

main analysis

2022-09-21

setup rmarkdown knit profile

```
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     vforcats 0.5.2
## -- Conflicts -----
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()   masks stats::lag()

# Some function Requires the in-house package "protoools"
if (!require("protoools", quietly = TRUE))
  devtools::install_github("https://github.com/FDUguchunhui/protoools")
library('protoools')
```

Import data

```
# import gastric primary cell nsaf data that filtered with Spectral count (SpC) >=2
gastric_nsaf <- as.matrix(read.csv('gastric-primary-cell/gastric_primary_cell_nsaf_2.csv', row.names = 1))
# # import gastric primary cell SpC data
gastric_count <- read_csv('gastric-primary-cell/gastric_primary_cell_count_2.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)
# load missing protein information
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 P2-5 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_gene_symnol <- missing_protein_df %>% filter(PE != 'Uncertain') %>% select(`gene name`)
P2_5_proteins_gene_symbol <- missing_protein_df %>% select(`gene name(s)`) %>% pull()

# missing_proteins is a character vector of MP accession ID
# similar P2_5_proteins are protein accession 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 sample
# 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(-accession))
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(-accession))
RP_counts_long$type <- 'Regular'

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

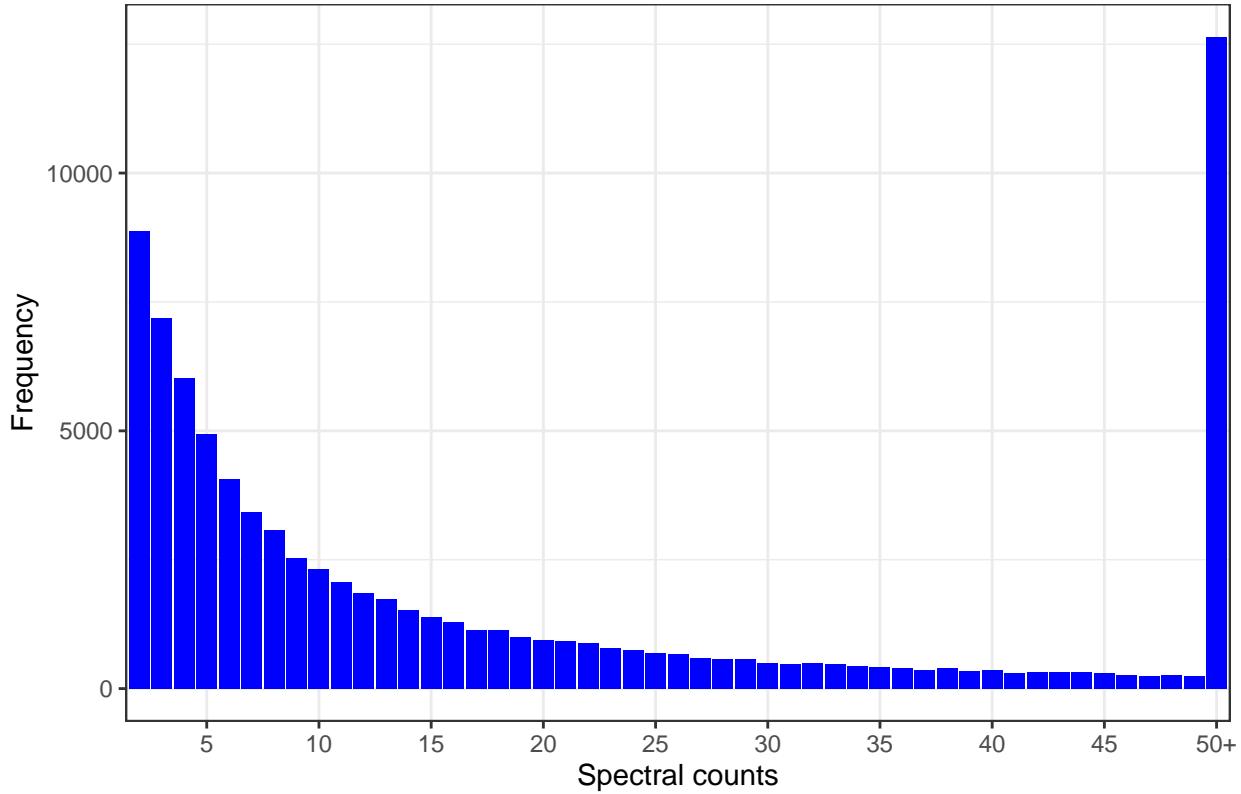
Figure S4 The peptide-spectral-matching (PSM) count distribution for regular proteins (Left blue) and missing proteins (Right red)

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), '50+'))
RP_barplot_data %>%
  ggplot(aes(x=count_discrete)) +
  geom_bar(position = 'identity', fill='blue') +
  theme_bw() +
  xlab('Spectral counts') +
  ylab('Frequency') +
```

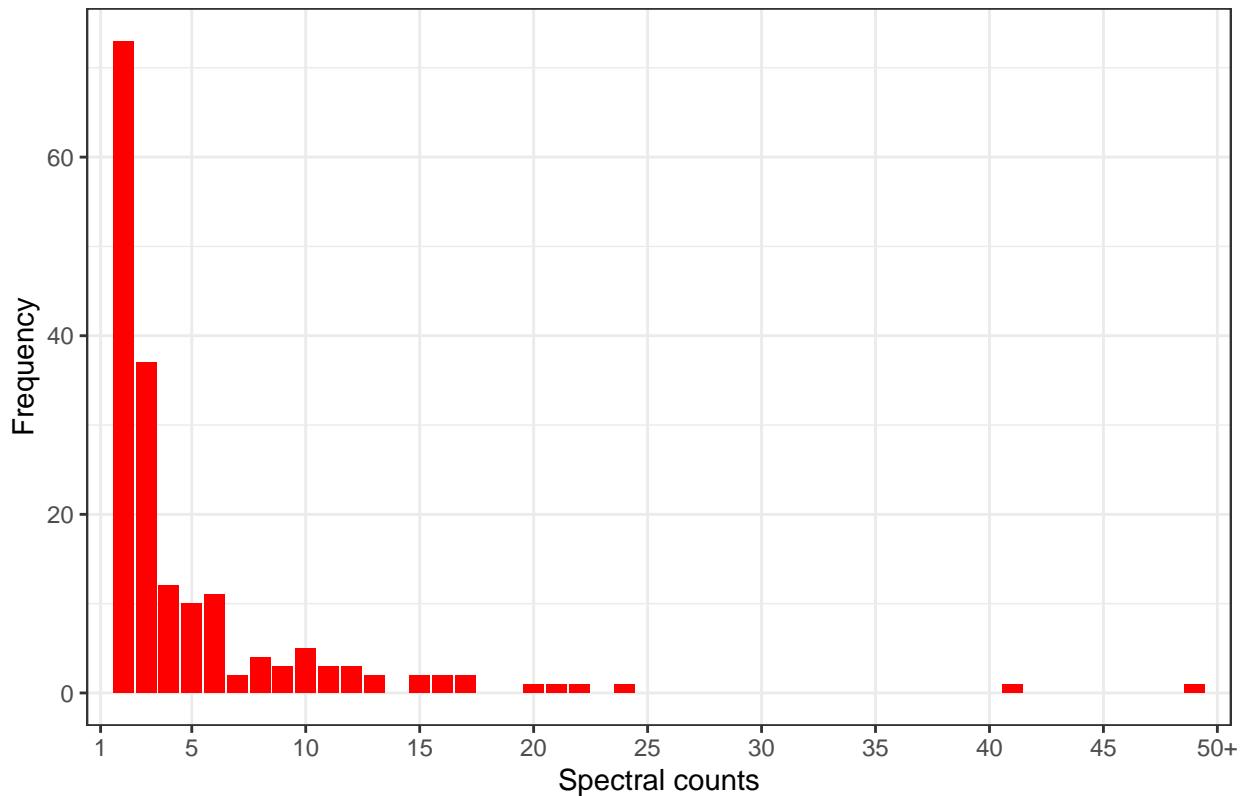
```
ggtitle('Spectral count frequency of regular proteins') +
scale_x_discrete(breaks=c('1', seq(5, 50, 5), '50+'))
```

