

# RNA data processing and exploration

2022-09-27

```
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()

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

## import the RNA data

```
count <- read_csv('RNA-data/merged_IPAS_RNAseq_counts_gastric.csv')

## New names:
## Rows: 63930 Columns: 10
## -- Column specification
## ----- Delimiter: "," chr
## (1): ...1 dbl (9): IPAS7103, IPAS0972, IPAS0995, IPAS0982, IPAS0999, IPAS7100,
## IPAS099...
## 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.
## * ' -> '...1'

count <- count[-c(1:5), ]
colnames(count) <- c('Ensemble_ID', colnames(count)[-1])
count$Ensemble_ID <- str_extract(count$Ensemble_ID, pattern = '.*(=?=\\.)')
head(count)

## # A tibble: 6 x 10
##   Ensemble_ID      IPAS7103 IPAS0972 IPAS0995 IPAS0982 IPAS0999 IPAS7100 IPAS099
##   <chr>          <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 ENSG00000000003      0      3891      2139      1397      1755      365      190
## 2 ENSG00000000005      0        0        0        0        0        0
```

```
## 3 ENSG00000000419      25      2173      5350      1388      1574      2032      2414
## 4 ENSG00000000457      175      2392      3390      1130      2927      1572      1546
## 5 ENSG00000000460      269      1602      2865      788      1368      839      1359
## 6 ENSG00000000938       3      1746      173      361      3089      27631      3783
## # i 2 more variables: IPAS0981 <dbl>, IPAS7105 <dbl>
```

## data wrangling

combine duplicate ensemble\_id 63,920 -> 63,875

```
count <- count %>% group_by(Ensemble_ID) %>% summarise_all(sum)
```

63,875 -> 43,939 remove rows that all cells are 0

```
count <- count[-which(rowSums(count[-1]) == 0),]
```

## import missing protein information

```
## read-in newest missing protein list
missing_protein_df <- readxl::read_xlsx('support-data/PE2-5.xlsx')
missing_protein_df <- missing_protein_df[-c(1:12), ]
head(missing_protein_df)

