Count normalization for Heaney et al. 2021

Sanjeev V Namjoshi (snamjoshi87@utexas.edu)

February 23rd, 2017 (revised June 15th, 2019)

Here we annotate our data in official gene symbol format and apply DESeq2 factor size normalization across all our libraries. We also perform filtering and some visualization of the count data before the final export.

Load packages

```
library(DESeq2)
library(AnnotationDbi)
library(org.Mm.eg.db)
library(HTSFilter)
library(magrittr)
library(ggplot2)
library(reshape2)
library(ggthemes)
```

Functions

The `squash()` function simply scales any data between 0-100 for easier visualization and comparison.

```
squash <- function(x) {
  output <- ((x - min(x)) / (max(x) - min(x))) * 100
  return(output)
}
```

The `percentChange()` function calculates the percent a measure changes from its baseline.

```
percentChange <- function(new, old) {
  return(((new - old) / abs(old)) * 100)
}
```

Count filtering

We used **HTSFilter 1.14.1** to filter the data using the standard settings.

```
countData <- read.csv(file = "combinedCounts.csv", header = TRUE)
rownames(countData) <- countData[ ,1]
countData[ ,1] <- NULL
```

```
countData$RO_KO1 <- countData$RO_KO2
countData$RO_KO3 <- countData$RO_KO2

cdMain <- subset(countData, select = c(16:18, 13:15, 10:12, 3, 22, 23, 7:9, 4:6))
conds <- sapply(names(cdMain), function(x) substr(x, 1, nchar(x)-1)) %>% unname

filter <- HTSFilter(x = cdMain,
                    conds = conds,
                    normalization = "none")

cdFilter <- as.data.frame(filter$filteredData)
cdFilter[ ,c(11, 12)] <- NULL   # Remove duplicated Ro KO columns
```

Annotation (official gene symbol)

Annotation weas performed with the Bioconductor packages **org.Mm.eg.db** and **AnnotationDbi**. Any sequences that did not map to a gene qualifier were filtered out of the dataset.

```
cdFilter$symbol <- mapIds(org.Mm.eg.db,
                      keys = row.names(cdFilter),
                      column = "SYMBOL",
                      keytype = "ENSEMBL",
                      multiVals = "first")

### Remove rows with no gene mapping
cdFilter <- subset(cdFilter, select = c(17, 1:16))
cdFilter <- cdFilter[complete.cases(cdFilter), ]
rownames(cdFilter) <- make.names(cdFilter$symbol, unique = TRUE)
cdFilter[ ,1] <- NULL
```

Count normalization (DESeq2 factor size)

First we create a metatable to organize the data. Next, the data is loaded into **DESeq2**. Since DESeq2 scaling applies across the entire library, it inflates the counts from our KO libraries. To fix this, we scale back the KO counts using ratios between WT and KO counts obtained from the BioAnalyzer results prior to sequencing. Then, all counts were collapsed by averaging across rows.

```
### Create metatable
metaTable <- data.frame(LibraryName = names(cdFilter),
                        LibraryLayout = rep("SINGLE", 16),
                        condition = c(rep("WT", 3), rep("KO", 3), rep("WT", 3), "KO", rep("WT", 3), rep
rownames(metaTable) <- metaTable[ ,1]   # Set first column to row names
metaTable[ ,1] <- NULL

### Get normalized counts
dd <- DESeqDataSetFromMatrix(countData = cdFilter,
                             colData = metaTable,
                             design = ~ 1)
dd <- estimateSizeFactors(dd)
nc <- as.data.frame(counts(dd, normalize = TRUE))

### Scale by BioAnalyzer values
```

```
BA <- read.csv("BA_values.csv", header = TRUE, stringsAsFactors = FALSE)
BA$mean <- rowMeans(BA[ ,2:4], na.rm = TRUE)

scale <- c(RO = BA$mean[2] / BA$mean[5],
           RO_RAP = BA$mean[3] / BA$mean[4],
           SAL = BA$mean[6] / BA$mean[7])

cdFilterScaled <- nc

cdFilterScaled[ ,10] <- cdFilterScaled[ ,10] / (scale[1] + 1)          # Scale Ro
cdFilterScaled[ ,14:16] <- cdFilterScaled[ ,14:16] / (scale[2] + 1)    # Scale Ro Rap
cdFilterScaled[ ,4:6] <- cdFilterScaled[ ,4:6] / (scale[3] + 1)        # Scale Sal

### Average across rows
nscMean <- data.frame(row.names = row.names(cdFilterScaled),
                      SAL_WT = rowMeans(cdFilterScaled[ ,1:3]),
                      SAL_KO = rowMeans(cdFilterScaled[ ,4:6]),
                      RO_WT = rowMeans(cdFilterScaled[ ,7:9]),
                      RO_KO = cdFilterScaled[ ,10],
                      RO_RAP_WT = rowMeans(cdFilterScaled[ ,11:13]),
                      RO_RAP_KO = rowMeans(cdFilterScaled[ ,14:16])) %>% squash()
```

Visualize normalized counts (figure not rendered, see FigureS2.Rmd)

For count visualization, we first plotted the WT and KO counts be each treatment

```
### Create variables for plotting with ggplot2
nscMean.m <- melt(nscMean)
nscMean.m$Genotype <- c(rep("WT", 14543), rep("KO", 14543), rep("WT", 14543), rep("KO", 14543), rep("WT
nscMean.m$variable <- c(rep("Saline", 29086), rep("Ro-25-6981", 29086), rep("Ro-25-6981 + Rapamycin", 2
nscMean.m$varOrder <- factor(nscMean.m$variable, levels = c("Saline", "Ro-25-6981", "Ro-25-6981 + Rapam

### Plot WT v. KO by treatment
ggplot(nscMean.m, aes(x = Genotype, y = value, color = Genotype)) +
  geom_point(position = position_jitter(width = 0.2, height = 0.2)) +
  facet_grid(~ varOrder) +
  xlab("") +
  ylab("Scaled Counts") +
  theme_few() +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())
```