Spectral count frequency of regular proteins



```
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), '50+'))
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)
```

Spectral count frequency of missing proteins



check the number of SpC for missing proteins

```
MP_counts_long %>% filter(count >= 2) %>% summary()
```

```
##   accession          IPAS      count       type
##   Length:177    Length:177   Min.   : 2.000  Length:177
##   Class  :character  Class  :character  1st Qu.: 2.000  Class  :character
##   Mode   :character  Mode   :character  Median  : 3.000  Mode   :character
##                                         Mean   : 5.153
##                                         3rd Qu.: 6.000
##                                         Max.   :49.000
```

Match mRNA expression to corresponding protein

The original proteomics data is in accession level and the original transcription data is in gene level.

For example: There are two proteins coded by gene CTAGE1: Q96RT6 and Q9HC47

and we have the following expression NSAF Q96RT6: 3 NSAF Q9HC47: 0 TPM CTAGE1: 3

Then after match mRNA expression, each protein will have a column indicating its protein expression in NSAF and another column indicating its corresponding gene expression in TPM

Q96RT6: NSAF: 3 TPM: 3 Q9HC47: NSAF: 0 TPM: 3

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
```

make long-format gastric NSAF data

```
gastric_nsaf_df <- as_tibble(gastric_nsaf, rownames='accession')
gastric_nsaf_df_long <- gastric_nsaf_df %>% pivot_longer(names_to = 'IPAS', values_to = 'NSAF', cols =
# type variable show whether a protein is a regular protein (RP, non-missing), missing protein (MP), or
gastric_nsaf_df_long <- gastric_nsaf_df_long %>% mutate(type=if_else(accession %in% missing_proteins_ac
                           if_else(!(accession %in% P2_5_proteins_accession)
```

get the correpsongding gene symbol for each protein

```
gastric_nsaf_df_long$gene_symbol <- mapping(gastric_nsaf_df_long$accession, dict_accession_to_gene_symbol)
rm(dict_accession_to_gene_symbol)
```

load transcription data

```
count TPM_gene_symbol <- read.csv("RNA-data/processed_RNA_data.csv", row.names = 1)
```

make long-format of RNA expression data

```
count TPM_gene_symbol_df <- as_tibble(count TPM_gene_symbol, rownames='gene_symbol')
count TPM_gene_symbol_df_long <- count TPM_gene_symbol_df %>% pivot_longer(names_to = 'IPAS', values_to
```

join the RNA expression to protein data

```
dat_combined_long <- left_join(gastric_nsaf_df_long, count TPM_gene_symbol_df_long, by=c('IPAS', 'gene_
dat_combined_long$NSAF <- ifelse(is.na(dat_combined_long$NSAF), 0, dat_combined_long$NSAF)
dat_combined_long$TPM <- ifelse(is.na(dat_combined_long$TPM), 0, dat_combined_long$TPM)
```

Unique Missing protein accession

```
(MP_unique_accession <- dat_combined_long %>% filter(NSAF > 0, TPM > 0, type=='MP') %>% select(accession
```

```

## [1] "AOA096LP55" "AOA0B4J273" "P58512"      "P60606"      "Q6ZVZ8"
## [6] "Q86UF2"     "Q8IXR9"      "Q8IZJ4"      "Q8N431"      "U3KPV4"
## [11] "A4D2H0"     "A6NFK2"      "A6NHG4"      "A6NK53"      "F2Z3F1"
## [16] "F8VTS6"     "PODJDO"      "PODKX4"      "PODN37"      "Q52LC2"
## [21] "Q5T035"     "Q5T2Q4"      "Q5VX52"      "Q6IEE8"      "Q6T423"
## [26] "Q6VEQ5"     "Q7RTU9"      "Q8N4K4"      "Q8N5U1"      "Q8N9H9"
## [31] "Q8NEX6"     "Q8NH16"      "Q96KF2"      "Q9H3Y0"      "Q9UIL4"
## [36] "Q9Y4R7"     "AOA1B0GVH4" "Q6PB30"      "Q7RTY5"      "Q8N9P6"
## [41] "Q96A28"     "Q96NG8"      "Q9Y6U7"      "A6NJJ6"      "A6PVI3"
## [46] "A9Z1Z3"     "E5RHQ5"      "O75678"      "Q3B8N5"      "Q8N8V8"
## [51] "A6NFN9"     "A8MTL9"      "Q6JVE9"      "Q7Z572"      "Q7Z5M5"
## [56] "AOA087WT02" "A6NH13"      "C9J202"      "Q5RIA9"      "Q8IZJ6"
## [61] "Q9NUZ1"     "AOA075B6T7" "A4QPB2"      "E5RQL4"      "Q86V71"
## [66] "Q96KX1"     "Q9H158"

```

```
length(MP_unique_accession)
```

```
## [1] 67
```

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

```
combined_long <- dat_combined_long %>% filter(!(TPM == 0 & NSAF == 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 exist
#' @param cases a vector of integer, used to decide whether cases to included for calculation
summarize_by_case <- function(tbl, protein_type, remove_IPAS=NULL, group_by_IPAS=FALSE, TPM_cutoff = 0, NSAF_cutoff = 0, cases=c(1,2))

tbl <- tbl %>% filter(type==protein_type)