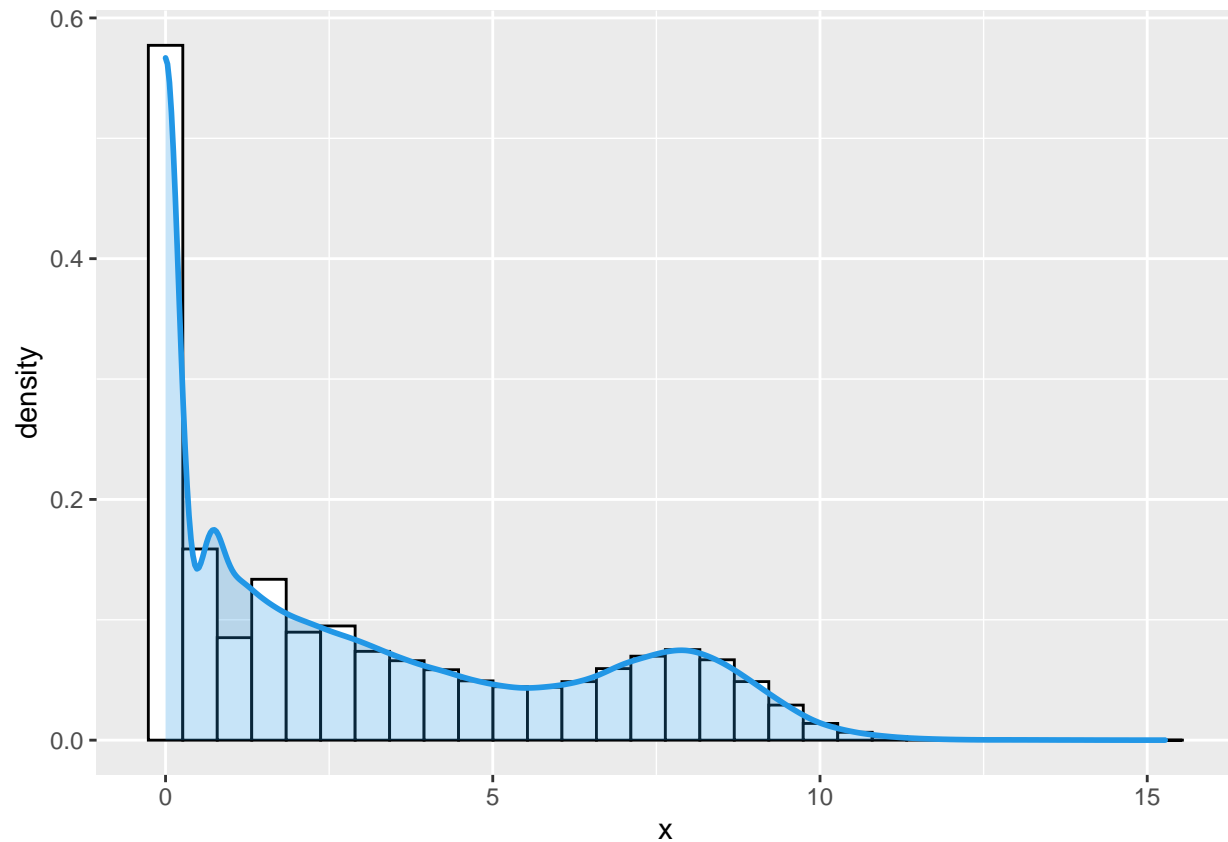
## # A tibble: 6 x 14
##   'acc. code'      Accession 'protein name' 'gene name(s)' chromosome proteomics
##   <chr>           <chr>      <chr>          <chr>          <chr>      <chr>
## 1 "{\NX_A0A087WT~ A0A087WT~ Putative pota~ KCNE1B      21p11.2    no
## 2 "{\NX_A0A087WU~ A0A087WU~ Neuroblastoma~ NBPF19      1q21.2    yes
## 3 "{\NX_A0A087WX~ A0A087WX~ Putative elon~ ELOA3CP     18q21.1    no
## 4 "{\NX_A0A087X1~ A0A087X1~ Putative cyto~ CYP2D7     22q13.2    yes
## 5 "{\NX_A0A0B4J2~ A0A0B4J2~ Putative glut~ GATD3B     21p12      yes
## 6 "{\NX_A0A0B4J2~ A0A0B4J2~ Putative seri~ SIK1B      21p12      yes
## # i 8 more variables: disease <chr>, structure <chr>, '#isof.' <dbl>,
## #   '#variants' <dbl>, '#PTMS' <dbl>, mutagenesis <chr>, 'tissue expr.' <chr>,
## #   PE <chr>
```

## Section 0: Exploration

### overall distribution

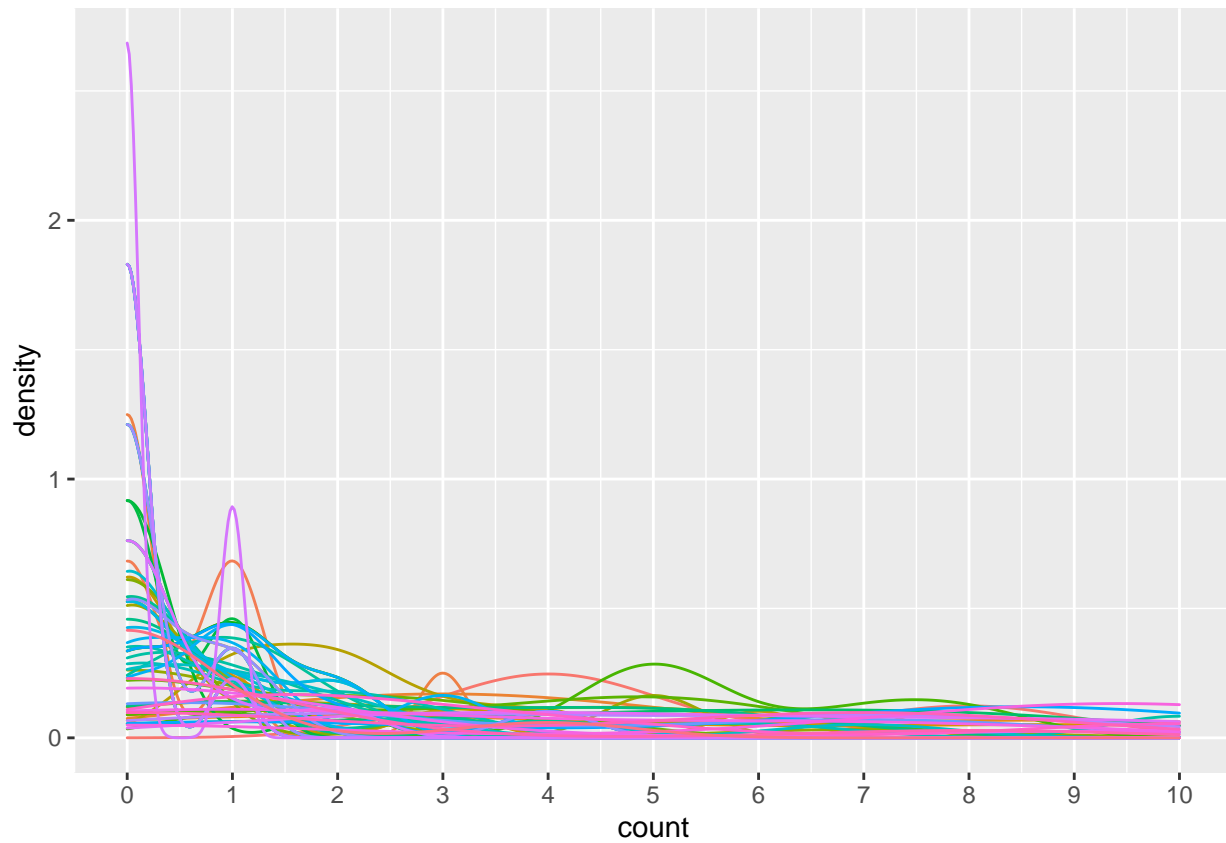
```
count_df <- as.data.frame(count[-1])
pseudo_counts <- log(count_df + 1)
ggplot(data = data.frame(x=c(as.matrix(pseudo_counts))), aes(x=x)) +
  geom_histogram(aes(y = ..density..),
    colour = 1, fill = "white") +
  geom_density(lwd = 1, colour = 4,
    fill = 4, alpha = 0.25)
```

