TPM_Normalization

Cera Fisher

September 14, 2018

Scaling TPM between two species with different numbers of annotated transcripts

based on Musser & Wagner (2015), JEZ:B

TPM, transcripts per million mapped, is a way of normalizing RNAseq data that accounts for differences in library size by scaling the abundance of a transcript (the "counts") to the total number of transcripts assumed to be present in the transcriptome. In short, TPM = count for transcript "i" / total number of annotated transcripts * a scaling factor.

Necessarily, TPM from one species does not map to the TPM of another species if their transcriptomes are of different sizes, which they almost certainly are, so the values for the species with the smaller transcriptome will be inflated relative to the species with the larger transcriptome. According to Musser & Wagner, these values can be rescaled by calculating a scaling factor α:

$$\alpha = 10^{-6} \sum_{j=1}^{N1} tpm(Aj)$$

Where N1 = the number of transcripts for species B (the smaller set), j = all the transcripts in the set, and tpm(Aj) is the transcripts per million for species A for each of the genes j in the set.

Here is how I interpret this to work in my transcriptomes for *Entylia carinata* and *Homalodisca vitripennis*.

1) Read in the un-normalized TPM values from RSEM/edgeR.

```
library("dplyr")
options(stringsAsFactors = FALSE)
Ecar.TPM <- read.table("C:/Users/cruth/Google Drive/Treehoppers/ResearchFiles/RNASeq/GeneExpression_2018
    sep = "\t", header = TRUE)
colnames(Ecar.TPM)
```

```
##  [1] "X"            "ECA_Abd"      "ECA_Ovi"      "ECB_Abd"      "ECC_Abd"
##  [6] "ECC_Ovi"      "ECEF_Abd"     "ECEF_Eye"     "ECEF_Leg"     "ECEF_Meso"
## [11] "ECEF_Ovi"     "ECEF_Pro"     "ECEF_Wing2"   "ECEF_Wing3"   "ECFisC_Abd"
## [16] "ECFisC_Eye"   "ECFisC_Leg"   "ECFisC_Meso"  "ECFisC_Ovi"   "ECFisC_Pro"
## [21] "ECFisC_Wing2" "ECFisC_Wing3" "ECLegs_A"     "ECLegs_B"     "ECLegs_C"
## [26] "ECMeso_A"     "ECMeso_B"     "ECMeso_C"     "ECPro_A"      "ECPro_B"
## [31] "ECPro_C"      "ECty4_Abd"    "ECty4_Eye"    "ECty4_Leg"    "ECty4_Meso"
## [36] "ECty4_Ovi"    "ECty4_Pro"    "ECty4_Wing2"  "ECty4_Wing3"  "ECWings_A"
## [41] "ECWings_B"    "ECWings_C"
```

```
colnames(Ecar.TPM)[1] <- "ECid"

Hvit.TPM <- read.table("C:/Users/cruth/Google Drive/Treehoppers/ResearchFiles/RNASeq/GeneExpression_2018
```

```
    sep = "\t", header = TRUE)
colnames(Hvit.TPM)
```

```
##  [1] "X"           "HV102_Abd"   "HV102_Eye"   "HV102_Leg"   "HV102_Meso"
##  [6] "HV102_Ovi"   "HV102_Pro"   "HV102_Wing2" "HV102_Wing3" "HV3a_Abd"
## [11] "HV3a_Eye"    "HV3a_Leg"    "HV3a_Meso"   "HV3a_Ovi"    "HV3a_Pro"
## [16] "HV3a_Wing2"  "HV3a_Wing3"  "HV7_Abd"     "HV7_Leg"     "HV7_Meso"
## [21] "HV7_Pro"     "HV7_Wing2"   "HVA_Abd"     "HVB_Abd"     "HVC_Ovi"
## [26] "HVLegs_A"    "HVLegs_B"    "HVLegs_C"    "HVMeso_A"    "HVMeso_B"
## [31] "HVMeso_C"    "HVPro_A"     "HVPro_B"     "HVPro_C"     "HVWings_A"
## [36] "HVWings_B"   "HVWings_C"
```

```
colnames(Hvit.TPM)[1] <- "HVid"
```

