

RNA-Seq Data Analysis 1

Lauren Vanderlinden

BIOS 6660

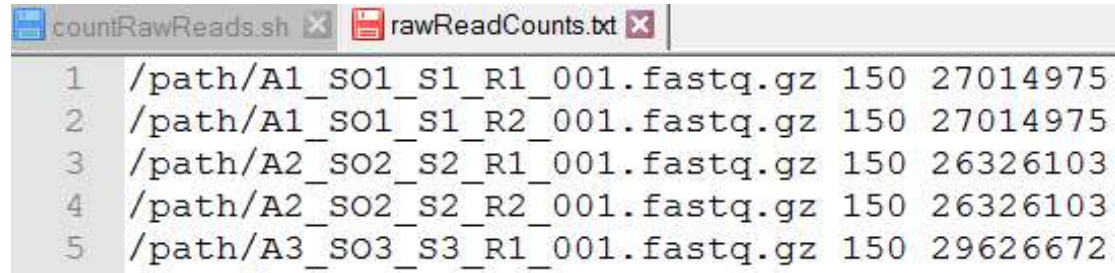
Spring 2019

BASH Scripts from Last Time

> cd /home/teststudent/hw6Test/code

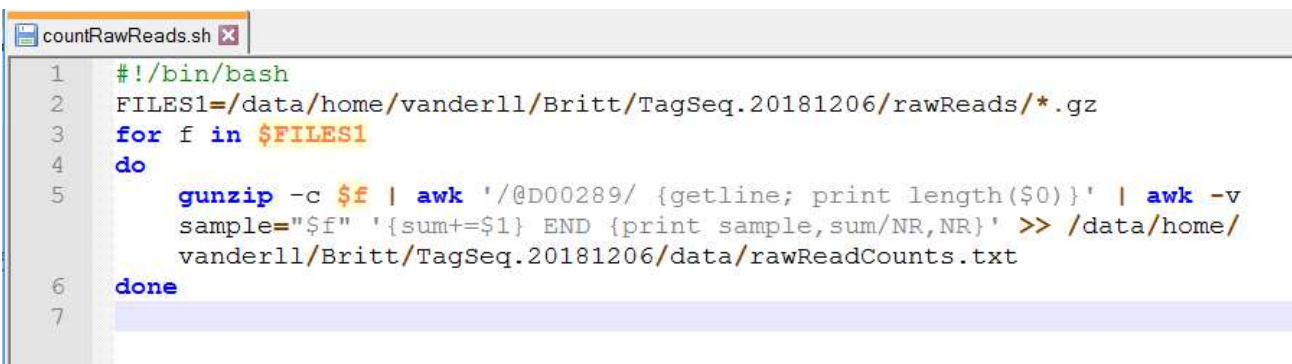
> chmod -R u+wrx *

> ./countRawReads.sh



1	/path/A1_S01_S1_R1_001.fastq.gz	150	27014975
2	/path/A1_S01_S1_R2_001.fastq.gz	150	27014975
3	/path/A2_S02_S2_R1_001.fastq.gz	150	26326103
4	/path/A2_S02_S2_R2_001.fastq.gz	150	26326103
5	/path/A3_S03_S3_R1_001.fastq.gz	150	29626672

