Report jan 21, 2021 (21:14:54)

Normalization of counts in TMM (Trimmed Average of M), removal of the effect of batches and analysis of main components (PCA)

Jean Silva de Souza[1] and Mauro A. A. Castro[1]

[1]Bioinformatics and Systems Biology Lab, Federal University of Paraná, Curitiba, 81520-260, Brazil.

E-mails: jean.souza@edu.unipar.br

Contents

1. Introduction

After the RNA-seq data pre-processing step, some exploratory analyzes are applied in order to verify the efficiency of the previous steps. The output file generated in the RNA-seq data pre-processing step, using the `tximeta` tool (Love *et al.*, 2019) as an importer is a SummarizedExperiment object. It contains three matrices (counts, abundances and length of genes) and information about the samples and genes identified.

To visualize the distribution of data obtained from pre-processing, we applied three exploratory analyzes to the counting matrix: normalization of counts, removal of the effect of batches and analysis of main components.

2. Normalization of counts

There are two main factors to consider when quantifying and normalizing RNA-seq data: gene length and sequencing depth (Kadota *et al.*, 2012). The gene length factor is evaluated by comparing different genes in the same sample. Whereas, the sequencing depth factor is used in the evaluation of genes in different samples. Therefore, the objective of normalization is to remove the technical effects generated in the data, ensuring a better biological result (Robinson and Oshlack, 2010).

In this pipeline, we consider the sequencing depth factor. We use the TMM normalization (trimmed mean of M values) proposed by Robinson and Oshlack considering the size of the library (Robinson and Oshlack, 2010).

The script used to normalize the gene expression matrix count data is described below.

```
# - Desktop
setwd("~/pipeline_RNAseq_normalizacaoTMM_remEfLote_PCA_data-master/
      pipeline_RNAseq_normalizacaoTMM_remEfLote_PCA")

load("~/pipeline_RNAseq_normalizacaoTMM_remEfLote_PCA_data-master/
     pipeline_RNAseq_normalizacaoTMM_remEfLote_PCA/data_gexp.RData")

# - Objects for section 4.1. and 4.3.

# -- Separating the count data from the gene expression matrix
gexp.counts.brut <- assay(gse)

# -- Standardization in TMM
library(edgeR)
dge <- DGEList(counts=gexp.counts.brut)
dge <- calcNormFactors(dge)
gexp.TMM <- cpm(dge)

# -- Substitution of gene expression matrix count data
gse.TMM <- gse
gse.TMM@assays@data$abundance <- gexp.TMM

# -- Removal of low expression genes
keep <- rowSums(gse.TMM@assays@data$abundance > 0)/ncol(gse.TMM)
gse.TMM <- gse.TMM[keep>0.2,]
nrow(gse.TMM)
```

3. Batch effect removal

The sequencing technology and sample preparation have limitations, so there are technical variations in the sequencing between readings from different batches (Fei and Yu, 2020) called batch effect, which can lead to false scientific discoveries (Hicks *et al.*, 2017). However, studies have indicated that correcting the effect of batches can produce better results (Fei *et al.*, 2018; Hicks *et al.*, 2017).

Among several methods of removing the effect of batches, ComBat is a method based on an empirical structure of Bayes (Johnson *et al.*, 2007), and has been shown to be efficient in removing these technical variations (Stein *et al.*, 2015). Therefore, we use the ComBat method available in the tool R `sva` (Leek *et al.*, 2019) for batch correction.

```
library(sva)
# - Objects for section 4.2 and 4.4
# -- Batch effect removal - normalized TMM counts
mat <- gse.TMM@assays@data$abundance
colAnnotation <- colData(gse.TMM)
modcombat <- model.matrix(~Risk + Condition, data = colAnnotation)
mat <- ComBat(dat = mat,
              batch = colAnnotation$Batch,
```

```
              mod = modcombat,
              prior.plots = TRUE)
gse.TMM.semEfLote <- gse.TMM
gse.TMM.semEfLote@assays@data$abundance <- mat

# -- Batch effect removal - non-normalized counts
mat <- gse.TMM@assays@data$counts
colAnnotation <- colData(gse.TMM)
modcombat <- model.matrix(~Risk + Condition, data = colAnnotation)
mat <- ComBat(dat = mat,
              batch = colAnnotation$Batch,
              mod = modcombat,
              prior.plots = TRUE)
gse.TMM.semEfLote@assays@data$counts <- mat
```

4. Principal component analysis (PCA)

The interpretation of large data sets requires methods that reduce their size in an interpretable way, preserving most of the information (Jolliffe and Cadima, 2016). The Principal Component Analysis (PCA) technique was developed for this purpose and is widely used. PCA is a multivariate analysis technique used to present the relationship between many variables, it summarizes the original information in a smaller set of statistical variables (components), minimizing the loss of information (Jolliffe and Cadima, 2016).