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(type=='MP') %>% filter(NSAF > 0) %>% arrange(accession_id)

write_csv(full_table_detected_MP, 'detected_177_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, protein_type = 'MP', group_by_IPAS = TRUE, cases = c(1, 2)))

## '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     4      9 44.44%
## 2 IP0981_1701     2     5      9 55.56%
## 3 IP0982_1701     1    27     44 61.36%
## 4 IP0982_1701     2    17     44 38.64%
## 5 IP0993_1701     1    13     28 46.43%
## 6 IP0993_1701     2    15     28 53.57%
## 7 IP0995_1701     1     9     23 39.13%
## 8 IP0995_1701     2    14     23 60.87%
## 9 IP0999_1701     1    11     18 61.11%
## 10 IP0999_1701    2     7     18 38.89%
## 11 IP7100_1701    1     9     16 56.25%
## 12 IP7100_1701    2     7     16 43.75%
## 13 IP7103_1701    1     3     15 20%
## 14 IP7103_1701    2    12     15 80%
## 15 IP7105_1701    1    12     24 50%
## 16 IP7105_1701    2    12     24 50%

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

## overall without removing the outlier
(b <- summarize_by_case(combined_long, protein_type = 'MP', group_by_IPAS = FALSE, cases = c(1, 2)))

## '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     88    177 49.72%
## 2     2     89    177 50.28%
```

```

## overall with removing the outlier
(c <- summarize_by_case(combined_long, protein_type = 'MP', group_by_IPAS = FALSE, cases = c(1, 2), remo

## `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     81    153 52.94%
## 2     2     72    153 47.06%

```

histogram

show the number of detected missing protein and proportion of them having corresponding mRNA expression

```

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_1701",
                                                 "IP7100_1701", "IP7103_1701", "IP7105_1701", "Combined"))

ggplot(data=dat_hist, aes(x=IPAS, y=count, group=case, fill=case)) +
  labs(title="Coverage rate for protein 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))

```

Coverage rate for protein product with RNA product in missing proteins

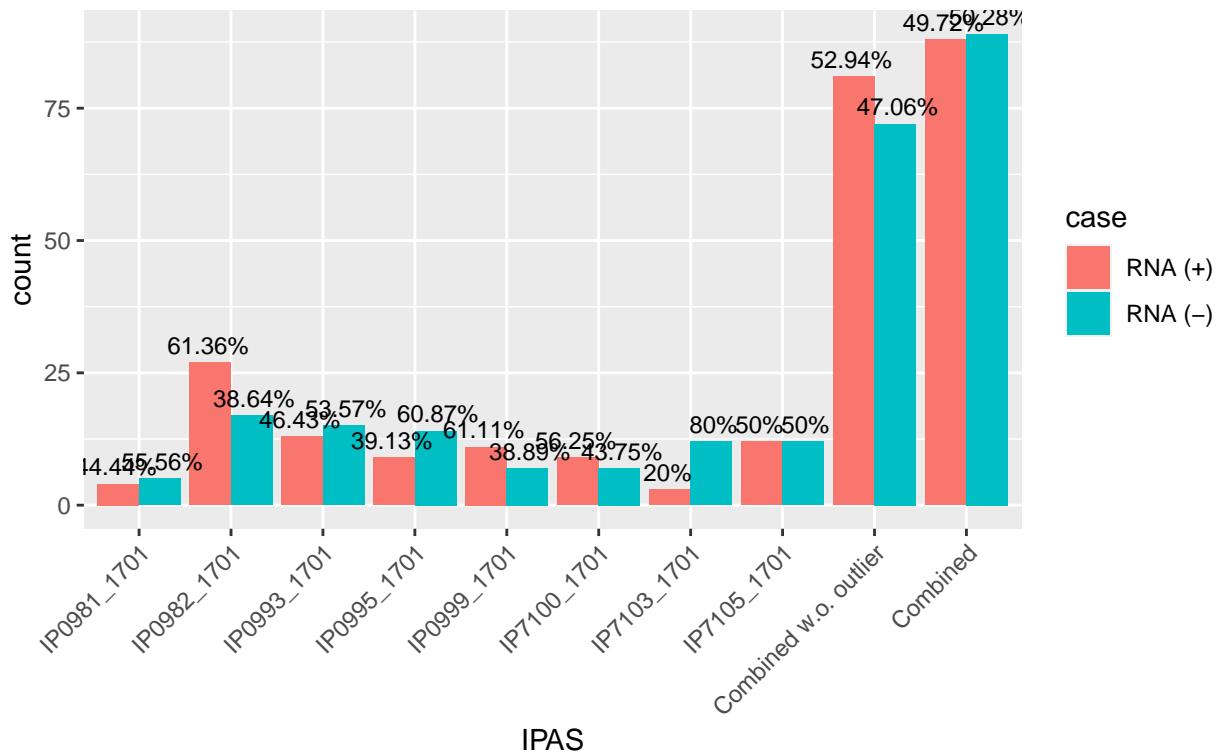


Table 2 non-missing proteins part

The execution of the following chunks of code could be slow for the size of non-missing proteins. Code is split into several chunk otherwise the computation time required is long.

```
# the coverage rate of RNA product in protein product for non-missing protein
(a_RP <- summarize_by_case(combined_long, protein_type = 'RP', group_by_IPAS = TRUE, cases = c(1, 2)))

## `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    5063    5593 90.52%
## 2 IP0981_1701     2     530    5593  9.48%
## 3 IP0982_1701     1   12642   13864 91.19%
## 4 IP0982_1701     2    1222   13864  8.81%
## 5 IP0993_1701     1   13113   14225 92.18%
## 6 IP0993_1701     2    1112   14225  7.82%
## 7 IP0995_1701     1    9667   10436 92.63%
## 8 IP0995_1701     2     769   10436  7.37%
```

```

##  9 IP0999_1701      1  9267   9915 93.46%
## 10 IP0999_1701      2   648   9915 6.54%
## 11 IP7100_1701      1  8273   8939 92.55%
## 12 IP7100_1701      2   666   8939 7.45%
## 13 IP7103_1701      1  7058   8017 88.04%
## 14 IP7103_1701      2   959   8017 11.96%
## 15 IP7105_1701      1 11064  11838 93.46%
## 16 IP7105_1701      2   774   11838 6.54%

## overall without removing the outlier
b_RP <- a_RP %>% group_by(case) %>%
  summarise(count= sum(count)) %>%
  mutate(countT=sum(count)) %>%
  mutate(per=paste0(round(100*count/countT,2), '%'))

```

```

## overall after removing the outlier
c_RP <- a_RP %>% filter(!IPAS %in% c('IP0981_1701', 'IP7103_1701')) %>%
  group_by(case) %>%
  summarise(count= sum(count)) %>%
  mutate(countT=sum(count)) %>%
  mutate(per=paste0(round(100*count/countT,2), '%'))

```

barplot for non-missing proteins