```
## 'stat_bin()' using 'bins = 30'. Pick better value with 'binwidth'.
```



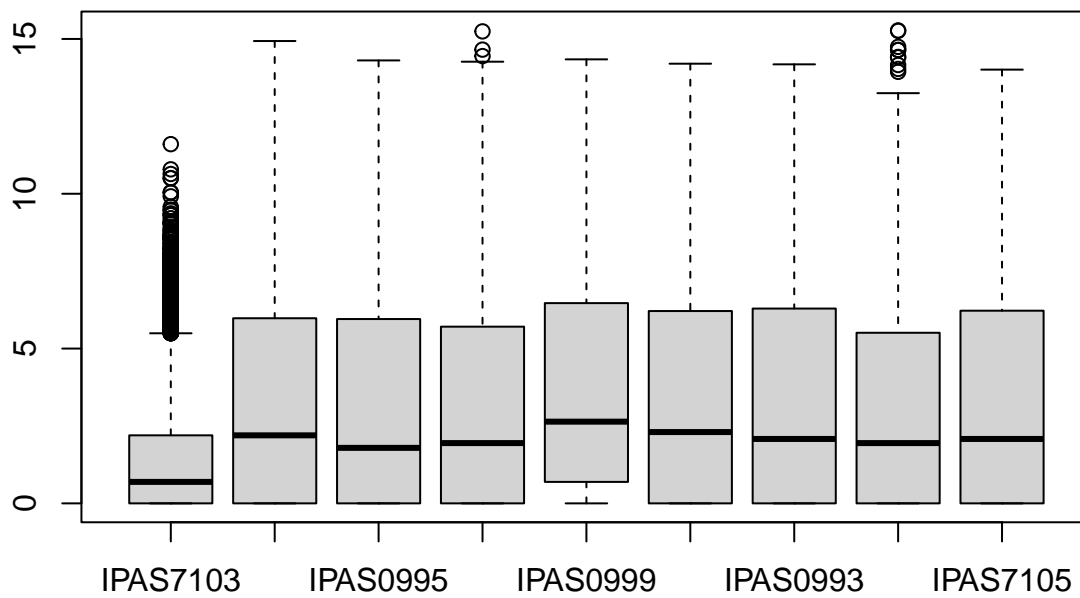
### tag-wise distribution

```
sample_filter <- sample(1:nrow(count_df), 100)
density_plot_dat <- t(count_df[sample_filter,]) %>% as_tibble() %>% pivot_longer(cols = everything(), n
plot <- ggplot(data = density_plot_dat, aes(x=count, color=ensemble_id)) +
  geom_density(alpha = 0.2) +
  scale_x_continuous(breaks=0:10, limits=c(0, 10)) +
  theme(legend.position="none")
suppressWarnings(print(plot))
```



sample-wise distribution

