Homework: Analyzing Count Data

J. Jedediah Smith

2023-04-15

## Reviewing SummarizedExperiment

The SummarizedExperiment data structure is commonly used to provide data for a related series of high-throughput experiments.

One way to familiarize yourself with this data structure is to create a new one:

set.seed(123)  
# Create a toy SummarizedExperiment object  
counts <- matrix(rnbinom(150, mu=50, size=10), nrow=15, ncol=10)  
rownames(counts) <- paste0("gene", 1:15)  
colnames(counts) <- paste0("sample", 1:10)  
col\_data <- data.frame(condition=rep(c("A", "B"), each=5))  
row\_data <- data.frame(symbol=paste0("gene", 1:15))

Let’s take a look at these datastructures:

knitr::kable(counts)

|  | sample1 | sample2 | sample3 | sample4 | sample5 | sample6 | sample7 | sample8 | sample9 | sample10 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| gene1 | 46 | 48 | 57 | 53 | 74 | 42 | 37 | 42 | 19 | 60 |
| gene2 | 25 | 29 | 38 | 58 | 16 | 55 | 39 | 88 | 36 | 45 |
| gene3 | 81 | 36 | 42 | 44 | 45 | 64 | 41 | 73 | 33 | 64 |
| gene4 | 20 | 31 | 36 | 70 | 51 | 66 | 30 | 28 | 29 | 77 |
| gene5 | 71 | 34 | 60 | 56 | 37 | 88 | 45 | 44 | 65 | 49 |
| gene6 | 54 | 51 | 52 | 71 | 56 | 23 | 69 | 47 | 28 | 26 |
| gene7 | 47 | 48 | 47 | 51 | 98 | 99 | 57 | 44 | 29 | 29 |
| gene8 | 60 | 23 | 46 | 41 | 54 | 34 | 121 | 51 | 39 | 50 |
| gene9 | 29 | 33 | 65 | 75 | 67 | 40 | 37 | 36 | 33 | 55 |
| gene10 | 37 | 49 | 72 | 77 | 30 | 34 | 20 | 33 | 52 | 68 |
| gene11 | 64 | 74 | 43 | 48 | 79 | 51 | 23 | 78 | 21 | 60 |
| gene12 | 24 | 55 | 56 | 30 | 71 | 64 | 28 | 43 | 58 | 71 |
| gene13 | 40 | 54 | 39 | 50 | 51 | 67 | 74 | 58 | 67 | 53 |
| gene14 | 51 | 79 | 39 | 41 | 33 | 101 | 70 | 74 | 54 | 37 |
| gene15 | 62 | 30 | 40 | 19 | 34 | 43 | 51 | 40 | 22 | 30 |

col\_data

## condition  
## 1 A  
## 2 A  
## 3 A  
## 4 A  
## 5 A  
## 6 B  
## 7 B  
## 8 B  
## 9 B  
## 10 B

row\_data

## symbol  
## 1 gene1  
## 2 gene2  
## 3 gene3  
## 4 gene4  
## 5 gene5  
## 6 gene6  
## 7 gene7  
## 8 gene8  
## 9 gene9  
## 10 gene10  
## 11 gene11  
## 12 gene12  
## 13 gene13  
## 14 gene14  
## 15 gene15

We are now ready to create a SummarizedExperiment object like so:

library(SummarizedExperiment)  
se <- SummarizedExperiment(assays=list(counts=counts), colData=col\_data, rowData=row\_data)

## Retrieving Data

We can retrieve the count data from SummarizeExperiments with function assays:

knitr::kable(assays(se)[[1]])

|  | sample1 | sample2 | sample3 | sample4 | sample5 | sample6 | sample7 | sample8 | sample9 | sample10 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| gene1 | 46 | 48 | 57 | 53 | 74 | 42 | 37 | 42 | 19 | 60 |
| gene2 | 25 | 29 | 38 | 58 | 16 | 55 | 39 | 88 | 36 | 45 |
| gene3 | 81 | 36 | 42 | 44 | 45 | 64 | 41 | 73 | 33 | 64 |
| gene4 | 20 | 31 | 36 | 70 | 51 | 66 | 30 | 28 | 29 | 77 |
| gene5 | 71 | 34 | 60 | 56 | 37 | 88 | 45 | 44 | 65 | 49 |
| gene6 | 54 | 51 | 52 | 71 | 56 | 23 | 69 | 47 | 28 | 26 |
| gene7 | 47 | 48 | 47 | 51 | 98 | 99 | 57 | 44 | 29 | 29 |
| gene8 | 60 | 23 | 46 | 41 | 54 | 34 | 121 | 51 | 39 | 50 |
| gene9 | 29 | 33 | 65 | 75 | 67 | 40 | 37 | 36 | 33 | 55 |
| gene10 | 37 | 49 | 72 | 77 | 30 | 34 | 20 | 33 | 52 | 68 |
| gene11 | 64 | 74 | 43 | 48 | 79 | 51 | 23 | 78 | 21 | 60 |
| gene12 | 24 | 55 | 56 | 30 | 71 | 64 | 28 | 43 | 58 | 71 |
| gene13 | 40 | 54 | 39 | 50 | 51 | 67 | 74 | 58 | 67 | 53 |
| gene14 | 51 | 79 | 39 | 41 | 33 | 101 | 70 | 74 | 54 | 37 |
| gene15 | 62 | 30 | 40 | 19 | 34 | 43 | 51 | 40 | 22 | 30 |