We apply the PCA to the gene expression matrix counts using the prcomp function of the R `stats` package (Team, 2019). We made several PCAs in order to identify the difference in the distribution of normalized and non-normalized data; with and without batch effect, considering the experimental design of the cohort used - data_gexp: condition B and STR.

4.1. PCA - non-normalized counts with batch effect

We used the raw counts of the gene expression matrix of the study cohort. We did not remove the effect of lots because we wanted to study the original distribution of the data.

```
# -- PCA - non-normalized counts with batch effect
dataset <- gse.TMM@assays@data$counts

# Takes variance of each gene
rv <- rowVars(dataset)

# Select the ntop genes by variance
ntop=500
select <- order(rv, decreasing=TRUE)[seq_len(min(ntop, length(rv)))]

# Perform a PCA on the data in assay(x) for the selected genes
pca <- prcomp(t(dataset[select,]), scale. = TRUE, center = TRUE, retx = TRUE)

# Prepare data for plotting
colAnnotation <- as.data.frame(colData(gse.TMM))
colAnnotation$Batch <-  as.factor(colAnnotation$Batch)
pcascores <- pca$x
all(rownames(pcascores)==rownames(colAnnotation))
```

```r
# [1] TRUE
pcaData2 <- cbind(pcascores[,1:2],colAnnotation[,c("Batch","Condition")])
pcaData2$group <- paste(pcaData2$Batch, pcaData2$Risk, sep=":")

# Rescale continuous vector to have specified minimum and maximum
library(scales)
pcaData2$PC1 <- rescale(pcaData2$PC1, to=c(-50,50))
pcaData2$PC2 <- rescale(pcaData2$PC2, to=c(-50,50))

# Plot with ggplot
library(ggplot2)
pdf(file = "~/pipeline_RNAseq_normalizacaoTMM_remEfLote_PCA_data-master/
    pipeline_RNAseq_normalizacaoTMM_remEfLote_PCA/figures_PCA/
    PCA_gross_with_batch_effect.pdf", width = 4.5, height = 4.5)
ggplot(pcaData2, aes(x = PC1, y = PC2, shape = Condition)) +
  geom_point(size = 2, aes(color = Batch)) +
  xlab("PC1 ") +
  ylab("PC2 ") +
  coord_fixed() +
  ggtitle("Non-normalized counts\nwith batch effect") +
  theme_light() +
  guides(color=guide_legend(title="Batch")) +
  guides(shape=guide_legend(title="Condition")) +
  theme(plot.title = element_text(size = 14, family = "Helvetica"),
        text =element_text(size = 16, family = "Helvetica"))+
  scale_colour_brewer(palette = "Set1")
dev.off()
```

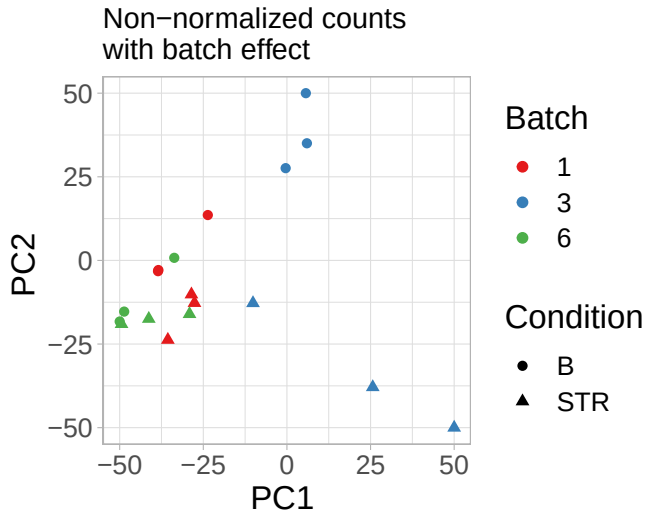

Figure 1: PCA - non-normalized counts with lots effect. The colors indicate the lots and the shapes represent the conditions