```
boxplot(log(count_df+1))
```



## Section 1 data processing

### do the conversion using biomaRt

the result is better than using org.Hs.eg.db no missing gene symbol

! there are some Ensembl gene id cannot be found by biomaRt <https://support.bioconductor.org/p/111608/>

```
# Uncomment the following code if you want to get the dictionary for mapping ensemble ID to gene symbol
# mart <- biomaRt::useDataset("hsapiens_gene_ensembl", useMart("ensembl"))
# dict_ensembl_to_symbol <- biomaRt::getBM(filters= "ensembl_gene_id", attributes= c("ensembl_gene_id",
# dict_ensembl_to_symbol$gene_length=dict_ensembl_to_symbol$end_position - dict_ensembl_to_symbol$start_position
# dict_ensembl_to_symbol[dict_ensembl_to_symbol == ''] <- NA
# save(dict_ensembl_to_symbol, file='cache/dict_ensemble_to_symbol.rda')
load('cache/dict_ensemble_to_symbol.rda')
head(dict_ensembl_to_symbol)
```

```
##   ensembl_gene_id hgnc_symbol start_position end_position gene_length
## 1 ENSG00000000003      TSPAN6    100627108    100639991     12883
## 2 ENSG000000000419      DPM1      50934867     50959140     24273
## 3 ENSG000000000457      SCYL3    169849631    169894267     44636
## 4 ENSG000000000460    C1orf112    169662007    169854080    192073
## 5 ENSG000000000938      FGR      27612064     27635185     23121
## 6 ENSG000000000971      CFH      196651754    196752476    100722
```

```
write_csv(dict_ensembl_to_symbol, 'support-data/dict_ensembl_to_symbol.csv')
```

```
# code below can be used to check those unmapped ensemble gene id
# count$Ensemble_ID[!count$Ensemble_ID %in% dict_ensembl_to_symbol$ensembl_gene_id]
```

### filtering unmapped ensemble gene id

```
count <- count %>% filter(Ensemble_ID %in% dict_ensembl_to_symbol$ensembl_gene_id)
```

### recreate data.frame and matrix

after several necessary pre-processing create data.frame and matrix version of the data

```
count_df <- as.data.frame(count[-1])
rownames(count_df) <- count$Ensemble_ID
count_mat <- as.matrix(count_df)
```

### normalization: calculate TPM (transcripts per million)

```
gene_length_df <- dict_ensembl_to_symbol %>% dplyr::select(ensembl_gene_id, gene_length) %>% unique()
gene_length_df <- data.frame(length=gene_length_df$gene_length, row.names=gene_length_df$ensembl_gene_id)
count_TPM_mat <- TPM(count_mat, gene_length_df)
head(count_TPM_mat)
```

```
##          IPAS7103  IPAS0972  IPAS0995  IPAS0982  IPAS0999
## ENSG000000000003  0.000000  148.723868  127.9468305  54.739058  100.699035
## ENSG000000000419  7.1717515  44.083160  169.8502069  28.865822  47.934309
## ENSG000000000457  27.2999253  26.388357  58.5261806  12.779419  48.473307
## ENSG000000000460  9.7520213  4.107075  11.4946052  2.070989  5.264838
## ENSG000000000938  0.9034899  37.185538  5.7660080  7.881675  98.758956
## ENSG000000000971  0.0000000  4.072463  0.1147632  17.731742  4.851123
##          IPAS7100  IPAS0993  IPAS0981  IPAS7105
## ENSG000000000003  20.516786  129.648564  37.334118  130.934586
## ENSG000000000419  60.622482  87.105952  29.002314  92.134711
## ENSG000000000457  25.503556  30.335995  6.555727  28.402952
## ENSG000000000460  3.163218  6.197081  1.173263  6.298571
## ENSG000000000938  865.413082  143.305795  17.820355  138.584587
## ENSG000000000971  31.742405  8.643628  13.801552  10.639516
```

```
save(count_TPM_mat, file='cache/count_TPM.rda')
write.csv(as.data.frame(count_TPM_mat), 'RNA-data/processed_RNA_data.csv')
```

recheck sample-level expression distribution after TPM normalization

```
boxplot(as.data.frame(log(count_TPM_mat+1)))
```