Similarly, we can retrieve the column data and row-data with colData and rowData respectively:

colData(se)

## DataFrame with 10 rows and 1 column  
## condition  
## <character>  
## sample1 A  
## sample2 A  
## sample3 A  
## sample4 A  
## sample5 A  
## sample6 B  
## sample7 B  
## sample8 B  
## sample9 B  
## sample10 B

rowData(se)

## DataFrame with 15 rows and 1 column  
## symbol  
## <character>  
## gene1 gene1  
## gene2 gene2  
## gene3 gene3  
## gene4 gene4  
## gene5 gene5  
## ... ...  
## gene11 gene11  
## gene12 gene12  
## gene13 gene13  
## gene14 gene14  
## gene15 gene15

## Using the ‘Airway’ RNA-Seq Data

The airway package contains a dataset of gene expression values from a study of airway epithelial cells in response to glucocorticoid treatment. This dataset is based on RNA-Seq data and includes information on gene expression levels for thousands of genes across several experimental conditions.

# BiocManager::install('airway')  
library(airway)  
data(airway)  
class(airway)

## [1] "RangedSummarizedExperiment"  
## attr(,"package")  
## [1] "SummarizedExperiment"

## Inspecting Read Match Count Data

### Column Meta-Data

colData(airway)

## DataFrame with 8 rows and 9 columns  
## SampleName cell dex albut Run avgLength  
## <factor> <factor> <factor> <factor> <factor> <integer>  
## SRR1039508 GSM1275862 N61311 untrt untrt SRR1039508 126  
## SRR1039509 GSM1275863 N61311 trt untrt SRR1039509 126  
## SRR1039512 GSM1275866 N052611 untrt untrt SRR1039512 126  
## SRR1039513 GSM1275867 N052611 trt untrt SRR1039513 87  
## SRR1039516 GSM1275870 N080611 untrt untrt SRR1039516 120  
## SRR1039517 GSM1275871 N080611 trt untrt SRR1039517 126  
## SRR1039520 GSM1275874 N061011 untrt untrt SRR1039520 101  
## SRR1039521 GSM1275875 N061011 trt untrt SRR1039521 98  
## Experiment Sample BioSample  
## <factor> <factor> <factor>  
## SRR1039508 SRX384345 SRS508568 SAMN02422669  
## SRR1039509 SRX384346 SRS508567 SAMN02422675  
## SRR1039512 SRX384349 SRS508571 SAMN02422678  
## SRR1039513 SRX384350 SRS508572 SAMN02422670  
## SRR1039516 SRX384353 SRS508575 SAMN02422682  
## SRR1039517 SRX384354 SRS508576 SAMN02422673  
## SRR1039520 SRX384357 SRS508579 SAMN02422683  
## SRR1039521 SRX384358 SRS508580 SAMN02422677

### Count Matrix

counts <- assays(airway)$counts  
counts <- counts[1:30,] # no more than this many genes  
knitr::kable(counts)