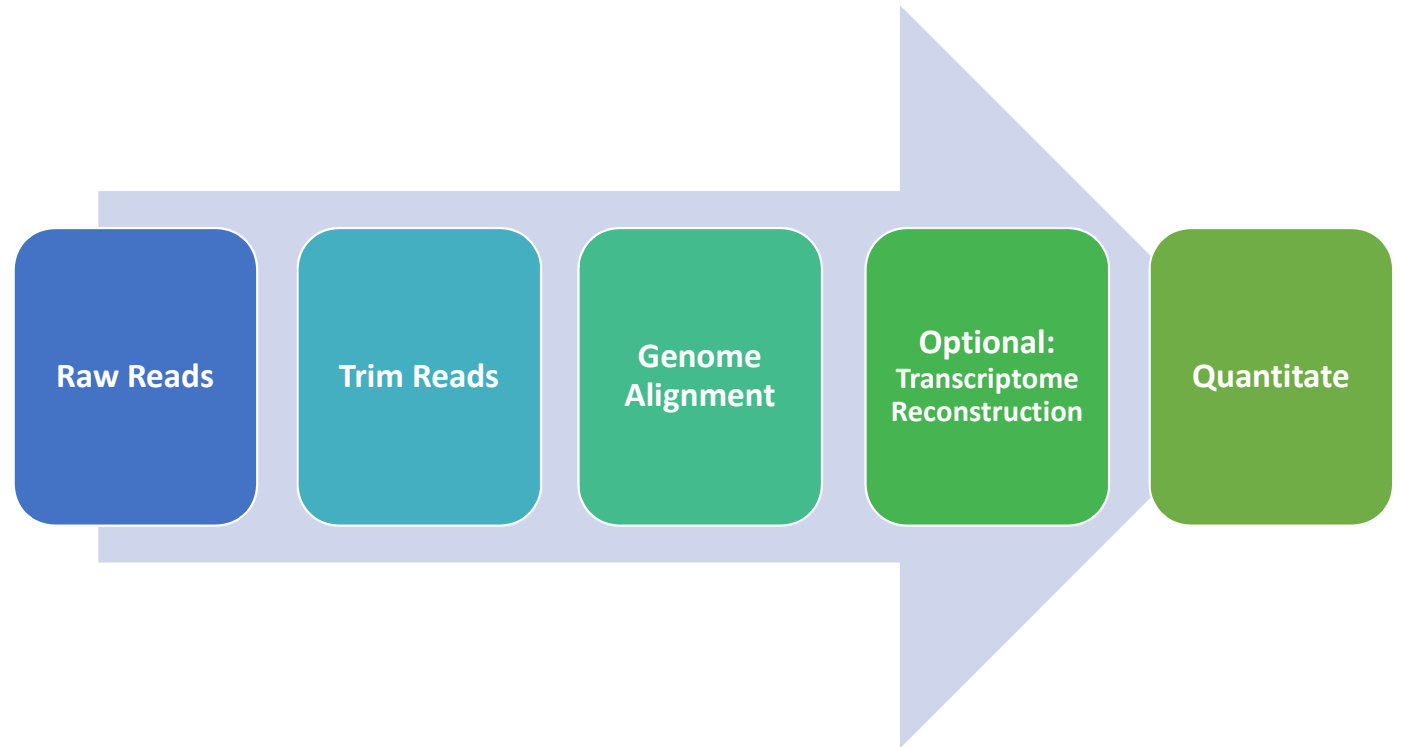
Yampa: /BIOS6660/Homework6/exampleProgramsAndReports



```
1 #!/bin/bash
2 FILES1=/data/home/vanderll/Britt/TagSeq.20181206/rawReads/*.gz
3 for f in $FILES1
4 do
5     gunzip -c $f | awk '@D00289/ {getline; print length($0)}' | awk -v
6     sample="$f" '{sum+= $1} END {print sample,sum/NR,NR}' >> /data/home/
7     vanderll/Britt/TagSeq.20181206/data/rawReadCounts.txt
8 done
```

Overview From Last Time

- We went from raw RNA-Seq reads to a nice matrix of counts
 - Rows = genes
 - Columns = samples
- Now all the work we will do is in R!



What Type of Data Do We Have?

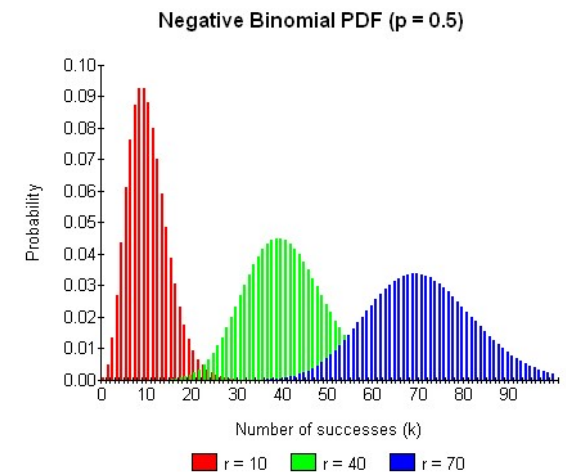
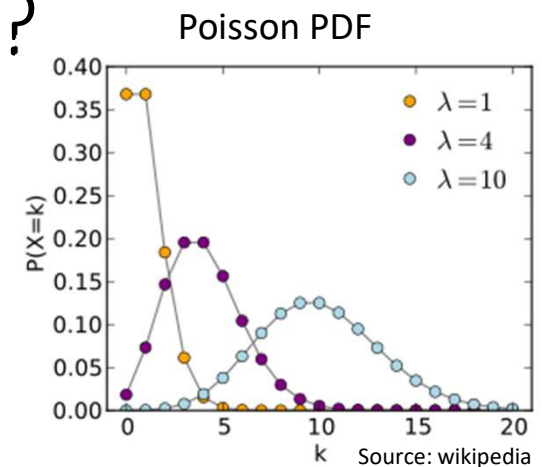
- We have a matrix of **counts**