```

b_RP$IPAS = 'Combined'
c_RP$IPAS = 'Combined2'
dat_hist <- rbind(a_RP, b_RP, c_RP)

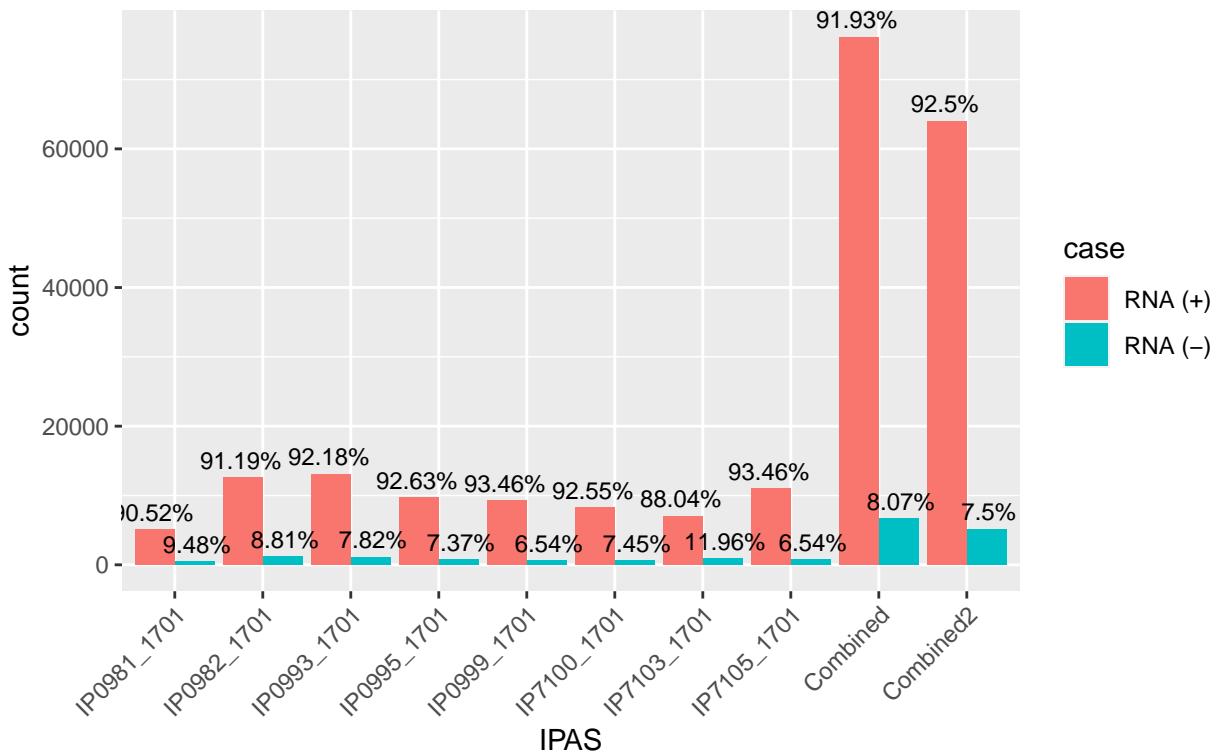
dat_hist$case <- factor(dat_hist$case, levels = c('1', '2'), labels = c('RNA (+)', 'RNA (-)'))
dat_hist$case <- relevel(dat_hist$case, 'RNA (+)')

dat_hist$IPAS <- factor(dat_hist$IPAS, levels = c("IP0981_1701", "IP0982_1701", "IP0993_1701", "IP0995_1701",
                                                   "IP7100_1701", "IP7103_1701", "IP7105_1701", "Combined"))

ggplot(data=dat_hist, aes(x=IPAS, y=count, group=case, fill=case)) +
  labs(title="Coverage rate for protein 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))

```

Coverage rate for protein product with RNA product in non-missing proteins



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

```
glmm_dat_missing <- dat_combined_long %>% filter(type=='MP') %>% mutate(TPM = log(TPM + 1), NSAF = log(NSAF + 1))
glmm_dat_missing_filtered <- glmm_dat_missing %>% filter(TPM > 0 & NSAF > 0)

glmm_dat_non_missing <- dat_combined_long %>% filter(type=='RP') %>% mutate(TPM = log(TPM + 1), NSAF = log(NSAF + 1))
glmm_dat_non_missing_filtered <- glmm_dat_non_missing %>% filter(TPM > 0 & NSAF > 0)
```

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:tidyverse':
## 
##     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: 240.6
## 
## Scaled residuals:
##       Min      1Q  Median      3Q     Max
## -2.09524 -0.69004  0.01369  0.61513  2.74569
## 
## Random effects:
##   Groups   Name Variance Std.Dev.
##   IPAS     TPM  0.009444 0.09718
##   Residual    0.834885 0.91372
##   Number of obs: 88, groups:  IPAS, 8
## 
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept) 4.22987   0.12801 33.043
## TPM         0.19079   0.07764  2.458
## 
## Correlation of Fixed Effects:
##   (Intr)  TPM 
## TPM -0.567
```

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      3 240.46 247.89 -117.23   234.46
## glmm_model     4 242.46 252.37 -117.23   234.46      0  1

```

So, a simple linear regression can be used

```

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
## -1.93447 -0.64185 -0.00102  0.56857  2.53510
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.23100   0.12980 32.597 < 2e-16 ***
## TPM         0.17953   0.06804  2.639  0.00988 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9275 on 86 degrees of freedom
## Multiple R-squared:  0.0749, Adjusted R-squared:  0.06414
## F-statistic: 6.962 on 1 and 86 DF,  p-value: 0.00988

```

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 (NSAF) 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'
```

Normalized spectral abundance factor (NSAF) vs. Transcripts per million (TPM) missing proteins on gene level in gastric cancer primary cell samples

