Preliminary RNAseq analysis

Jieun Jeong

2025-02-10

This example analyzes RNAseq data from GEO data series GSE263611. In this project, four cell lines of cardiomyocytes derived from patients' blood are cultured in two ways: control and treated with IFNg. Thus we observe expression variability stemming from initial genetic and epigenetic differences and from INFg treatment.

Input tables for processing (count data, meta data) are created with separate script, shown in another document. The original read count data uses Ensembl gene identifiers used in mapping fragments to genes in this project. We convert identifiers as needed later and reformat as a matrix with row names.

```
library(tinytex)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
pa <- function(x) file.path('/Users/jieun/Work/Git_test/RNAseq_limma',x)
# making input tables
hugo <- readRDS(pa('HUGO.RDS'))
samples <- readRDS(pa('GSE263611_key.RDS')) %>% filter(Assay == "RNA-seq")
print(samples)
```

```
##      Accession Treatment   Assay CellLine
## S26 GSM8195226   Control RNA-seq  UDID006
## S27 GSM8195227      IFNg RNA-seq  UDID006
## S28 GSM8195228   Control RNA-seq  UDID076
## S29 GSM8195229      IFNg RNA-seq  UDID076
## S30 GSM8195230   Control RNA-seq  UDID088
## S31 GSM8195231      IFNg RNA-seq  UDID088
## S32 GSM8195232   Control RNA-seq  UDID148
## S33 GSM8195233      IFNg RNA-seq  UDID148
```

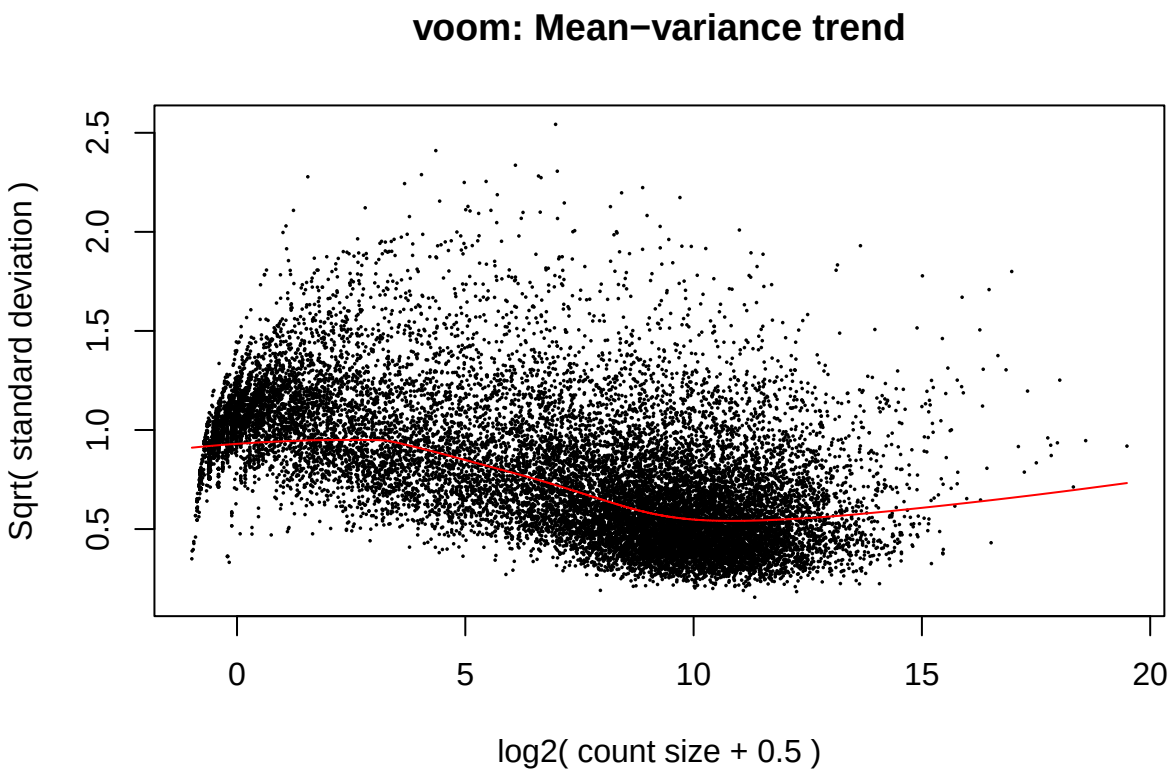
```
types_of_interest <- c("gene with protein product", "pseudogene", "RNA,
                       long non-coding")
counts <- readRDS(pa('GSE263611_counts.RDS')) %>% left_join(hugo, by="Ensembl") %>%
  filter(Type %in% types_of_interest) %>% select(-Type)

countsAS <- counts %>% filter(Symbol != "") %>% distinct(Symbol, .keep_all = T) %>%
  select(-Entrez, -Ensembl)
countsAS <- as.matrix(column_to_rownames(countsAS, "Symbol"))

countsN <- counts %>% filter(Entrez != "") %>% distinct(Entrez, .keep_all = T) %>%
  select(-Symbol, -Ensembl)
countsN <- as.matrix(column_to_rownames(countsN, "Entrez"))
```

