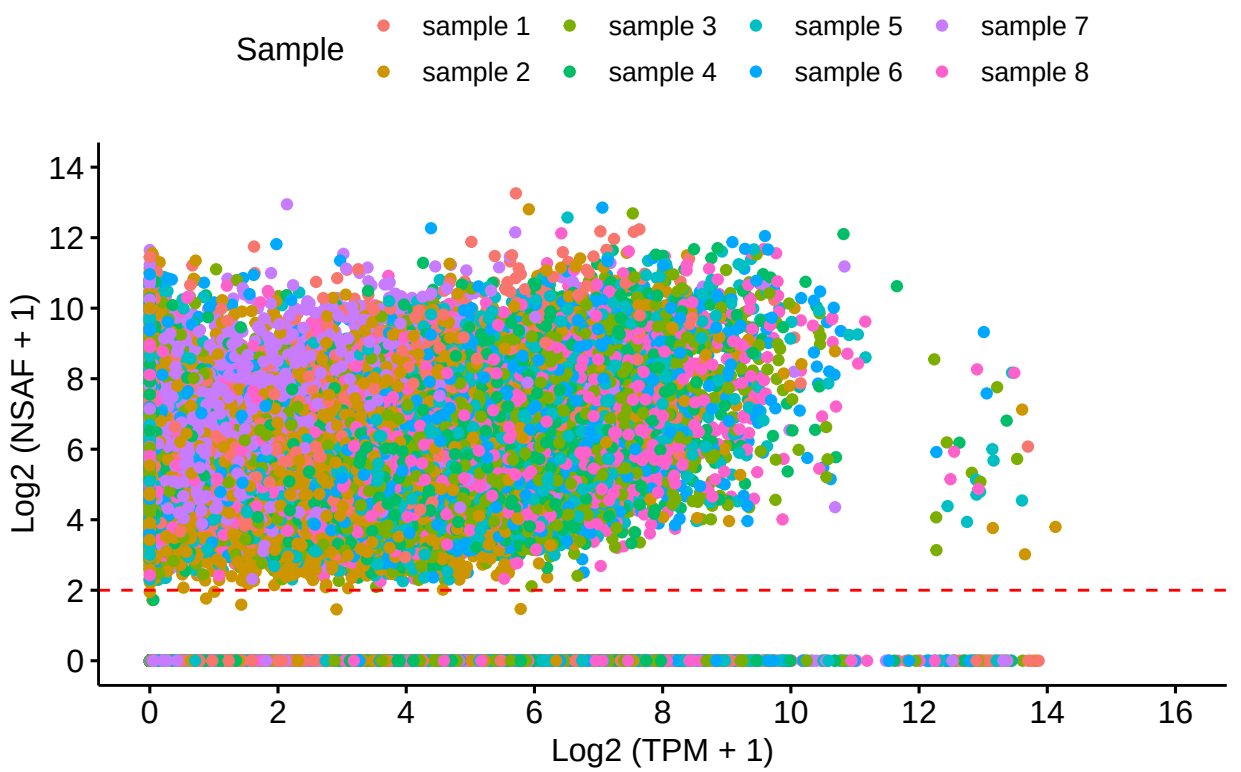

```
# plot with spearman correlation for missing proteins
plt <- ggpubr::ggscatter(glmm_dat_missing_filtered, x = "TPM", y = "NSAF",
   color = 'IPAS', size = 1.5, # Points color, shape and size
   add = "reg.line",  # Add regressin line
   add.params = list(color = "blue", fill = "lightgray"), # Customize reg. line
   conf.int = TRUE, # Add confidence interval
   cor.coef = TRUE, # Add correlation coefficient. see ?stat_cor
   cor.coeff.args = list(method = "spearman", cor.coef.name = "rho", label.x.npc=0.8, label.sep = "\n",
                         label.y.npc = 0.2),
   title='Plot of NSAF vs. TPM for missing proteins with NSAF > 0 and TPM > 0',
   # subtitle='missing proteins on gene level in gastric cancer primary cell sample',
   xlab = 'Log2 (TPM + 1)',
   ylab = 'Log2 (NSAF + 1)',
   ) +
   scale_fill_discrete(labels=paste('sample', 1:8)) +
   scale_y_continuous(name='Log2 (NSAF + 1)', breaks=c(0, 2, 4, 6, 8, 10), limits = c(0, 8)) +
   theme(legend.text = element_text(size=10)) +
   labs(color='Sample')

dens1 <- ggplot(glmm_dat_missing_filtered, aes(x = TPM, fill = IPAS)) +
  geom_density(alpha = 0.4) +
  theme_void() +
  theme(legend.position = "none")

dens2 <- ggplot(glmm_dat_missing_filtered, aes(x = NSAF, fill = IPAS)) +
  geom_density(alpha = 0.4) +
  theme_void() +
```

```
  theme(legend.position = "none") +
  coord_flip()

dens1 + patchwork::plot_spacer() + plt + dens2 +  patchwork::plot_layout(ncol = 2, nrow = 2, widths = c
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
# plot with spearman correlation for non-missing proteins
ggpubr::ggscatter(glmm_dat_non_missing, x = "TPM", y = "NSAF",
   color = 'IPAS', size = 1.5, # Points color, shape and size
   title='Plot of NSAF vs. TPM for regular proteins',
   xlab = 'Log2 (TPM + 1)',
   ylab = 'Log2 (NSAF + 1)') +
  scale_color_discrete(labels=paste('sample', 1:8)) +