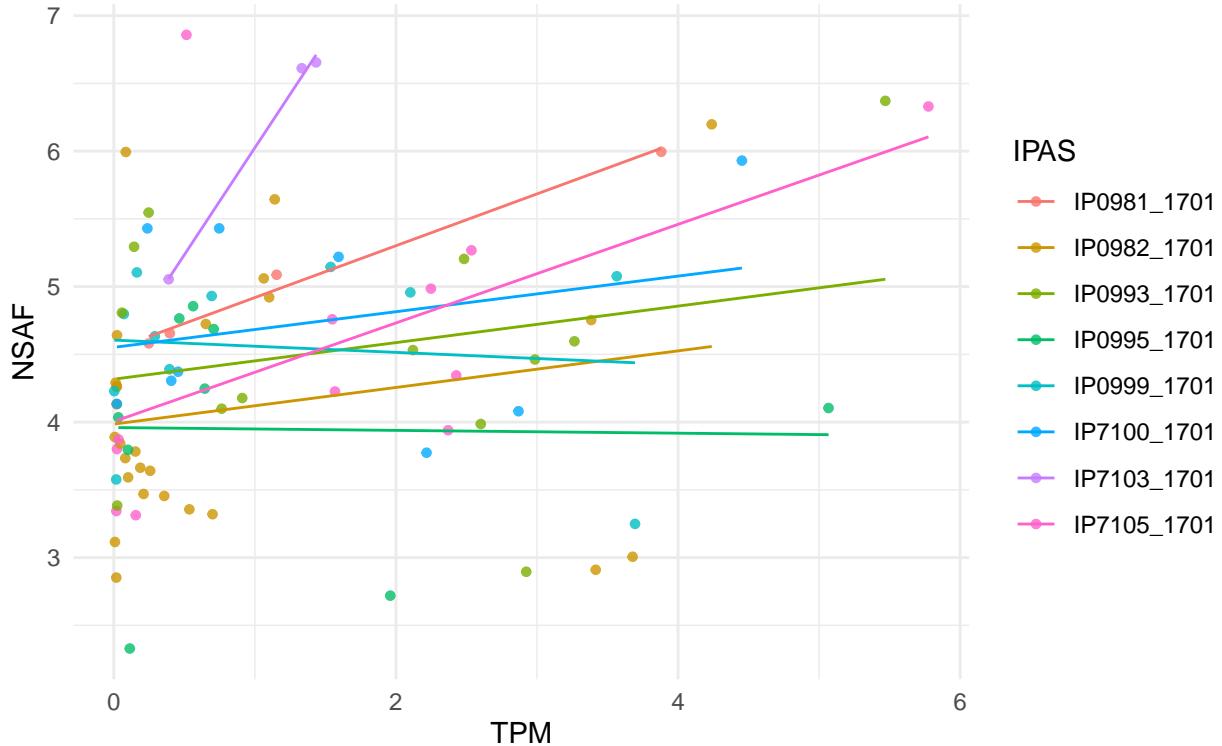
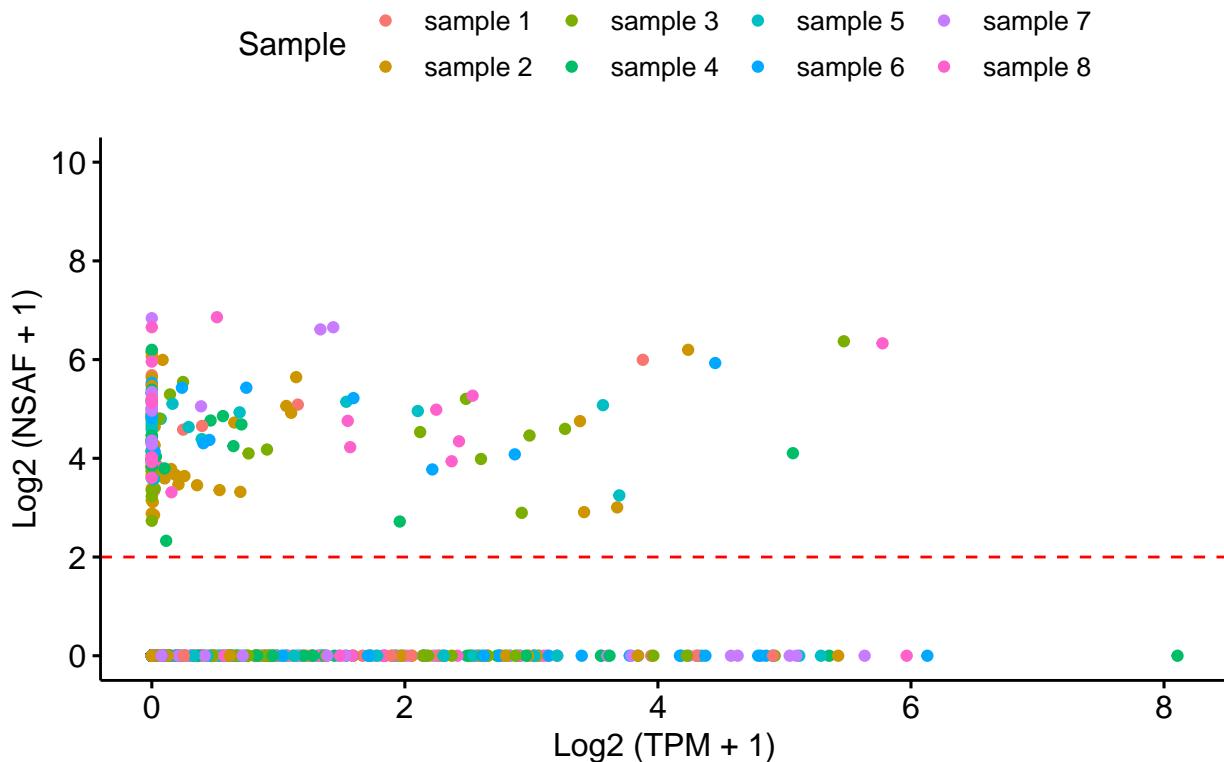


Figure 3 with spearman correlation for missing proteins

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(element_text(size=10)) +
  labs(color='Sample')
```

Plot of NSAF vs. TPM for missing proteins



```
# 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 regression 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_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, 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()