|  | SRR1039508 | SRR1039509 | SRR1039512 | SRR1039513 | SRR1039516 | SRR1039517 | SRR1039520 | SRR1039521 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ENSG00000000003 | 679 | 448 | 873 | 408 | 1138 | 1047 | 770 | 572 |
| ENSG00000000005 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ENSG00000000419 | 467 | 515 | 621 | 365 | 587 | 799 | 417 | 508 |
| ENSG00000000457 | 260 | 211 | 263 | 164 | 245 | 331 | 233 | 229 |
| ENSG00000000460 | 60 | 55 | 40 | 35 | 78 | 63 | 76 | 60 |
| ENSG00000000938 | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 |
| ENSG00000000971 | 3251 | 3679 | 6177 | 4252 | 6721 | 11027 | 5176 | 7995 |
| ENSG00000001036 | 1433 | 1062 | 1733 | 881 | 1424 | 1439 | 1359 | 1109 |
| ENSG00000001084 | 519 | 380 | 595 | 493 | 820 | 714 | 696 | 704 |
| ENSG00000001167 | 394 | 236 | 464 | 175 | 658 | 584 | 360 | 269 |
| ENSG00000001460 | 172 | 168 | 264 | 118 | 241 | 210 | 155 | 177 |
| ENSG00000001461 | 2112 | 1867 | 5137 | 2657 | 2735 | 2751 | 2467 | 2905 |
| ENSG00000001497 | 524 | 488 | 638 | 357 | 676 | 806 | 493 | 475 |
| ENSG00000001561 | 71 | 51 | 211 | 156 | 23 | 38 | 134 | 172 |
| ENSG00000001617 | 555 | 394 | 905 | 415 | 727 | 697 | 618 | 599 |
| ENSG00000001626 | 10 | 2 | 9 | 2 | 10 | 6 | 5 | 5 |
| ENSG00000001629 | 1660 | 1251 | 2259 | 1079 | 2462 | 2514 | 1888 | 1660 |
| ENSG00000001630 | 59 | 54 | 66 | 23 | 84 | 87 | 31 | 59 |
| ENSG00000001631 | 729 | 692 | 943 | 475 | 1034 | 1163 | 731 | 744 |
| ENSG00000002016 | 201 | 161 | 256 | 99 | 268 | 257 | 160 | 137 |
| ENSG00000002079 | 3 | 0 | 3 | 1 | 4 | 0 | 0 | 1 |
| ENSG00000002330 | 206 | 174 | 184 | 111 | 194 | 260 | 156 | 177 |
| ENSG00000002549 | 1459 | 1294 | 1317 | 998 | 1451 | 1824 | 853 | 1031 |
| ENSG00000002586 | 7507 | 7203 | 9501 | 6214 | 10973 | 12863 | 6834 | 7225 |
| ENSG00000002587 | 2 | 0 | 1 | 0 | 0 | 2 | 0 | 0 |
| ENSG00000002726 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| ENSG00000002745 | 4 | 6 | 22 | 10 | 2 | 1 | 5 | 3 |
| ENSG00000002746 | 151 | 139 | 117 | 65 | 90 | 102 | 86 | 119 |
| ENSG00000002822 | 411 | 303 | 446 | 195 | 445 | 523 | 295 | 300 |
| ENSG00000002834 | 6314 | 6364 | 7831 | 5809 | 6677 | 11970 | 5766 | 7825 |

## Our DIY Approach

Before utilizing ready-made approaches for analyzing RNA-Seq data, we will use our own approach(es). This will help us understand what actually the challenge is when analyzing high-throughput experiment data.

Our approach will be to view RNA-Seq data as *count* data where we count for each gene or for each transcript how many reads we find in our sample that align with a reference sequence from a database corresponding to this transcript.

Counts of random events are often well-described by the Poisson distribution, so this is how we will start.

## ‘Melting’ The Matrix

Before fitting models to the count data, we have to convert it to a ‘melted form’ like so:

library(reshape2) # for 'melt'  
counts\_df <- as.data.frame(counts)  
countsm = melt(t(counts\_df)) # little trick to swap columns and rows  
colnames(countsm) <- c('Sample', 'Gene', 'Count')  
knitr::kable(head(countsm,n=10))

| Sample | Gene | Count |
| --- | --- | --- |
| SRR1039508 | ENSG00000000003 | 679 |
| SRR1039509 | ENSG00000000003 | 448 |
| SRR1039512 | ENSG00000000003 | 873 |
| SRR1039513 | ENSG00000000003 | 408 |
| SRR1039516 | ENSG00000000003 | 1138 |
| SRR1039517 | ENSG00000000003 | 1047 |
| SRR1039520 | ENSG00000000003 | 770 |
| SRR1039521 | ENSG00000000003 | 572 |
| SRR1039508 | ENSG00000000005 | 0 |
| SRR1039509 | ENSG00000000005 | 0 |

## Modeling RNA-Seq Count Data Assuming a Poisson Distribution

We can create a model of the count data using a generalized linear model (glm). This means that ‘under the hood’ R uses still code for the regular linear model (function lm), but the input data is transformed before starting a regular lineary regresion so that we can treat it as if we deal with a new way of fitting specialized data that is not linear. The code is showing below.

# Fit a Poisson GLM to the count data  
poisson\_model <- glm(Count ~ Sample+Gene, data = countsm, family = "poisson")  
summary(poisson\_model)