4.2. PCA - non-normalized counts without batch effect

In order to study the removal of technical variables in stages, in this section we chose to remove the effect of batches, however we did not apply the normalization of the counts.

```
# - PCA - non-normalized counts without batch effect
dataset <- gse.TMM.semEfLote@assays@data$counts

# Takes variance of each gene
rv <- rowVars(dataset)

# Select the ntop genes by variance
ntop=500
select <- order(rv, decreasing=TRUE)[seq_len(min(ntop, length(rv)))]

# Perform a PCA on the data in assay(x) for the selected genes
pca <- prcomp(t(dataset[select,]), scale. = TRUE, center = TRUE, retx = TRUE)

# Prepare data for plotting
colAnnotation <- as.data.frame(colData(gse.TMM.semEfLote))
colAnnotation$Batch <-  as.factor(colAnnotation$Batch)
pcascores <- pca$x
all(rownames(pcascores)==rownames(colAnnotation))
# [1] TRUE
pcaData2 <- cbind(pcascores[,1:2],colAnnotation[,c("Batch","Condition")])
pcaData2$group <- paste(pcaData2$Batch, pcaData2$Risk, sep=":")

# Rescale continuous vector to have specified minimum and maximum
library(scales)
pcaData2$PC1 <- rescale(pcaData2$PC1, to=c(-50,50))
pcaData2$PC2 <- rescale(pcaData2$PC2, to=c(-50,50))

# Plot with ggplot
pdf(file = "~/pipeline_RNAseq_normalizacaoTMM_remEfLote_PCA_data-master/
    pipeline_RNAseq_normalizacaoTMM_remEfLote_PCA/figures_PCA/
    PCA_no_batch_effect.pdf", width = 4.5, height = 4.5)
ggplot(pcaData2, aes(x = PC1, y = PC2, shape = Condition)) +
  geom_point(size = 2, aes(color = Batch)) +
  xlab("PC1 ") +
  ylab("PC2 ") +
  coord_fixed() +
  ggtitle("Non-normalized counts\nwithout batch effect") +
  theme_light() +
  guides(color=guide_legend(title="Batch")) +
  guides(shape=guide_legend(title="Condition")) +
  theme(plot.title = element_text(size = 14, family = "Helvetica"),
        text =element_text(size = 16, family = "Helvetica"))+
  scale_colour_brewer(palette = "Set1")
dev.off()
```

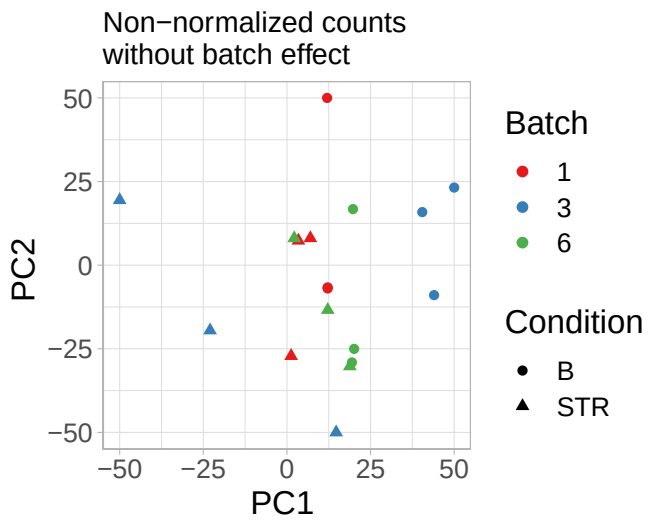

Figure 2: PCA - non-normalized counts with no effect of batches. The colors indicate the lots and the shapes represent the conditions.