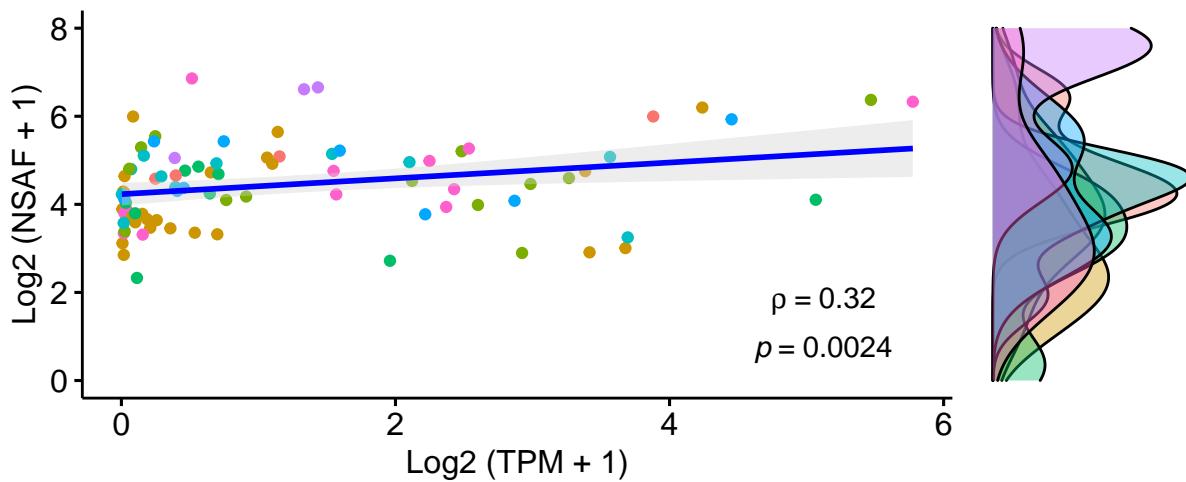
dens1 + patchwork::plot_spacer() + plt + dens2 +  patchwork::plot_layout(ncol = 2, nrow = 2, widths = c
## `geom_smooth()` using formula 'y ~ x'

```



Plot of NSAF vs. TPM for missing proteins with NSAF > 0 and TPM >

Sample	● sample 1	● sample 3	● sample 5	● sample 7
	● sample 2	● sample 4	● sample 6	● sample 8

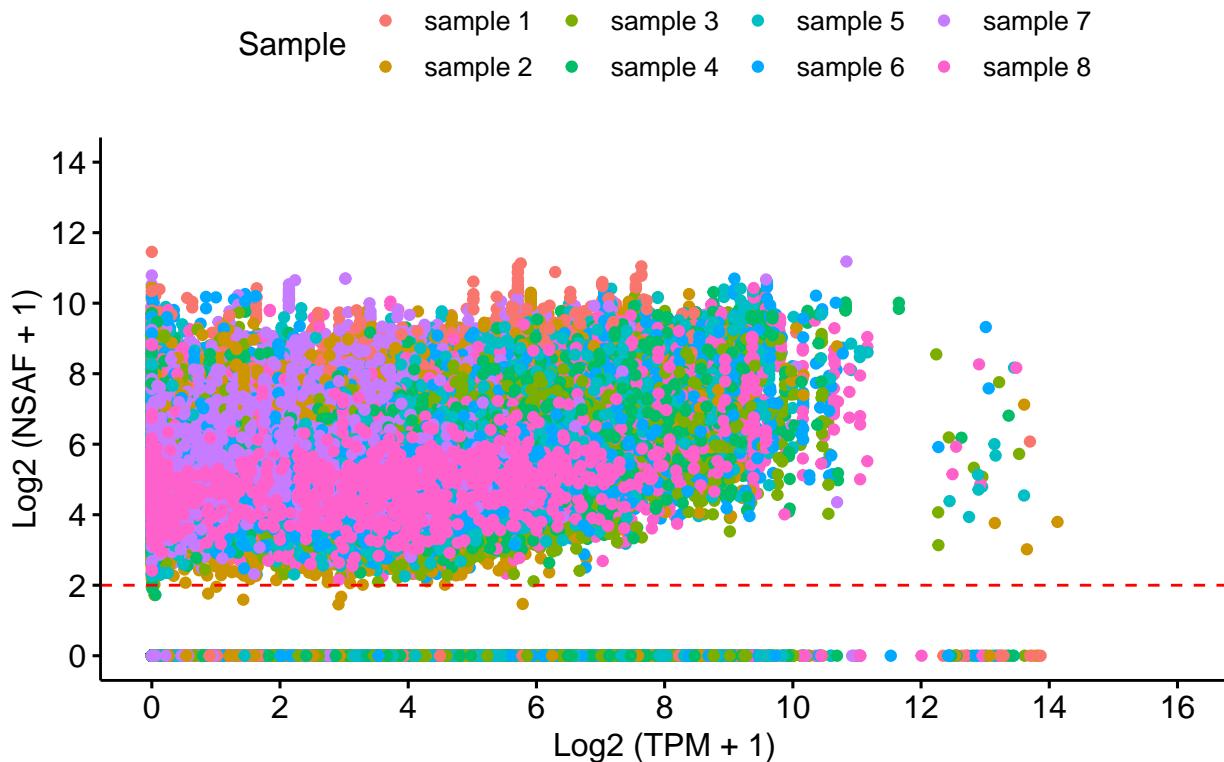


```

# plot with spearman correlation for non-missing proteins
ggpubr::ggsscatter(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')

```

Plot of NSAF vs. TPM for regular proteins



```

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 regression 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(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") +

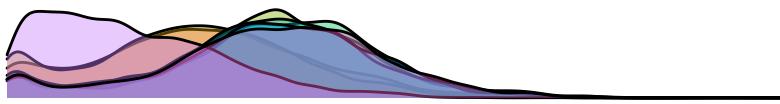
```

```

coord_flip()

dens1 + patchwork::plot_spacer() + plt + dens2 +  patchwork::plot_layout(ncol = 2, nrow = 2, widths = c
## `geom_smooth()` using formula 'y ~ x'

```



Plot of NSAF vs. TPM for regular proteins with NSAF > 0 and TPM >

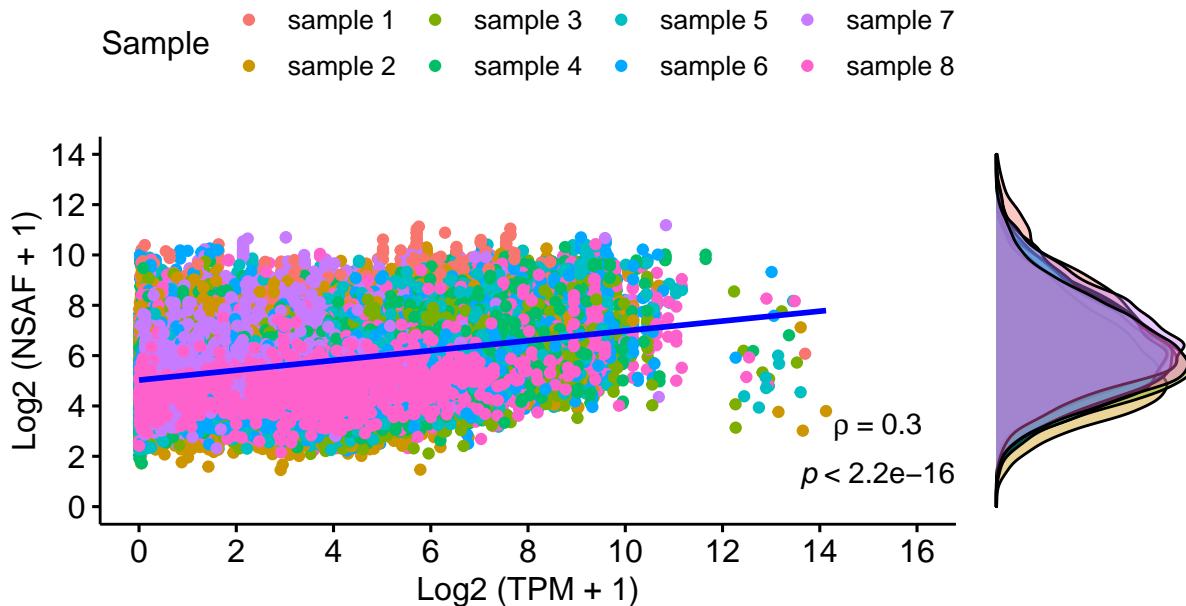


Figure 2: Heatmap for missing protein

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