##   
## Call:  
## glm(formula = Count ~ Sample + Gene, family = "poisson", data = countsm)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -27.042 -1.872 -0.288 1.806 29.358   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 6.430177 0.014112 455.656 < 2e-16 \*\*\*  
## SampleSRR1039509 -0.071507 0.008426 -8.486 < 2e-16 \*\*\*  
## SampleSRR1039512 0.336003 0.007661 43.858 < 2e-16 \*\*\*  
## SampleSRR1039513 -0.133703 0.008565 -15.610 < 2e-16 \*\*\*  
## SampleSRR1039516 0.308449 0.007706 40.029 < 2e-16 \*\*\*  
## SampleSRR1039517 0.578129 0.007310 79.089 < 2e-16 \*\*\*  
## SampleSRR1039520 0.018686 0.008236 2.269 0.0233 \*   
## SampleSRR1039521 0.182447 0.007922 23.031 < 2e-16 \*\*\*  
## GeneENSG00000000005 -19.884725 163.702615 -0.121 0.9033   
## GeneENSG00000000419 -0.327148 0.020055 -16.313 < 2e-16 \*\*\*  
## GeneENSG00000000457 -1.120243 0.026173 -42.802 < 2e-16 \*\*\*  
## GeneENSG00000000460 -2.542293 0.048061 -52.898 < 2e-16 \*\*\*  
## GeneENSG00000000938 -7.590010 0.577496 -13.143 < 2e-16 \*\*\*  
## GeneENSG00000000971 2.096109 0.013755 152.387 < 2e-16 \*\*\*  
## GeneENSG00000001036 0.564778 0.016257 34.741 < 2e-16 \*\*\*  
## GeneENSG00000001084 -0.187355 0.019280 -9.718 < 2e-16 \*\*\*  
## GeneENSG00000001167 -0.636644 0.022067 -28.850 < 2e-16 \*\*\*  
## GeneENSG00000001460 -1.372074 0.028861 -47.541 < 2e-16 \*\*\*  
## GeneENSG00000001461 1.338454 0.014584 91.778 < 2e-16 \*\*\*  
## GeneENSG00000001497 -0.286391 0.019821 -14.449 < 2e-16 \*\*\*  
## GeneENSG00000001561 -1.936352 0.036561 -52.962 < 2e-16 \*\*\*  
## GeneENSG00000001617 -0.189593 0.019291 -9.828 < 2e-16 \*\*\*  
## GeneENSG00000001626 -4.796802 0.143446 -33.440 < 2e-16 \*\*\*  
## GeneENSG00000001629 0.911934 0.015368 59.339 < 2e-16 \*\*\*  
## GeneENSG00000001630 -2.550895 0.048253 -52.865 < 2e-16 \*\*\*  
## GeneENSG00000001631 0.092626 0.017947 5.161 2.45e-07 \*\*\*  
## GeneENSG00000002016 -1.349734 0.028605 -47.185 < 2e-16 \*\*\*  
## GeneENSG00000002079 -6.203716 0.288967 -21.469 < 2e-16 \*\*\*  
## GeneENSG00000002330 -1.401062 0.029197 -47.986 < 2e-16 \*\*\*  
## GeneENSG00000002549 0.544164 0.016318 33.348 < 2e-16 \*\*\*  
## GeneENSG00000002586 2.443336 0.013533 180.553 < 2e-16 \*\*\*  
## GeneENSG00000002587 -7.079184 0.447402 -15.823 < 2e-16 \*\*\*  
## GeneENSG00000002726 -8.688622 1.000084 -8.688 < 2e-16 \*\*\*  
## GeneENSG00000002745 -4.718330 0.137973 -34.198 < 2e-16 \*\*\*  
## GeneENSG00000002746 -1.921279 0.036321 -52.897 < 2e-16 \*\*\*  
## GeneENSG00000002822 -0.709969 0.022610 -31.401 < 2e-16 \*\*\*  
## GeneENSG00000002834 2.289117 0.013622 168.041 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for poisson family taken to be 1)  
##   
## Null deviance: 640474.6 on 239 degrees of freedom  
## Residual deviance: 7839.1 on 203 degrees of freedom  
## AIC: 9490.9  
##   
## Number of Fisher Scoring iterations: 11

Let’s take a look at the obtained summary of the model. Most prominent is the section called ‘Coefficients’. We will focus on their name (left-most column), the column ‘Estimate’ and the P-value (“Pr(>|z|)”).

For a regular linear regression, the column ‘Estimate’ corresponds to the slope of a line. Because we performed a specialized type of regression to fit counts of a Poisson distribution, the column ‘Estimate’ has to be interpret differently. It stands for the logarithm of the predicted outcome for a that variable to be increased by one unit. In our case this stands for the predicted count values of a certain gene divided by the predicted count values averages over all except that particular gene. For example, for gene ENSG00000000971, the summary shows a value for the Estimate of 2.096109. This means or In other words, the gene ENSG00000000971 has count values that are about 8 times higher compared to the count values of all the other genes. We should also look at the P-values and ensure that they indicate statistical significance.

## Exercise 1

1. Interpret the summary of the model fit. What is the maximum and minimum deviation between the ground-truth and prediction? Hint: Look at min, max of Deviance Residuals

# Grab our values  
dr\_min <- -27.042  
dr\_max <- 29.358  
  
# Calculate the difference. A larger difference may indicate poorer fit.  
dr\_diff <- dr\_max - dr\_min  
dr\_diff

## [1] 56.4

1. Consider gene ENSG00000000938. What is the raw Estimate value shown in the summary for this gene? Does this indicate that the count values for this genes are higher or lower compared to the counts of the other genes?