Next, we plotted WT and KO counts with known FMRP targets and non-targets (from *Darnell et al. 2011 [1]*) (figure not rendered, see FigureS2.Rmd)

```
### Load targets and non-targets
targets <- c("Gabbr1", "Gabbr2","Nisch", "Prrc2a", "Kif1a", "Grin2a", "Adgrb2", "Cyfip2", "Map1b", "Bsn
nontargets <- c("Pabpc1", "Tmem65", "Hprt1", "St8sia3", "Sae1", "Glrb", "Gria2", "Tcerg1", "Eif3a", "Sl

### Prepare variables for plotting
nscRawTargets <- nscMean[row.names(nscMean) %in% c(targets, nontargets), ] %>% squash()
nscRawTargets$targets <- ifelse(row.names(nscRawTargets) %in% targets, "Target", "Non-target")
```

```
nscRawTargets.m <- melt(nscRawTargets)
nscRawTargets.m$Genotype <- c(rep("WT", 25), rep("KO", 25), rep("WT", 25), rep("KO", 25), rep("WT", 25)
nscRawTargets.m$variable <- c(rep("Saline", 50), rep("Ro-25-6981", 50), rep("Ro-25-6981 + Rapamycin", 5
nscRawTargets.m$group <- rep(1:25, 6)
nscRawTargets.m$varOrder <- factor(nscRawTargets.m$variable, levels = c("Saline", "Ro-25-6981", "Ro-25-
nscRawTargets.m$GenotypeOrder <- factor(nscRawTargets.m$Genotype, levels = c("WT", "KO"))

### Plot WT v. KO for FMRP targets and non-targets
ggplot(nscRawTargets.m, aes(x = GenotypeOrder, y = value, group = group, color = targets)) +
  geom_point() +
  geom_line() +
  facet_grid(~ varOrder) +
  xlab("") +
  ylab("Scaled Counts") +
  theme_few() +
  scale_color_discrete(name = "FMRP Targets")
```

Finally, for the percent changes between targets and non-targets reported in the paper, we simply calculated the percent changes across all the dataframe for the mean normalized (scaled) counts.

```
new <- nscRawTargets[nscRawTargets$targets == "Target", 1:6] %>% colMeans()
old <- nscRawTargets[nscRawTargets$targets == "Non-target", 1:6] %>% colMeans()

percentChange(new[1], old[1]) %>% round(2)
percentChange(new[3], old[3]) %>% round(2)
percentChange(new[5], old[5]) %>% round(2)
```

References

[1] Darnell, J.C., Van Driesche, S.J., Zhang, C., Hung, K.Y., Mele, A., Fraser, C.E., Stone, E.F., Chen, C., Fak, J.J., Chi, S.W., Licatalosi, D.D., Richter, J.D., Darnell, R.B. (2011). FMRP stalls ribosomal translocation on mRNAs linked to synaptic function and autism. *Cell* 146, 247-261.

Session info:

```
## R version 3.3.2 (2016-10-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.1 LTS
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
```

```
##
## other attached packages:
##  [1] ggthemes_3.3.0             reshape2_1.4.2
##  [3] ggplot2_2.2.0              magrittr_1.5
##  [5] HTSFilter_1.14.1           org.Mm.eg.db_3.4.0
##  [7] AnnotationDbi_1.36.0       DESeq2_1.14.0
##  [9] SummarizedExperiment_1.4.0 Biobase_2.34.0
## [11] GenomicRanges_1.26.1       GenomeInfoDb_1.10.1
## [13] IRanges_2.8.1              S4Vectors_0.12.0
## [15] BiocGenerics_0.20.0
##
## loaded via a namespace (and not attached):
##  [1] genefilter_1.56.0   locfit_1.5-9.1      splines_3.3.2
##  [4] lattice_0.20-34     colorspace_1.3-1    htmltools_0.3.5
##  [7] yaml_2.1.14         survival_2.40-1     XML_3.98-1.5
## [10] foreign_0.8-67      DBI_0.5-1           BiocParallel_1.8.1
## [13] RColorBrewer_1.1-2  plyr_1.8.4          stringr_1.1.0
## [16] zlibbioc_1.20.0     munsell_0.4.3       gtable_0.2.0
## [19] evaluate_0.10       memoise_1.0.0       labeling_0.3
## [22] latticeExtra_0.6-28 knitr_1.15.1        geneplotter_1.52.0
## [25] htmlTable_1.7       Rcpp_0.12.8         edgeR_3.16.4
## [28] acepack_1.4.1       xtable_1.8-2        scales_0.4.1
## [31] backports_1.0.4     limma_3.30.5        Hmisc_4.0-0
## [34] annotate_1.52.0     XVector_0.14.0      gridExtra_2.2.1
## [37] digest_0.6.10       DESeq_1.26.0        stringi_1.1.2
## [40] grid_3.3.2          rprojroot_1.1       tools_3.3.2
## [43] bitops_1.0-6        lazyeval_0.2.0      RCurl_1.95-4.8
## [46] tibble_1.2          RSQLite_1.1         Formula_1.2-1
## [49] cluster_2.0.5       Matrix_1.2-7.1      data.table_1.9.8
## [52] assertthat_0.1      rmarkdown_1.2       rpart_4.1-10
## [55] nnet_7.3-12
```