As the next step, we normalize count data and check the distribution of values.

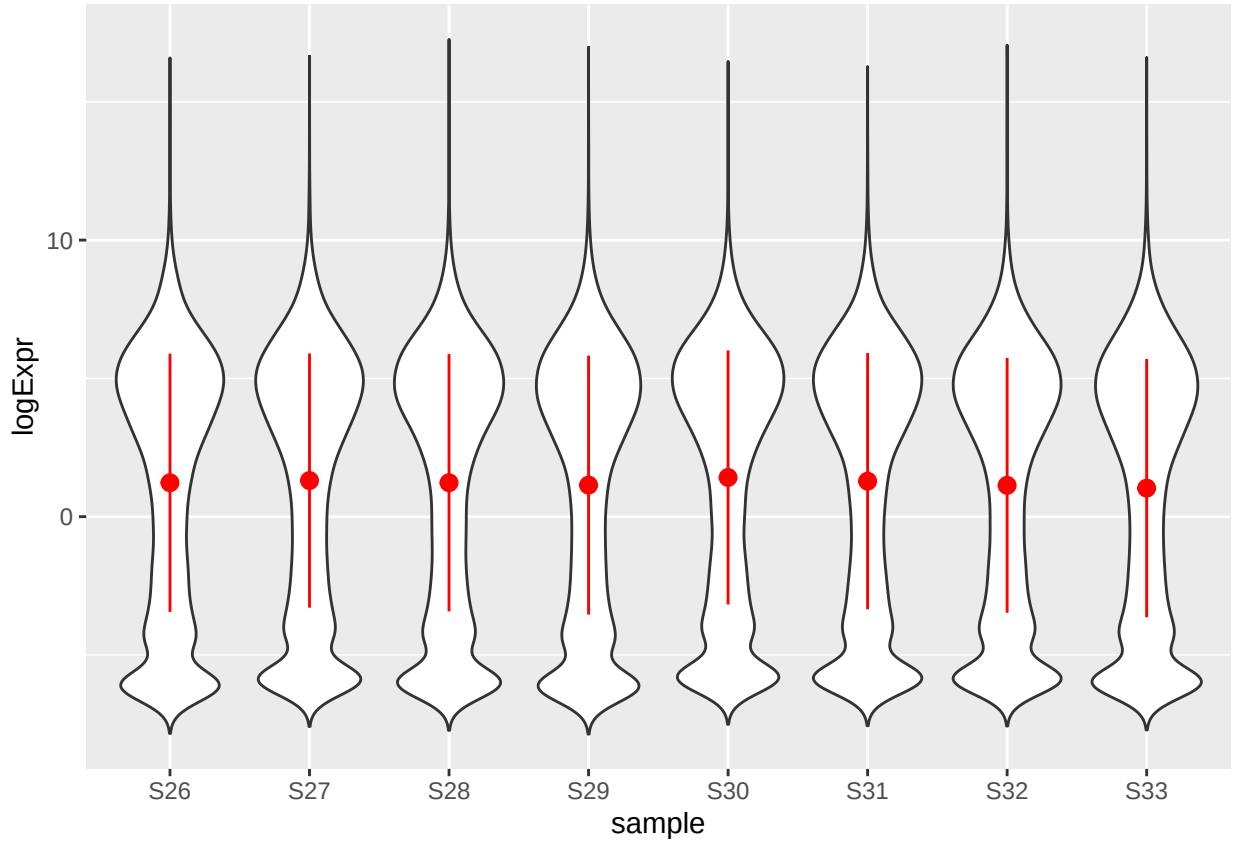
```
library(edgeR)
```

```
## Loading required package: limma
```

```
### Definitions for limma
treatment <- as.factor(samples$Treatment)
cell_line <- as.factor(samples$CellLine)
design <- model.matrix(~ 0 + treatment) # another design will be used too
contrast <- makeContrasts(treatmentIFNg - treatmentControl, levels = design)

# Normalization for limma
dge <- DGEList(counts = countsAS) # TMM normalization, reformmating
dge <- calcNormFactors(dge)
v <- voom(countsAS, design, plot = T) # log normalization and adjustment
```




```
# checking the normalized distribution of gene expression
A <- as.data.frame(v$E)
A <- rownames_to_column(A,var='gene')
AL <- pivot_longer(A,-gene,names_to = "sample", values_to = "logExpr")
AL$sample <- as.factor(AL$sample)
p <- ggplot(AL,aes(x=sample,y=logExpr)) + geom_violin(trim=F,width=0.8)
p + stat_summary(fun.data=mean_sdl, geom="pointrange", color="red",fun.args=list(mult=1))
```