1. Poisson

- Assuming mean = variance
- 1 parameter (λ) (Figure k = # occurrences)

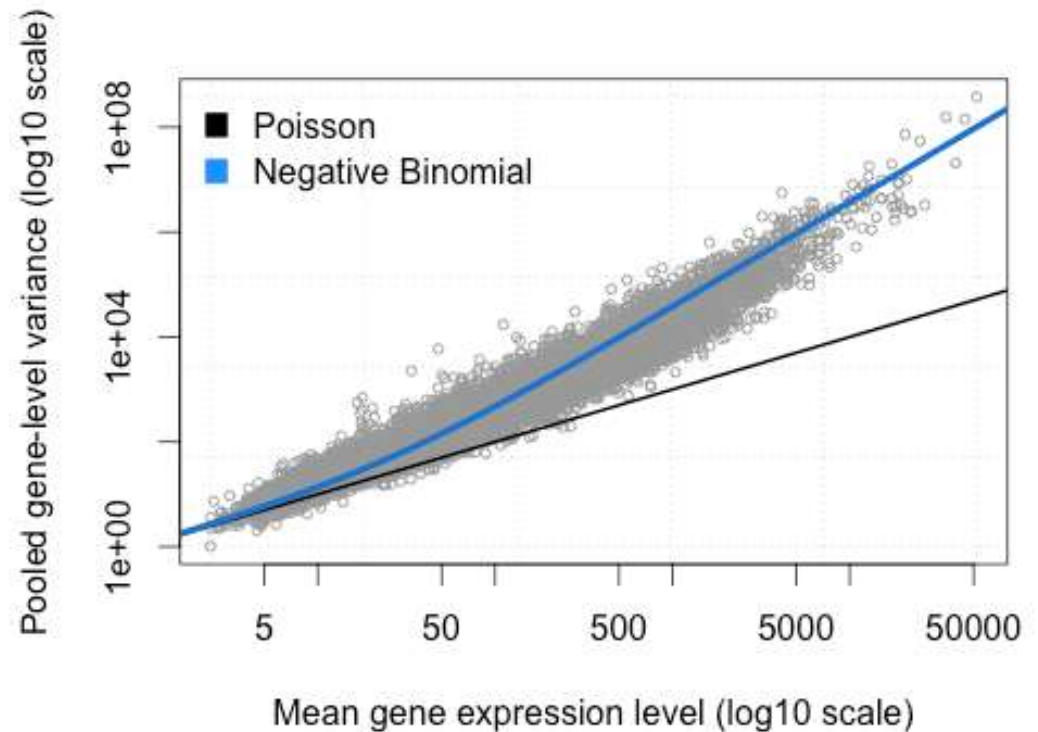
2. Negative Binomial

- Does not make mean = variance assumption
- 2 parameters:
 - r is the total # successes
 - p is the probability of success of a single trial



Negative Binomial

- Variance of counts is generally greater than their mean, especially for genes expressed at a higher level.
- This is called **overdispersion**
- $1/r$ is also sometimes called the “dispersion” or “shape” parameter
- Negative Binomial current choice