# The estimate is -7.590010. This means it has gene counts x7.5 times lower than all of the other compared genes.

1. Based on what you obtained in b), what is the odds ratio in terms of gene counts for this gene divided by the average counts of all the other genes?

# Grab our count for the gene  
library(dplyr)  
count938 <- sum(countsm[which(countsm[,2]=="ENSG00000000938"),3])  
  
# Grab our count of the other genes  
group\_countsm <- countsm[,2:3] %>%  
 filter(Gene != "ENSG00000000938") %>%  
 group\_by(Gene) %>%  
 summarise(Freq = sum(Count))  
  
# Calculate the odds ratio  
count938 / mean(group\_countsm$Freq)

## [1] 0.0003112556

1. Apply the function tidy from R package broom to the Poisson model. Use function knitr::kable to print the resulting data frame.

library(broom)  
knitr::kable(tidy(poisson\_model))

| term | estimate | std.error | statistic | p.value |
| --- | --- | --- | --- | --- |
| (Intercept) | 6.4301766 | 0.0141119 | 455.6563343 | 0.0000000 |
| SampleSRR1039509 | -0.0715071 | 0.0084262 | -8.4863308 | 0.0000000 |
| SampleSRR1039512 | 0.3360027 | 0.0076612 | 43.8577867 | 0.0000000 |
| SampleSRR1039513 | -0.1337026 | 0.0085650 | -15.6103047 | 0.0000000 |
| SampleSRR1039516 | 0.3084488 | 0.0077057 | 40.0288608 | 0.0000000 |
| SampleSRR1039517 | 0.5781288 | 0.0073098 | 79.0894552 | 0.0000000 |
| SampleSRR1039520 | 0.0186858 | 0.0082358 | 2.2688404 | 0.0232780 |
| SampleSRR1039521 | 0.1824471 | 0.0079217 | 23.0312033 | 0.0000000 |
| GeneENSG00000000005 | -19.8847252 | 163.7026146 | -0.1214686 | 0.9033199 |
| GeneENSG00000000419 | -0.3271477 | 0.0200547 | -16.3127596 | 0.0000000 |
| GeneENSG00000000457 | -1.1202430 | 0.0261729 | -42.8016327 | 0.0000000 |
| GeneENSG00000000460 | -2.5422930 | 0.0480606 | -52.8976742 | 0.0000000 |
| GeneENSG00000000938 | -7.5900100 | 0.5774962 | -13.1429617 | 0.0000000 |
| GeneENSG00000000971 | 2.0961089 | 0.0137552 | 152.3867221 | 0.0000000 |
| GeneENSG00000001036 | 0.5647776 | 0.0162566 | 34.7414056 | 0.0000000 |
| GeneENSG00000001084 | -0.1873553 | 0.0192796 | -9.7178017 | 0.0000000 |
| GeneENSG00000001167 | -0.6366442 | 0.0220672 | -28.8501898 | 0.0000000 |
| GeneENSG00000001460 | -1.3720741 | 0.0288608 | -47.5411540 | 0.0000000 |
| GeneENSG00000001461 | 1.3384536 | 0.0145835 | 91.7784766 | 0.0000000 |
| GeneENSG00000001497 | -0.2863911 | 0.0198206 | -14.4491293 | 0.0000000 |
| GeneENSG00000001561 | -1.9363519 | 0.0365611 | -52.9620359 | 0.0000000 |
| GeneENSG00000001617 | -0.1895931 | 0.0192914 | -9.8278569 | 0.0000000 |
| GeneENSG00000001626 | -4.7968020 | 0.1434457 | -33.4398563 | 0.0000000 |
| GeneENSG00000001629 | 0.9119342 | 0.0153682 | 59.3388490 | 0.0000000 |
| GeneENSG00000001630 | -2.5508953 | 0.0482527 | -52.8653808 | 0.0000000 |
| GeneENSG00000001631 | 0.0926260 | 0.0179465 | 5.1612197 | 0.0000002 |
| GeneENSG00000002016 | -1.3497342 | 0.0286053 | -47.1847185 | 0.0000000 |
| GeneENSG00000002079 | -6.2037157 | 0.2889668 | -21.4686087 | 0.0000000 |
| GeneENSG00000002330 | -1.4010617 | 0.0291974 | -47.9858829 | 0.0000000 |
| GeneENSG00000002549 | 0.5441643 | 0.0163179 | 33.3477815 | 0.0000000 |
| GeneENSG00000002586 | 2.4433355 | 0.0135325 | 180.5529332 | 0.0000000 |
| GeneENSG00000002587 | -7.0791844 | 0.4474019 | -15.8228739 | 0.0000000 |
| GeneENSG00000002726 | -8.6886223 | 1.0000842 | -8.6878904 | 0.0000000 |
| GeneENSG00000002745 | -4.7183304 | 0.1379725 | -34.1976095 | 0.0000000 |
| GeneENSG00000002746 | -1.9212792 | 0.0363213 | -52.8966994 | 0.0000000 |
| GeneENSG00000002822 | -0.7099686 | 0.0226096 | -31.4012475 | 0.0000000 |
| GeneENSG00000002834 | 2.2891165 | 0.0136224 | 168.0406718 | 0.0000000 |