   scale_x_continuous(name='Log2 (TPM + 1)', breaks=seq(0, 16, 2), limits = c(0, 16)) +
   scale_y_continuous(name='Log2 (NSAF + 1)', breaks=seq(0, 14, 2), limits = c(0, 14)) +
   theme(legend.text = element_text(size=10)) +
  # geom_vline(xintercept=0.1, linetype=2, color='red', size=0.5) +
  geom_hline(yintercept=2, linetype=2, color='red', size=0.5) +
  labs(color='Sample')
```




```
plt <- ggpubr::ggscatter(glmm_dat_non_missing_filtered, x = "TPM", y = "NSAF",
   color = 'IPAS', size = 1.5, # Points color, shape and size
   add = "reg.line",  # Add regressin line
   add.params = list(color = "blue", fill = "lightgray"), # Customize reg. line
   conf.int = TRUE, # Add confidence interval
   cor.coef = TRUE, # Add correlation coefficient. see ?stat_cor
   cor.coeff.args = list(method = "spearman", cor.coef.name = "rho",  label.x.npc=0.85, label.sep = "\n
                         label.y.npc = 0.2),
   title='Plot of NSAF vs. TPM for regular proteins with NSAF > 0 and TPM > 0',
   # subtitle='missing proteins on gene level in gastric cancer primary cell sample',
   xlab = 'Log2 (TPM + 1)',
   ylab = 'Log2 (NSAF + 1)') +
  scale_color_discrete(labels=paste('sample', 1:8)) +
   scale_x_continuous(name='Log2 (TPM + 1)', breaks=seq(0, 16, 2), limits = c(0, 16)) +
   scale_y_continuous(name='Log2 (NSAF + 1)', breaks=seq(0, 14, 2), limits = c(0, 14)) +
    theme(legend.text = element_text(size=10)) +
  labs(color='Sample')

dens1 <- ggplot(glmm_dat_non_missing_filtered, aes(x = TPM, fill = IPAS)) +
  geom_density(alpha = 0.4) +
  theme_void() +
  theme(legend.position = "none")

dens2 <- ggplot(glmm_dat_non_missing_filtered, aes(x = NSAF, fill = IPAS)) +
  geom_density(alpha = 0.4) +
  theme_void() +
  theme(legend.position = "none") +
```

```
  coord_flip()

dens1 + patchwork::plot_spacer() + plt + dens2 +  patchwork::plot_layout(ncol = 2, nrow = 2, widths = c
```

```
## `geom_smooth()` using formula 'y ~ x'
```