Other useful visualization give highlights of the effect of the treatment, i.e. INFg. First we compute a new table

```
### Clustering on v$E
library(SummarizedExperiment)
```

```
## Loading required package: MatrixGenerics
```

```
## Loading required package: matrixStats
```

```
##
## Attaching package: 'matrixStats'
```

```
## The following object is masked from 'package:dplyr':
##
##     count
```

```
##
## Attaching package: 'MatrixGenerics'
```

```
## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
```

```
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: GenomicRanges

## Loading required package: stats4

## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following object is masked from 'package:limma':
##
##     plotMA

## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union

## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, saveRDS, setdiff,
##     table, tapply, union, unique, unsplit, which.max, which.min

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'
```

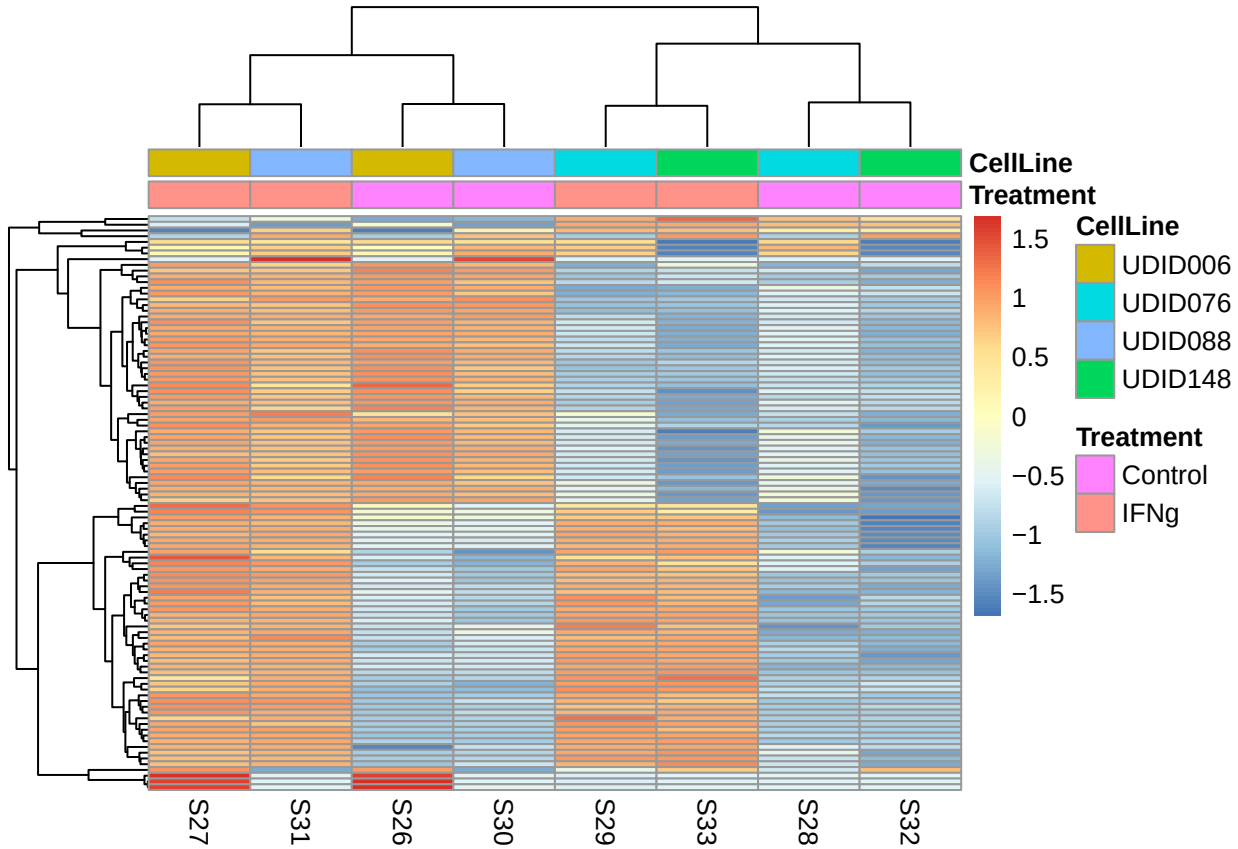
```
## The following objects are masked from 'package:lubridate':
##
##     second, second<-

## The following objects are masked from 'package:dplyr':
##
##     first, rename

## The following object is masked from 'package:tidyr':
##
##     expand

## The following object is masked from 'package:utils':
##
##     findMatches

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following object is masked from 'package:lubridate':
##
##     %within%

## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice

## The following object is masked from 'package:purrr':
##
##     reduce

## Loading required package: GenomeInfoDb

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'Biobase'
```

```
## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians
```

```
sample_info <- samples %>% select(-Assay, -Accession)
se <- SummarizedExperiment(assays = list(counts = v$E),
                           colData = sample_info)
colData = as.data.frame(colData(se))
```

then we see the heatmap

```
library(pheatmap)
# Select the top 100 most variable genes among the samples
VG <- apply(v$E, 1, var)
selectedGenes <- names(VG[order(VG, decreasing = T)])[1:100]

p <- pheatmap(v$E[selectedGenes, ],
         scale = 'row',
         show_rownames = FALSE,
         annotation_col = colData)
```