2) Calculate α for *H. vitripennis*:

```
HvitN1 <- as.numeric(length(Hvit.TPM$HVid))
# 19,126
EcarN2 <- as.numeric(length(Ecar.TPM$ECid))
# 19,975
sumN2.EcarTPM <- sum(Ecar.TPM[, 2:42])/41
sumN2.EcarTPM
```

```
## [1] 1e+06
```

```
## Sum of TPM for any given sample should, by definition, be 1,000,000 The average
## TPM for Ecar is the sum divided by the number of transcripts.
Ec.avg.TPM <- sumN2.EcarTPM/19975

## Getting the sum of Ecar TPM for the # of transcripts in Hvit's transcriptome
## i.e, multiplying the average times 19,126
sum.ECavgTPM.HvitN1 <- Ec.avg.TPM * HvitN1

# To get alpha, divide that amount by 1,000,000
alpha <- sum.ECavgTPM.HvitN1 * (10^(-6))

alpha
```

```
## [1] 0.9574969
```

This value for α, 0.957....etc is very close to the ratio of the smaller number of transcripts to the larger:

```
checksum <- HvitN1/EcarN2
checksum
```

```
## [1] 0.9574969
```

```
checksum - alpha
```

```
## [1] 7.006075e-09
```

Which perhaps should be expected, since those are the only values in this calculation that don't cancel out.

3) Calculating alpha for *Oncopeltus fasciatus*:

```
Ofas.TPM <- read.csv("C:/Users/cruth/Google Drive/Treehoppers/ResearchFiles/RNASeq/GeneExpression_2018/
    sep = "\t")

names(Ofas.TPM)[1] <- "OFid"
OfasN2 <- length(Ofas.TPM$OFid)
```

```
sum.ECavgTPM.OfasN2 <- Ec.avg.TPM * OfasN2

OFalpha <- sum.ECavgTPM.OfasN2 * (10^(-6))
```

Scaling Hvit.TPM

```
HV_scaled <- Hvit.TPM[, -1]
row.names(HV_scaled) <- Hvit.TPM[, 1]
Hvit.TPM.scaled <- HV_scaled * alpha
dim(HV_scaled)
```

```
## [1] 19126    36
```

```
head(Hvit.TPM[, 3])
```

```
## [1] 0.00 2.46 2.03 0.00 0.94 0.00
```

```
head(Hvit.TPM.scaled[, 2])
```

```
## [1] 0.0000000 2.3554423 1.9437186 0.0000000 0.9000471 0.0000000
```

Scaling Ofas.TPM

```
OF_scaled <- Ofas.TPM[, -1]
row.names(OF_scaled) <- Ofas.TPM[, 1]
Ofas.TPM.scaled <- OF_scaled * OFalpha
dim(OF_scaled)
```

```
## [1] 19811    11
```

```
head(Ofas.TPM[, 3])
```

```
## [1] 13.87  3.52 42.38  2.47 15.21  0.00
```

```
head(Ofas.TPM.scaled[, 2])
```

```
## [1] 13.756124  3.491100 42.032049  2.449721 15.085122  0.000000
```

Multiplying by α results in our Hvit TPM numbers being just a little smaller, though it will matter a lot for some of the outrageously large numbers:

```
which(Hvit.TPM[, 3] == max(Hvit.TPM[, 3]))
## [1] 8504
# 8504
Hvit.TPM[, 3][8504]
## [1] 136409.6

Hvit.TPM.scaled[, 2][8504]
## [1] 130611.7

which(Ofas.TPM[, 3] == max(Ofas.TPM[, 3]))
## [1] 15507

Ofas.TPM[, 3][15507]
## [1] 34293.29
```

```
Ofas.TPM.scaled[, 2][15507]
## [1] 34011.73
```