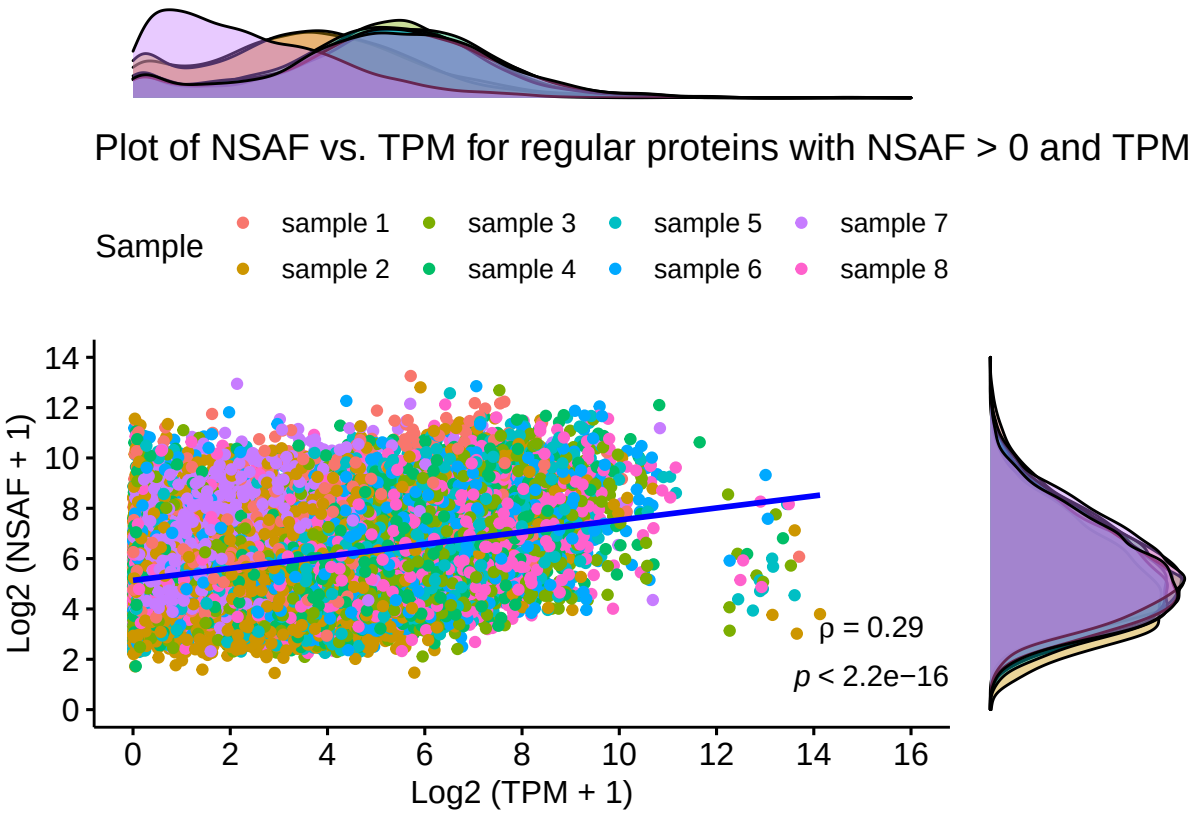


Figure 2: Heatmap for missing protein

Extract the 85 identified unique missing proteins Criterion: has both valid protein expression and RNA expression i.e. NASF > 0 & TPM > 0

```
# the number should be 85 if using two-SpC-rule
detected_MP_proteins <- combined_long  %>%  filter(accession %in% missing_proteins) %>% filter(NSAF > 0
# the details of the 85 identified missing proteins
detected_MP_proteins
```

```
##  [1] "A3GALT2"    "ACOXL"      "AGAP4"      "ALG1L2"     "ANKUB1"
##  [6] "ASB18"      "ATP6AP1L"   "C12orf56"   "C19orf67"   "C1orf127"
## [11] "C1orf141"   "C4orf36"    "C5orf67"    "C9orf129"   "C9orf163"
## [16] "CBWD5"      "CCDC163"    "CCNYL2"     "CSAG1"      "CTAGE1"
## [21] "CTAGE15"    "CTAGE6"     "CTXN1"      "DDTL"       "DNAJC9-AS1"
## [26] "ETV2"       "FER1L4"     "FTCDNL1"    "GRXCR2"     "HES2"
## [31] "HMSD"       "KIF25"      "LCN8"       "LINC01547"  "LRP5L"
## [36] "MRO"        "MS4A15"     "NCBP2L"     "NPIPB11"    "NPIPB8"
```

```
## [41] "OR2L2"      "OVOL3"      "PCDHAC1"    "PPIAL4G"    "PPM1N"
## [46] "PRAC1"      "PRAMEF19"   "PROX2"      "PRSS48"     "PRSS51"
## [51] "PTGES3L"    "R3HDML"     "RASGEF1C"   "RFPL2"      "RFPL4AL1"
## [56] "RGL4"       "RGPD1"      "RNF215"     "RPRML"      "SLAMF9"
## [61] "SLC22A25"   "SLFN12L"    "SMCO2"      "SMIM18"     "SPATA1"
## [66] "SPATA21"    "STPG2"      "STRC"       "TDH"        "TMC3"
## [71] "TMEM105"    "TMEM235"    "TRAV34"     "TRAV6"      "TRAV9-2"
## [76] "TSTD3"      "TTLL3"      "UQCRHL"     "USP50"      "WASH2P"
## [81] "WFDC11"     "ZNF208"     "ZNF233"     "ZNF429"     "ZNF582"
```

the PE (protein evidence) distribution of the identified missing proteins

```
missing_protein_df %>% filter(`gene name(s)` %in% detected_MP_proteins) %>% select(PE) %>% pull() %>% ta
```

```
## .
## Evidence at transcript level       Inferred from homology
##                           83                            1
##                    Predicted
##                            1
```

Figure 2A)

Heatmap representations for quantified missing proteins in the eight gastric cancer primary cell samples. Color represents the level of protein expression, and values were log (NSAF +1) transformed

```
# double check clustering
library(pheatmap)
heatmap_dat <- log(gastric_nsaf_gene_level[detected_MP_proteins, ] + 1)
colnames(heatmap_dat) =paste('sample', 1:8)
pheatmap(heatmap_dat,
         show_rownames = T,
         angle_col=45,
         cluster_cols = T,
         cluster_rows = T)
```