## Modeling RNA-Seq Count Data Assuming a Negative Bionomial Distribution

Next, we will use a similar approach for fitting a model that assumes a negative binomial distribution of the count data. One can view the Poisson distribution as a special case of the negative binomial distribution where the variance is given through the mean values. In contrast, the negative binomial distribution does not have this restriction, it can allow for larger variances as they are common in biological data.

The function for fitting a generalized linear model based on this distribution can be found in the MASS package (the name of the library is based on the book title of the classic “Modern Applied Statistics with S”).

library(MASS) # for glm.nb  
# Fit a negative binomial GLM to the count data  
nb\_model <- glm.nb(Count ~ Gene + Sample, data = countsm)  
summary(nb\_model)

##   
## Call:  
## glm.nb(formula = Count ~ Gene + Sample, data = countsm, init.theta = 22.15666089,   
## link = log)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -5.5437 -0.6492 -0.0673 0.5018 3.3931   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 6.543e+00 8.736e-02 74.905 < 2e-16 \*\*\*  
## GeneENSG00000000005 -3.686e+01 8.030e+05 0.000 0.99996   
## GeneENSG00000000419 -3.052e-01 1.082e-01 -2.821 0.00479 \*\*   
## GeneENSG00000000457 -1.091e+00 1.095e-01 -9.964 < 2e-16 \*\*\*  
## GeneENSG00000000460 -2.503e+00 1.168e-01 -21.440 < 2e-16 \*\*\*  
## GeneENSG00000000938 -7.572e+00 5.883e-01 -12.872 < 2e-16 \*\*\*  
## GeneENSG00000000971 2.104e+00 1.072e-01 19.629 < 2e-16 \*\*\*  
## GeneENSG00000001036 5.929e-01 1.075e-01 5.514 3.50e-08 \*\*\*  
## GeneENSG00000001084 -1.539e-01 1.080e-01 -1.424 0.15432   
## GeneENSG00000001167 -6.484e-01 1.087e-01 -5.968 2.40e-09 \*\*\*  
## GeneENSG00000001460 -1.353e+00 1.103e-01 -12.273 < 2e-16 \*\*\*  
## GeneENSG00000001461 1.368e+00 1.073e-01 12.750 < 2e-16 \*\*\*  
## GeneENSG00000001497 -2.686e-01 1.082e-01 -2.483 0.01302 \*   
## GeneENSG00000001561 -1.860e+00 1.123e-01 -16.567 < 2e-16 \*\*\*  
## GeneENSG00000001617 -1.736e-01 1.081e-01 -1.606 0.10820   
## GeneENSG00000001626 -4.783e+00 1.795e-01 -26.651 < 2e-16 \*\*\*  
## GeneENSG00000001629 9.204e-01 1.074e-01 8.570 < 2e-16 \*\*\*  
## GeneENSG00000001630 -2.547e+00 1.172e-01 -21.739 < 2e-16 \*\*\*  
## GeneENSG00000001631 1.058e-01 1.078e-01 0.981 0.32639   
## GeneENSG00000002016 -1.345e+00 1.102e-01 -12.199 < 2e-16 \*\*\*  
## GeneENSG00000002079 -6.185e+00 3.086e-01 -20.043 < 2e-16 \*\*\*  
## GeneENSG00000002330 -1.375e+00 1.103e-01 -12.464 < 2e-16 \*\*\*  
## GeneENSG00000002549 5.767e-01 1.075e-01 5.363 8.18e-08 \*\*\*  
## GeneENSG00000002586 2.461e+00 1.071e-01 22.974 < 2e-16 \*\*\*  
## GeneENSG00000002587 -7.061e+00 4.607e-01 -15.325 < 2e-16 \*\*\*  
## GeneENSG00000002726 -8.669e+00 1.006e+00 -8.613 < 2e-16 \*\*\*  
## GeneENSG00000002745 -4.691e+00 1.743e-01 -26.905 < 2e-16 \*\*\*  
## GeneENSG00000002746 -1.873e+00 1.124e-01 -16.667 < 2e-16 \*\*\*  
## GeneENSG00000002822 -7.010e-01 1.087e-01 -6.447 1.14e-10 \*\*\*  
## GeneENSG00000002834 2.317e+00 1.071e-01 21.627 < 2e-16 \*\*\*  
## SampleSRR1039509 -1.807e-01 6.459e-02 -2.797 0.00515 \*\*   
## SampleSRR1039512 2.659e-01 6.398e-02 4.155 3.25e-05 \*\*\*  
## SampleSRR1039513 -3.396e-01 6.485e-02 -5.237 1.63e-07 \*\*\*  
## SampleSRR1039516 2.067e-01 6.406e-02 3.227 0.00125 \*\*   
## SampleSRR1039517 3.000e-01 6.394e-02 4.692 2.70e-06 \*\*\*  
## SampleSRR1039520 -6.234e-02 6.442e-02 -0.968 0.33311   
## SampleSRR1039521 -5.048e-03 6.433e-02 -0.078 0.93746   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for Negative Binomial(22.1567) family taken to be 1)  
##   
## Null deviance: 17737.71 on 239 degrees of freedom  
## Residual deviance: 261.36 on 203 degrees of freedom  
## AIC: 2524.8  
##   
## Number of Fisher Scoring iterations: 1  
##   
##   
## Theta: 22.16   
## Std. Err.: 2.70   
##   
## 2 x log-likelihood: -2448.785