4.3. PCA - normalized counts with batch effect

In this section we apply normalization to the counts, but we do not remove the effect of batches, in order to evaluate only the effect of normalization on the data.

```
# - PCA - normalized counts with batch effect
dataset <- gse.TMM@assays@data$abundance

# Takes variance of each gene
rv <- rowVars(dataset)

# Select the ntop genes by variance
ntop=500
select <- order(rv, decreasing=TRUE)[seq_len(min(ntop, length(rv)))]

# Perform a PCA on the data in assay(x) for the selected genes
pca <- prcomp(t(dataset[select,]), scale. = TRUE, center = TRUE, retx = TRUE)

# Prepare data for plotting
colAnnotation <- as.data.frame(colData(gse.TMM))
colAnnotation$Batch <-  as.factor(colAnnotation$Batch)
pcascores <- pca$x
all(rownames(pcascores)==rownames(colAnnotation))
# [1] TRUE
pcaData2 <- cbind(pcascores[,1:2],colAnnotation[,c("Batch","Condition")])
pcaData2$group <- paste(pcaData2$Batch, pcaData2$Risk, sep=":")

# Rescale continuous vector to have specified minimum and maximum
library(scales)
```

```
pcaData2$PC1 <- rescale(pcaData2$PC1, to=c(-50,50))
pcaData2$PC2 <- rescale(pcaData2$PC2, to=c(-50,50))

# Plot with ggplot
pdf(file = "~/pipeline_RNAseq_normalizacaoTMM_remEfLote_PCA_data-master/
    pipeline_RNAseq_normalizacaoTMM_remEfLote_PCA/figures_PCA/
    PCA_normalized_with_batch_effect.pdf", width = 4.5, height = 4.5)
ggplot(pcaData2, aes(x = PC1, y = PC2, shape = Condition)) +
  geom_point(size = 2, aes(color = Batch)) +
  xlab("PC1 ") +
  ylab("PC2 ") +
  coord_fixed() +
  ggtitle("Normalized counts\nwith batch effect") +
  theme_light() +
  guides(color=guide_legend(title="Batch")) +
  guides(shape=guide_legend(title="Condition")) +
  theme(plot.title = element_text(size = 14, family = "Helvetica"),
        text =element_text(size = 16, family = "Helvetica"))+
  scale_colour_brewer(palette = "Set1")
dev.off()
```

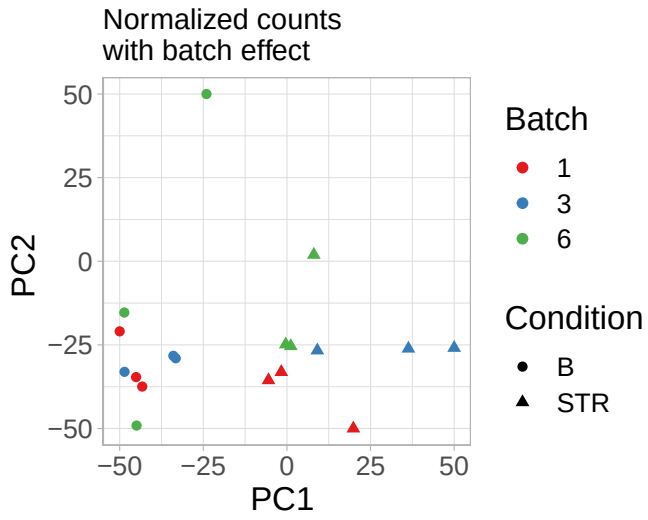

Figure 3: PCA - normalized counts with batch effect. The colors indicate the lots and the shapes represent the conditions.

4.4. PCA - normalized counts without batch effect

We removed the effect of batches from normalized counts. In this way, we remove technical variations while preserving the biological effect.

```r
# - PCA - normalized counts without batch effect
dataset <- gse.TMM.semEfLote@assays@data$abundance

# Takes variance of each gene
rv <- rowVars(dataset)

# Select the ntop genes by variance
ntop=500
select <- order(rv, decreasing=TRUE)[seq_len(min(ntop, length(rv)))]

# Perform a PCA on the data in assay(x) for the selected genes
pca <- prcomp(t(dataset[select,]), scale. = TRUE, center = TRUE, retx = TRUE)

# Prepare data for plotting
colAnnotation <- as.data.frame(colData(gse.TMM.semEfLote))
colAnnotation$Batch <-  as.factor(colAnnotation$Batch)
pcascores <- pca$x
all(rownames(pcascores)==rownames(colAnnotation))
# [1] TRUE
pcaData2 <- cbind(pcascores[,1:2],colAnnotation[,c("Batch","Condition")])
pcaData2$group <- paste(pcaData2$Batch, pcaData2$Risk, sep=":")

# Rescale continuous vector to have specified minimum and maximum
library(scales)
pcaData2$PC1 <- rescale(pcaData2$PC1, to=c(-50,50))
pcaData2$PC2 <- rescale(pcaData2$PC2, to=c(-50,50))

# Plot with ggplot
pdf(file = "~/pipeline_RNAseq_normalizacaoTMM_remEfLote_PCA_data-master/
    pipeline_RNAseq_normalizacaoTMM_remEfLote_PCA/figures_PCA/
    PCA_normalized_without_batch_effect.pdf", width = 4.5, height = 4.5)
ggplot(pcaData2, aes(x = PC1, y = PC2, shape = Condition)) +
  geom_point(size = 2, aes(color = Batch)) +
  xlab("PC1 ") +
  ylab("PC2 ") +
  coord_fixed() +
  ggtitle("Normalized counts\nwithout batch effects") +
  theme_light() +
  guides(color=guide_legend(title="Batch")) +
  guides(shape=guide_legend(title="Condition")) +
  theme(plot.title = element_text(size = 14, family = "Helvetica"),
        text =element_text(size = 16, family = "Helvetica"))+
  scale_colour_brewer(palette = "Set1")
dev.off()
```