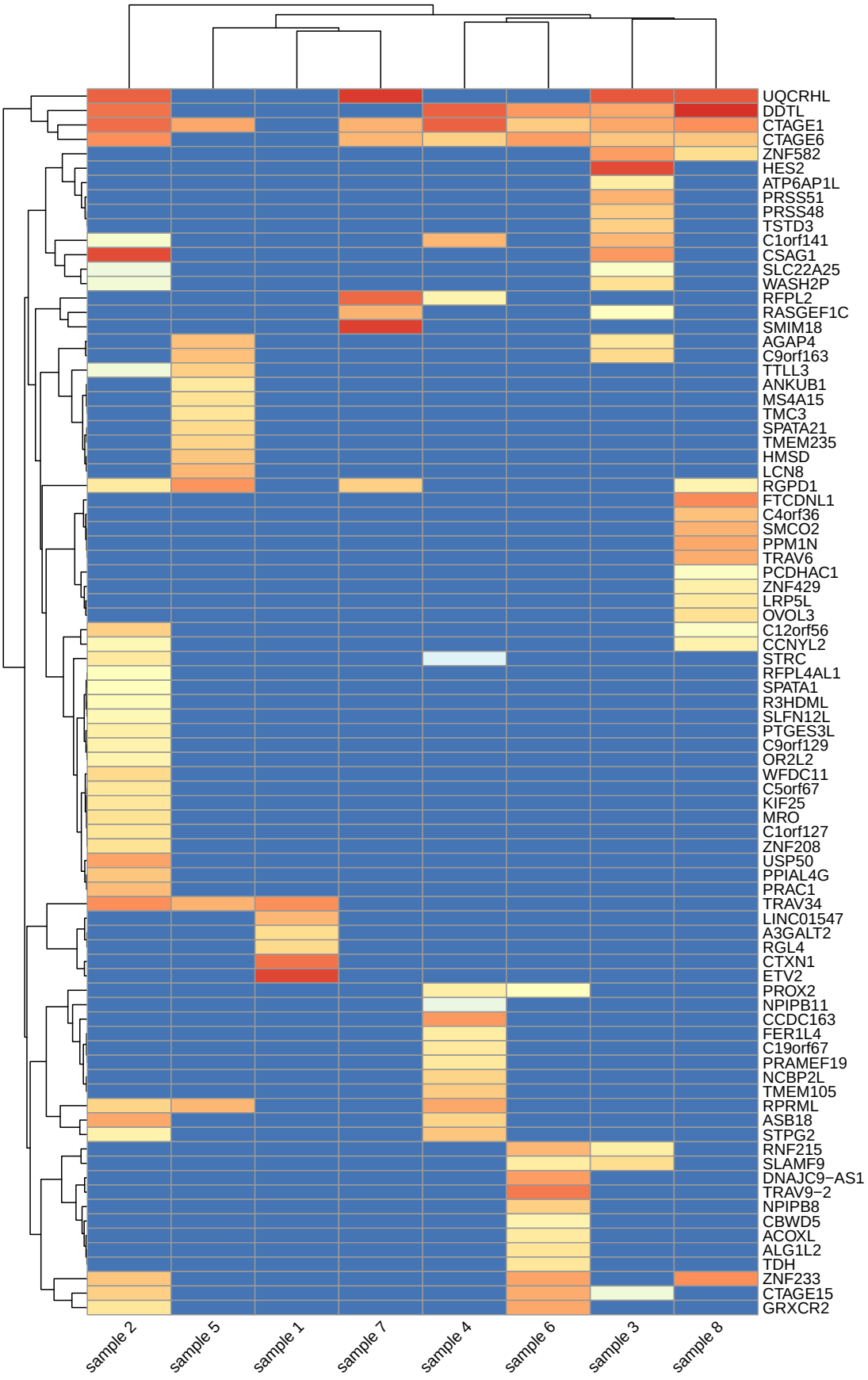

UQCRHL
DDTL
CTAGE1
CTAGE6
ZNF582
HES2
ATP6AP1L
PRSS51
PRSS48
TSTD3
C1orf141
CSAG1
SLC22A25
WASH2P
RFPL2
RASGEF1C
SMIM18
AGAP4
C9orf163
TTLL3
ANKUB1
MS4A15
TMC3
SPATA21
TMEM235
HMSD
LCN8
RGPD1
FTCDNL1
C4orf36
SMCO2
PPM1N
TRAV6
PCDHAC1
ZNF429
LRP5L
OVOL3
C12orf56
CCNYL2
STRC
RFPL4AL1
SPATA1
R3HDML
SLFN12L
PTGES3L
C9orf129
OR2L2
WFDC11
C5orf67
KIF25
MRO
C1orf127
ZNF208
USP50
PPIAL4G
PRAC1
TRAV34
LINC01547
A3GALT2
RGL4
CTXN1
ETV2
PROX2
NPIPB11
CCDC163
FER1L4
C19orf67
PRAMEF19
NCBP2L
TMEM105
RPRML
ASB18
STPG2
RNF215
SLAMF9
DNAJC9−AS1
TRAV9−2
NPIPB8
CBWD5
ACOXL
ALG1L2
TDH
ZNF233
CTAGE15
GRXCR2
sample 2
sample 5
sample 1
sample 7
sample 4
sample 6
sample 3
sample 8
6
5
4
3
2
1
0