## Exercise 2: Comparing Residuals of the Models

Let us inspect the quality of the generated models. One way is to look at the difference between the ground-truth values and the predicted values. This is called ‘residual’ in statisics.

The formulate is: residual = observed y - predicted y

1. Use function residuals to compute how much our predictions deviate from the ground-truth values of the counts for the Poisson model. Summarize the numeric results (be creative).

# Save our residuals separately  
pmr <- data.frame(residuals(poisson\_model))  
  
# Count which bin they fall under  
pmr\_counts <- c(  
 sum(pmr[,1] >= 15),  
 sum(pmr[,1] >= 10 & pmr[,1] < 15),  
 sum(pmr[,1] >= 5 & pmr[,1] < 10),  
 sum(pmr[,1] > 0 & pmr[,1] < 5),  
 sum(pmr[,1] == 0),  
 sum(pmr[,1] < 0 & pmr[,1] > -5),  
 sum(pmr[,1] <= -5 & pmr[,1] > -10),  
 sum(pmr[,1] <= -10 & pmr[,1] > -15),  
 sum(pmr[,1] <= -15)  
)  
  
# Label the bins  
bins <- c("15 or More", "14 to 10", "9 to 5","4 to 0","0","-0 to -4", "-5 to -9", "-10 to -14", "-15 or Less")  
  
# Build a table illustrating the distribution of residuals, the closer the spread is to 0 the better.  
knitr::kable(data.frame(bins, pmr\_counts))

| bins | pmr\_counts |
| --- | --- |
| 15 or More | 3 |
| 14 to 10 | 3 |
| 9 to 5 | 22 |
| 4 to 0 | 77 |
| 0 | 0 |
| -0 to -4 | 106 |
| -5 to -9 | 20 |
| -10 to -14 | 5 |
| -15 or Less | 4 |

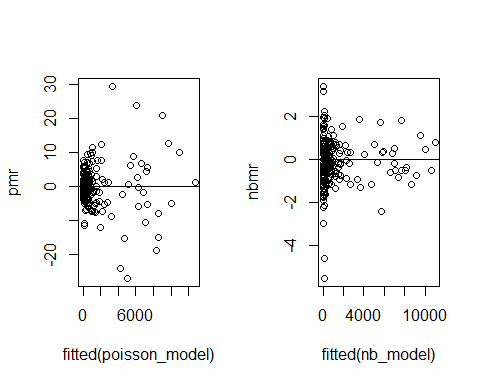
1. Apply the same approach to the residuals corresponding to the negative binomial model.

# Save our residuals separately  
nbmr <- data.frame(residuals(nb\_model))  
  
# Count which bin they fall under  
nbmr\_counts <- c(  
 sum(nbmr[,1] >= 15),  
 sum(nbmr[,1] >= 10 & nbmr[,1] < 15),  
 sum(nbmr[,1] >= 5 & nbmr[,1] < 10),  
 sum(nbmr[,1] > 0 & nbmr[,1] < 5),  
 sum(nbmr[,1] == 0),  
 sum(nbmr[,1] < 0 & nbmr[,1] > -5),  
 sum(nbmr[,1] <= -5 & nbmr[,1] > -10),  
 sum(nbmr[,1] <= -10 & nbmr[,1] > -15),  
 sum(nbmr[,1] <= -15)  
)  
  
# Label the bins  
bins <- c("15 or More", "14 to 10", "9 to 5","4 to 0","0","-0 to -4", "-5 to -9", "-10 to -14", "-15 or Less")  
  
# Build a table illustrating the distribution of residuals, the closer the spread is to 0 the better.  
knitr::kable(data.frame(bins, nbmr\_counts))

| bins | nbmr\_counts |
| --- | --- |
| 15 or More | 0 |
| 14 to 10 | 0 |
| 9 to 5 | 0 |
| 4 to 0 | 102 |
| 0 | 0 |
| -0 to -4 | 137 |
| -5 to -9 | 1 |
| -10 to -14 | 0 |
| -15 or Less | 0 |

1. Find a way to plot your findings from a) and b) in an informative way. Be creative: you can create a combined plot or two separate plots; feel free to use plotting functions from either base-R or other libraries such as ggplot.