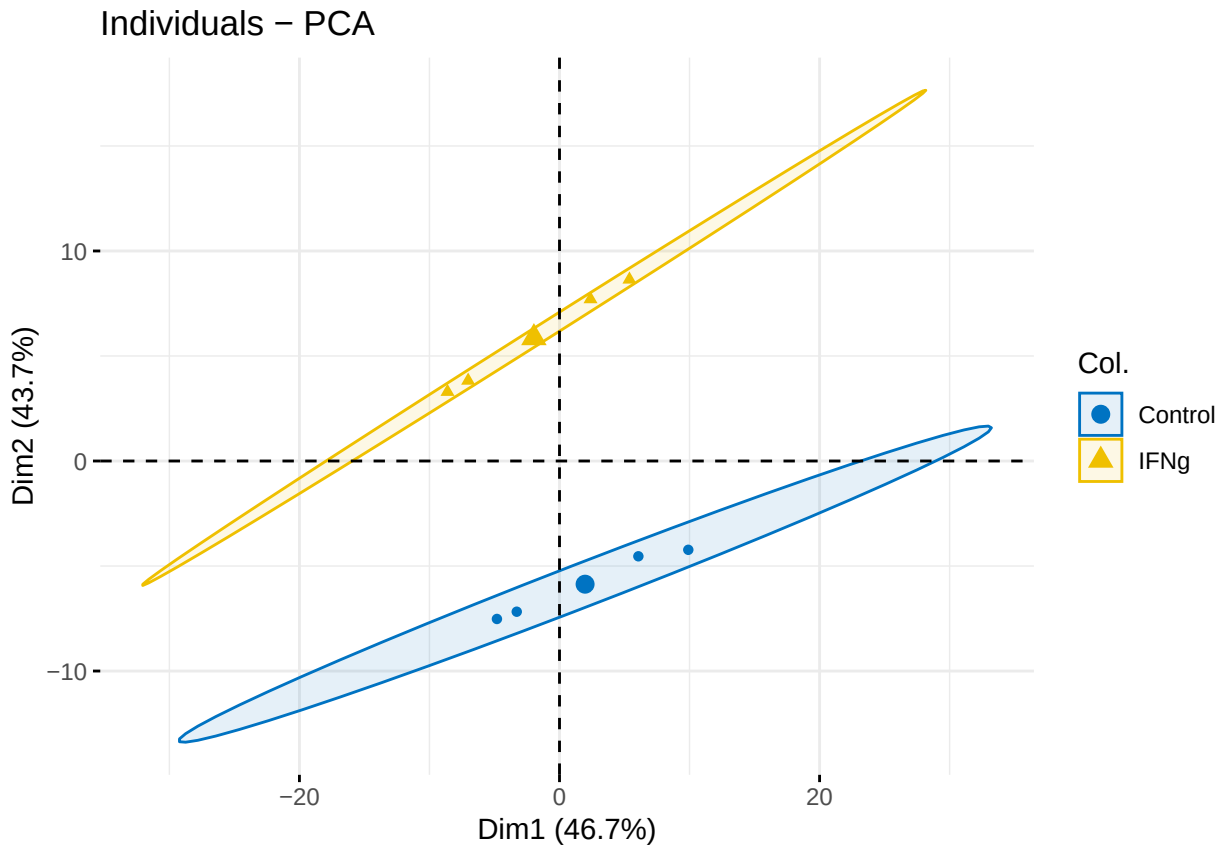


and finally, PCA results

```
library(factoextra) # for PCA
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
df_t <- t(v$E)

# Using only most variable genes
pr1 <- prcomp(df_t[,selectedGenes], scale. = TRUE)

p <- fviz_pca_ind(pr1,
             geom.ind = "point",
             col.ind = colData$Treatment,
             palette = "jco",
             addEllipses = TRUE,
             legene.title = "Treatment")
p
```



The heatmap shows that some genes have high fold change explained by Treatment and some my the difference between cell lines UDID006 and UDID088 (S26, S27, S30, S31) and the other two. PCA plot groups by Treatment, but we can also check how cell line look in two principal PCA components

```
pca_coords <- as.data.frame(pr1$x)
pca_coords$Sample <- rownames(pca_coords)
pca_coords$CellLine <- colData$CellLine
pca_coords$Treatment <- colData$Treatment  # Add treatment groups
```

```r
pca_mid <- pca_coords %>%
  group_by(CellLine) %>%
  summarise(
    x_mid = mean(PC1),
    y_mid = mean(PC2)
  )
p <- ggplot(pca_coords, aes(x = PC1, y = PC2, color = Treatment)) +
  geom_point(size = 4) +
  # stat_ellipse(type = "norm", level = 0.95) +  # 95% confidence ellipses
  geom_line(aes(group = CellLine), color = "gray") +
  geom_text(data = pca_mid, aes(x=x_mid, y=y_mid, label=CellLine), color="black", size=3) +

  xlim(-10, 10) + ylim(-10, 10) +
  theme_minimal()
p
```