Figure 2 additional

Heatmap representations for quantified missing proteins in the eight gastric cancer primary cell samples. Color represents the level of mRNA expression, and values were log (TPM +1) transformed

```
heatmap_dat_TPM <- log(count_TPM_gene_symbol[detected_MP_proteins, ] + 1)
colnames(heatmap_dat_TPM) =paste('sample', 1:8)
pheatmap(heatmap_dat_TPM,
         show_rownames = T,
         angle_col=45,
         cluster_cols = T,
         cluster_rows = T)
```

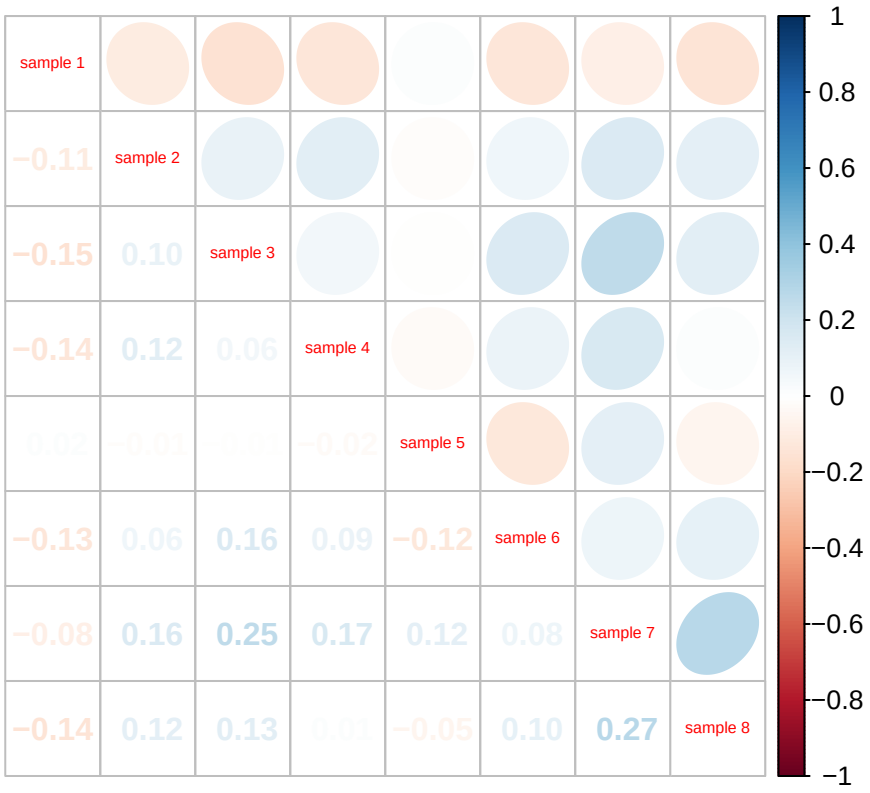

CTXN1
UQCRHL
FER1L4
TTLL3
WASH2P
LINC01547
RNF215
C4orf36
ETV2
C9orf163
PPM1N
CSAG1
RGL4
OVOL3
C9orf129
DNAJC9−AS1
SPATA1
SPATA21
ACOXL
RFPL2
NPIPB8
C12orf56
MS4A15
OR2L2
PRSS48
TDH
ZNF208
DDTL
TSTD3
CCNYL2
RFPL4AL1
LCN8
R3HDML
SLC22A25
RGPD1
C5orf67
WFDC11
GRXCR2
RASGEF1C
C1orf141
ASB18
STPG2
CTAGE6
PRAC1
SMCO2
A3GALT2
HMSD
KIF25
PCDHAC1
NCBP2L
TMC3
ANKUB1
USP50
PROX2
SMIM18
TMEM235
TMEM105
ALG1L2
C19orf67
STRC
PRAMEF19
PPIAL4G
ATP6AP1L
FTCDNL1
C1orf127
CTAGE1
HES2
ZNF582
ZNF233
ZNF429
TRAV34
SLFN12L
TRAV6
MRO
TRAV9−2
CCDC163
SLAMF9
AGAP4
LRP5L
CBWD5
NPIPB11
PRSS51
RPRML
CTAGE15
PTGES3L
sample 7
sample 6
sample 8
sample 3
sample 5
sample 4
sample 1
sample 2
8
6
4
2
0

correlation plot for pairwise correlation of missing protein expression

```
library(corrplot)
```