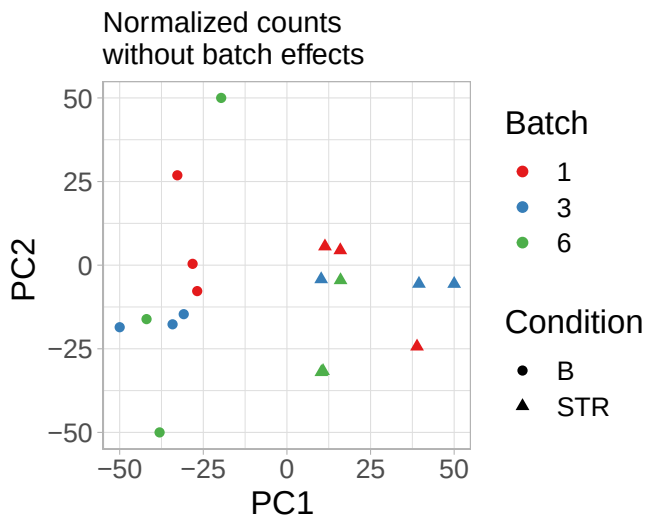

Figure 4: PCA - normalized counts without batch effect. The colors indicate the lots and the shapes represent the conditions.

We saw in Figures 1 to 4 that the normalization of counts and the removal of the batch effect, preserved the biological effect by separating conditions B and STR.

5. Session information

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.1 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.0.3  magrittr_2.0.1  tools_4.0.3     htmltools_0.5.1
##  [5] yaml_2.2.1      stringi_1.5.3   rmarkdown_2.6   knitr_1.30
##  [9] stringr_1.4.0   xfun_0.20       digest_0.6.27   rlang_0.4.10
## [13] evaluate_0.14
```

6. References

Fei,T. and Yu,T. (2020) ScBatch: Batch-effect correction of rna-seq data through sample distance matrix adjustment. *Bioinformatics*, 1–10.

Fei,T. *et al.* (2018) Mitigating the adverse impact of batch effects in sample pattern detection. *Bioinformatics*, **34**, 2634–2641.

Hicks,C.,Stephanie *et al.* (2017) Missing data and technical variability in single-cell rna-sequencing experiments. *Biostatistics*, 1–17.

Johnson,E.,W. *et al.* (2007) Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, **8**, 118–127.

Jolliffe,T.,Ian and Cadima,J. (2016) Principal component analysis:A review and recent developments. *Phil. Trans. R. Soc.*, **374**, 1–16.

Kadota,K. *et al.* (2012) A normalization strategy for comparing tag count data. *Algorithms for Molecular Biology*, **7**, 1–13.

Leek,T.,Jeffrey *et al.* (2019) Sva: Surrogate variable analysis. *R package version 3.34.0*.

Love,M.I. *et al.* (2019) Tximeta: Reference sequence checksums for provenance identification in rna-seq. *bioRxiv*.

Robinson,D.,Mark and Oshlack,A. (2010) A scaling normalization method for differential expression analysis of rna-seq data. *Genome Biology*, **11**, 1–9.

Stein,K.,Caleb *et al.* (2015) Removing batch effects from purified plasma cell gene expression microarrays with modified combat. *BMC Bioinformatics*, **16**, 1–9.

Team,R.C. (2019) R: A language and environment for statistical computing. *R package version 3.34.0*.