Finally, we want to normalize for size using DESeq2, and then write the resulting matrices to file for filtering to single copy orthologs and other downstream analysis.

```
write.table(Ofas.TPM.scaled, "C:/Users/cruth/Google Drive/Treehoppers/SRAProject/WGCNA/Ofasciatus/Ofas_
    sep = "\t")

write.table(Hvit.TPM.scaled, "C:/Users/cruth/Google Drive/Treehoppers/SRAProject/WGCNA/Hvitripennis/Hvi
    sep = "\t")

library(DESeq2)
colData <- read.table("C:/Users/cruth/Google Drive/Treehoppers/SRAProject/WGCNA/Hemiptera_SampleInforma
    header = TRUE)
hvCol <- (colData[c(42:62, 74:88), ])
hvTPM <- as.matrix(Hvit.TPM.scaled)
storage.mode(hvTPM) = "integer"
hv.dds <- DESeqDataSetFromMatrix(hvTPM, colData = na.omit(hvCol), design = ~Tissue +
    Pool)

hv.dds <- estimateSizeFactors(hv.dds)  ### These steps will take a little bit....
hv.dds <- estimateDispersions(hv.dds)
hvScaledNorm <- as.data.frame(assay(hv.dds), normalized = TRUE)  ### Normalizes TPM to account for libr
write.table(hvScaledNorm, "Hvit_Scaled_SizeNormed_Integer_TPM.matrix", sep = "\t")


ofTPM <- as.matrix(Ofas.TPM.scaled)
ofCol <- colData[c(63:73), ]
storage.mode(ofTPM) = "integer"
of.dds <- DESeqDataSetFromMatrix(ofTPM, colData = ofCol, design = ~Tissue + Pool)
of.dds <- estimateSizeFactors(of.dds)
of.dds <- estimateDispersions(of.dds)
ofScaledNorm <- as.data.frame(assay(of.dds), normalized = TRUE)
write.table(ofScaledNorm, "Ofas_Scaled_SizeNormed_Integer_TPM.matrix", sep = "\t")


ECid <- Ecar.TPM[, 1]
Ecar.TPM <- Ecar.TPM[, -1]
rownames(Ecar.TPM) <- ECid
ec.dds <- as.matrix(Ecar.TPM)
ecCol <- colData[1:41, ]
storage.mode(ec.dds) = "integer"

ec.dds <- DESeqDataSetFromMatrix(ec.dds, colData = ecCol, design = ~Tissue + Pool)

ec.dds <- estimateSizeFactors(ec.dds)
ec.dds <- estimateDispersions(ec.dds)
ecNorm <- as.data.frame(assay(ec.dds), normalized = TRUE)
write.table(ecNorm, "Ecar_SizeNormed_Integer_TPM.matrix", sep = "\t")
```

That's it. Now we have

1) Scaled the larger transcriptomes (Hvit, Ofas) to the smallest (Ecar)
2) Used DESeq2 to account for different sequencing depths within species

3) Wrote new tables of scaled and size-normalized TPM

The next step in my pipeline is filtering these matrices for single-copy orthologs.

R session information

```
sessionInfo()
```

```
## R version 3.6.2 (2019-12-12)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17134)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] dplyr_0.8.3
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.3       crayon_1.3.4     digest_0.6.23    assertthat_0.2.1
##  [5] R6_2.4.1         formatR_1.7      magrittr_1.5     evaluate_0.14
##  [9] pillar_1.4.3     rlang_0.4.2      stringi_1.4.3    rmarkdown_2.0
## [13] tools_3.6.2      stringr_1.4.0    glue_1.3.1       purrr_0.3.3
## [17] xfun_0.11        yaml_2.2.0       compiler_3.6.2   pkgconfig_2.0.3
## [21] htmltools_0.4.0  tidyselect_0.2.5 knitr_1.26       tibble_2.1.3
```