```

# the number should be 85 if using two-SpC-rule
identified_MP <- combined_long %>% filter(NSAF > 0, TPM > 0, type=='MP') %>% arrange(accession) %>% g
# the details of the 85 identified missing proteins
identified_MP

```

```

## [1] "AOA075B6T7" "AOA087WT02" "AOA096LP55" "AOA0B4J273" "AOA1B0GVH4"
## [6] "A4D2H0"      "A4QPB2"      "A6NFK2"      "A6NFN9"      "A6NH13"
## [11] "A6NHG4"     "A6NJ6"       "A6NK53"      "A6PVI3"      "A8MTL9"
## [16] "A9Z1Z3"      "C9J202"      "E5RHQ5"      "E5RQL4"      "F2Z3F1"
## [21] "F8VTS6"      "075678"      "PODJDO"      "PODKX4"      "PODN37"
## [26] "P58512"      "P60606"      "Q3B8N5"      "Q52LC2"      "Q5RIA9"
## [31] "Q5T035"      "Q5T2Q4"      "Q5VX52"      "Q6IEE8"      "Q6JVE9"
## [36] "Q6PB30"      "Q6T423"      "Q6VEQ5"      "Q6ZVZ8"      "Q7RTU9"

```

```

## [41] "Q7RTY5"      "Q7Z572"       "Q7Z5M5"       "Q86UF2"       "Q86V71"
## [46] "Q8IXR9"      "Q8IZJ4"       "Q8IZJ6"       "Q8N431"       "Q8N4K4"
## [51] "Q8N5U1"       "Q8N8V8"       "Q8N9H9"       "Q8N9P6"       "Q8NEX6"
## [56] "Q8NH16"       "Q96A28"       "Q96KF2"       "Q96KX1"       "Q96NG8"
## [61] "Q9H158"       "Q9H3Y0"       "Q9NUZ1"       "Q9UIL4"       "Q9Y4R7"
## [66] "Q9Y6U7"       "U3KPV4"

```

```
identified_MP_gene_symol <- combined_long %>% filter(NSAF > 0, TPM > 0, type=='MP') %>% select(gene_sy
```

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

```

missing_protein_df %>% filter(`Accession` %in% identified_MP) %>% select(PE) %>% pull() %>% table()