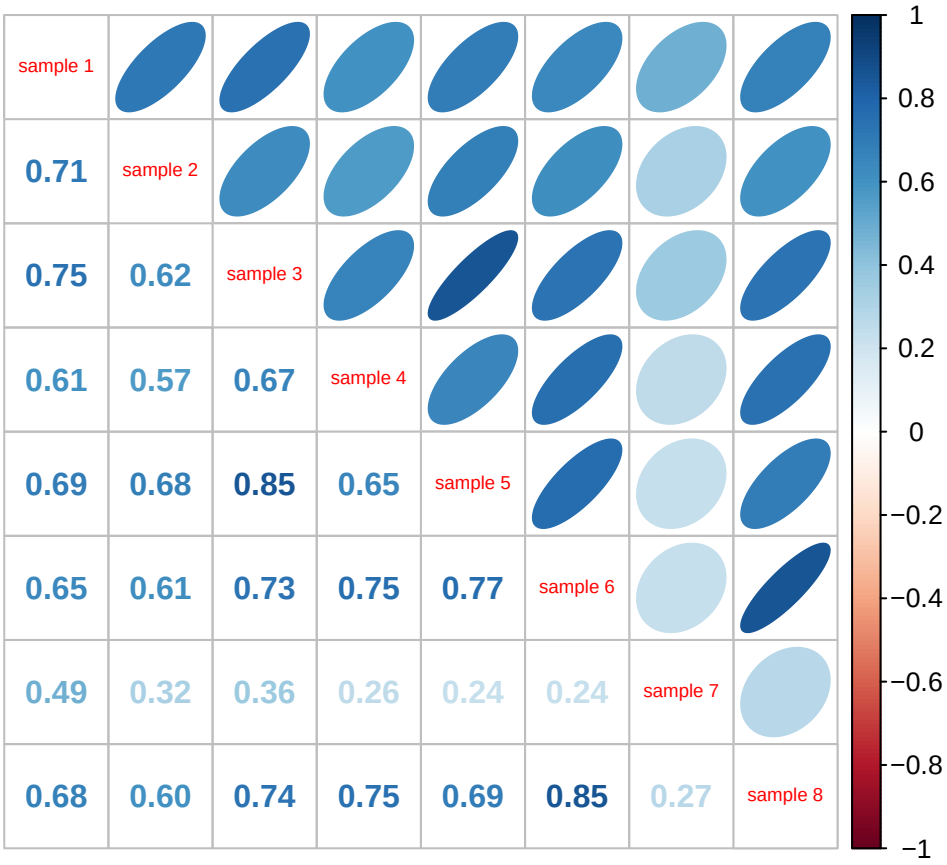
```
## corrplot 0.92 loaded
```

```
corr <- cor(heatmap_dat, method = "spearman")
corrplot.mixed(corr, lower = 'number', upper='ellipse', tl.cex=0.5,
               na.label.col='black',
               mar=c(0,0,2,0))
```



correlation plot for pairwise correlation of missing protein gene expression

```
corr <- cor(heatmap_dat_TPM, method = "spearman")
corrplot.mixed(corr, lower = 'number', upper='ellipse', tl.cex=0.6, na.label.col='black')
```


sample 1
0.71
sample 2
0.75
0.62
sample 3
0.61
0.57
0.67
sample 4
0.69
0.68
0.85
0.65
sample 5
0.65
0.61
0.73
0.75
0.77
sample 6
0.49
0.32
0.36
0.26
0.24
0.24
sample 7
0.68
0.60
0.74
0.75
0.69
0.85
0.27
sample 8
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1