# Grab our residuals  
pmr <- residuals(poisson\_model)  
nbmr <- residuals(nb\_model)  
  
# Set plot frame  
par(mfrow=c(1,2))   
  
# Graph residual plots. Will visualize the distribution of residuals. The better the model is, the closer the spread should be to 0.  
plot(fitted(poisson\_model), pmr)  
abline(0,0)  
  
plot(fitted(nb\_model), nbmr)  
abline(0,0)



# Reset plot frame  
par(mfrow=c(1,1))

1. Based on your findings in a)-c), what model seems superior and why?

# The NB Model is superior. Its residuals, which represent deviation between prediction and ground-truth, are less broadly distributed and on average closer to the 0.

## Choosing the Better Model using the Aikike Information Criterion

In statistic and machine learning we have the challenge that models with more parameters are intrinsically better able to obtain better fits to data compared to models with less parameters. But ‘more’ is not necessarily ‘better’ as it can lead to over-fitting! Somehow we need a way to balance the need for fitting the data well with the need to avoid a proliferation in model complexity.

The Akaike Information Criterion (AIC) is a measure of the quality of a statistical model that takes into account both how well the model fits the data and how complex the model is. The AIC score is calculated by taking the log-likelihood of the model and subtracting a penalty term that increases as the number of parameters in the model increases.

The AIC helps us choose the best model by balancing two things: how well the model fits the data and how complicated the model is. The AIC score is lower for models that fit the data well while using fewer parameters, which means they are more likely to generalize well to new data. So, the model with the lowest AIC score is usually considered the best model for a given set of data.

In R we have the function AIC that computes this metric for a given model (for example ACI(model) )

## Exercise 3

1. Use the function AIC() to compute the AIC metric for the model corresponding to the Poisson distribution. What is the value?

# Calculate AIC for Poisson Model. Value is 9490.93  
AIC(poisson\_model)

## [1] 9490.93

1. Use the function AIC() to compute the AIC metric for the model corresponding to the negative binomial distribution. What is the value?

# Calculate AIC for NB Model. Value is 2524.785  
AIC(nb\_model)

## [1] 2524.785

1. Based on your results in a) and b), which model seems to be preferable?

# The NB Model remains superior, owing to its lower AIC value. It uses less parameters and is more likely to adapt well to general data.

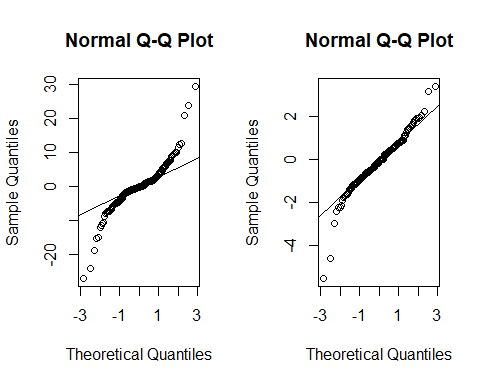
## Exercise 4: General understanding of Count Matrix Analysis

1. Use common sense: Why is the number RNA-Seq reads one identifies for a certain transcript variant approximately proportional to the RNA sequence length of that transcript variant?

# A longer variant will have more reads that map to it.

1. It is tempting to use the odds-ratios of counts we obtained earlier in form of the glm estimates as measures for gene expression. Why is that not directly workable? Hint: see a)

# A straight odds-ratio will be bias and misleading. Longer variants will have more reads that map to them, and therefore appear to be more expressed. We need to normalize for read length to draw any meaningful conclusions. You can see this in our data. The QQ plots are non-linear, indicating that they need to be normalized.  
  
par(mfrow=c(1,2))   
qqnorm(pmr)  
qqline(pmr)  
  
qqnorm(nbmr)  
qqline(nbmr)



par(mfrow=c(1,1))

1. Another key aspect that our initial modeling using glm did not account for is that experiments are typically grouped together, and one compares one condition versus another, each supported by counts for several samples. Present ideas how our models could be refined such that we do not simply compare each sample individually but instead of sample groups by different treatments.

# We can add variables in our regression models. If possible, one solution would be to have experimental conditions included with the data, then list it as a variable to account for in the model. It might also be possible to do something with clustering, but this seems less reliable than utilizing the original data.

## Resources

* <https://stats.oarc.ucla.edu/r/dae/poisson-regression/>
* <https://bioconductor.org/packages/release/data/experiment/html/airway.html>