## .
## Evidence at transcript level          Predicted
##                               66                  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[identified_MP, ] + 1)
colnames(heatmap_dat) = paste('sample', 1:8)
pheatmap(heatmap_dat,
         show_rownames = T,
         angle_col=45,
         cluster_cols = T,
         cluster_rows = T)

```

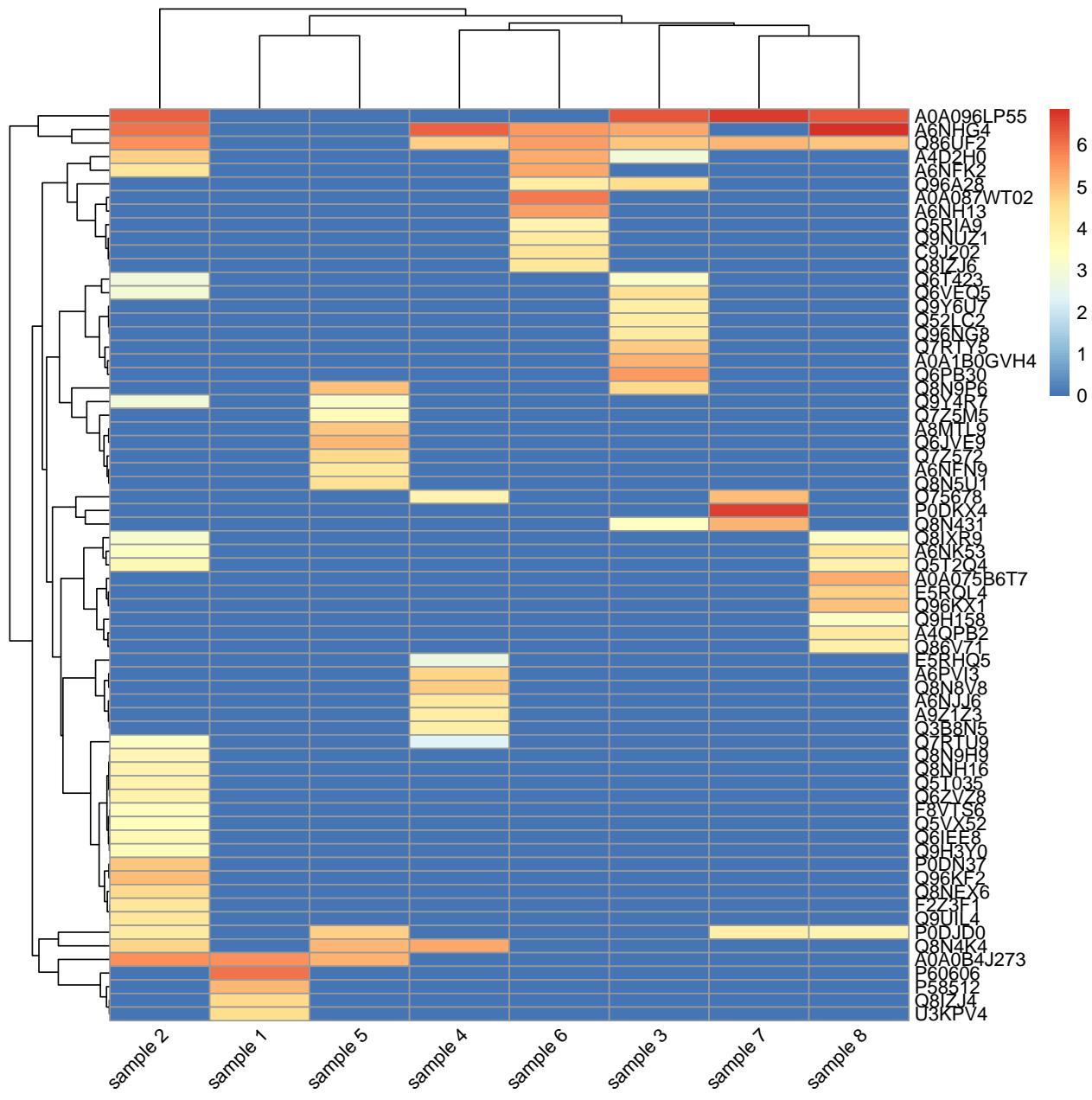
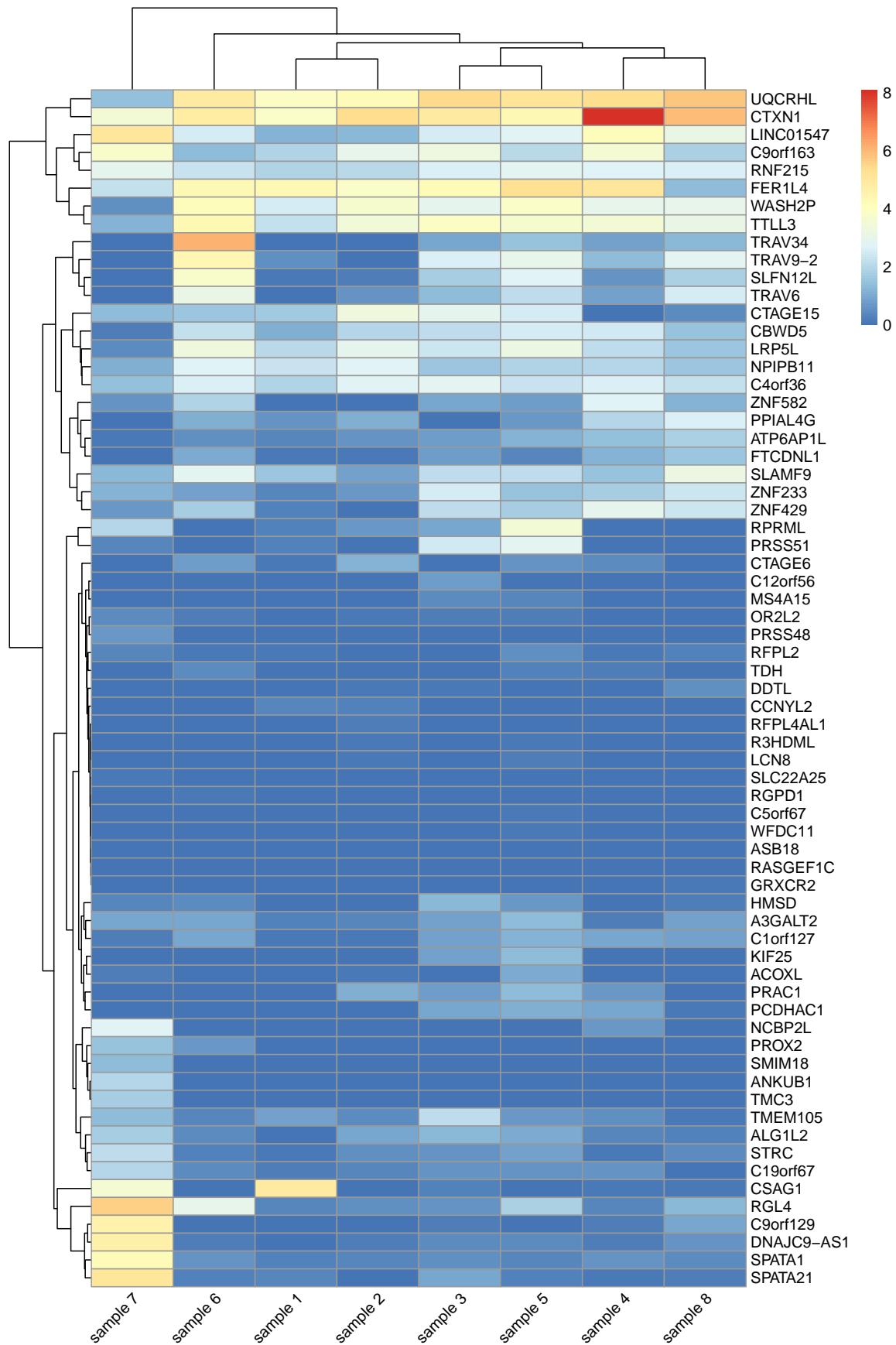


Figure 2 additional

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[identified_MP$gene_symbol, ] + 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)
```



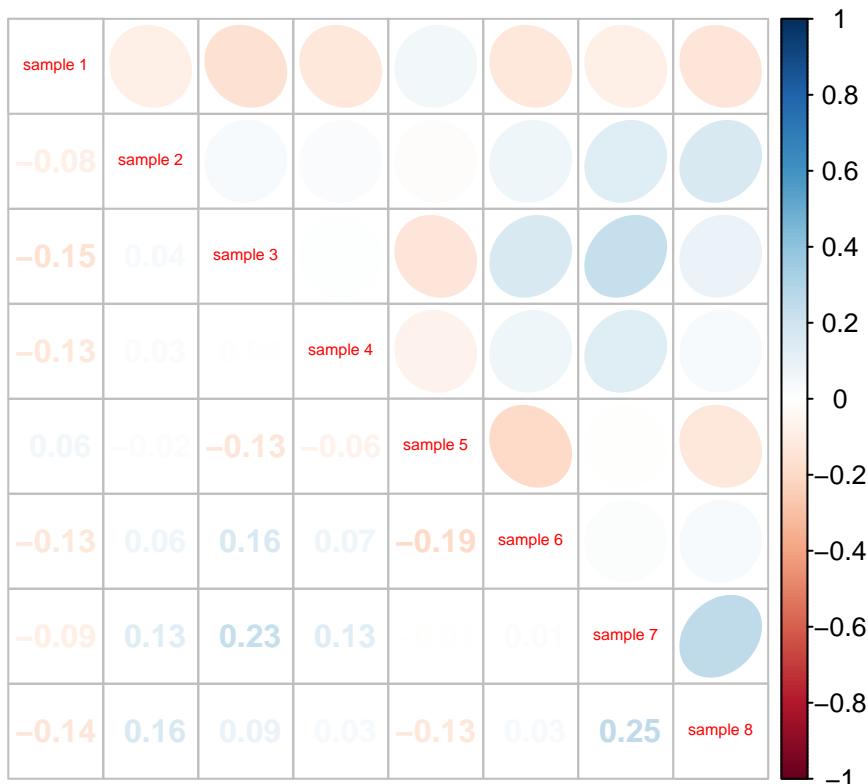
Checking Sample heterogeneity

Check whether RNA expression, protein expression, and protein-RNA-expression relationship were similar between samples ## 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')
```