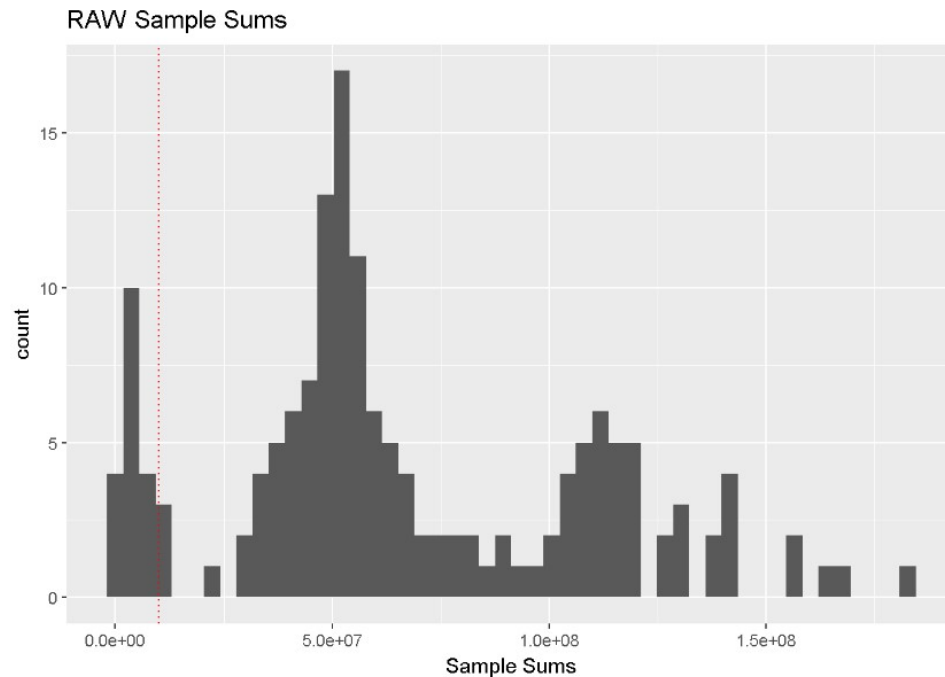
Source: RECount Project

Exploring Data Through Visualization

- Normally, plot your data in a histogram and see what it looks like.
- This is not so easy with any 'Omics data as you have large number of features (~20,000 genes)
- Next few slides I'm going to walk through some visualizations I look at when analyzing a dataset

Visualizing Data – Library Sizes

```
ggplot(countSums, geom="histogram", main = "RAW Sample Sums", xlab = "Sample Sums", bins=50) + geom_vline(xintercept=10000000, col="red", linetype="dotted")
```

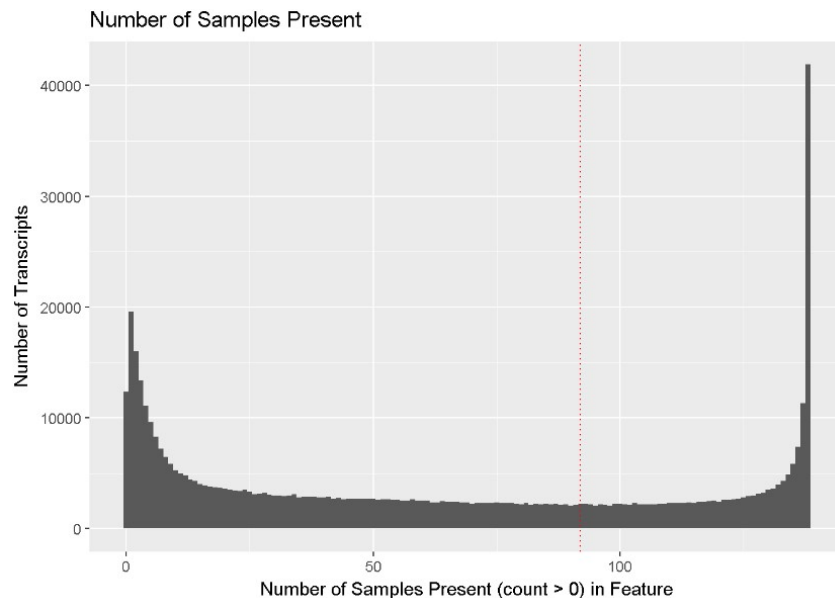


- ggplot2 is a great tool
- Sample-level QC
- In this example I had 158 samples, so it was difficult to look at a table of library sizes

Visualizing Data – Number of 0 Counts

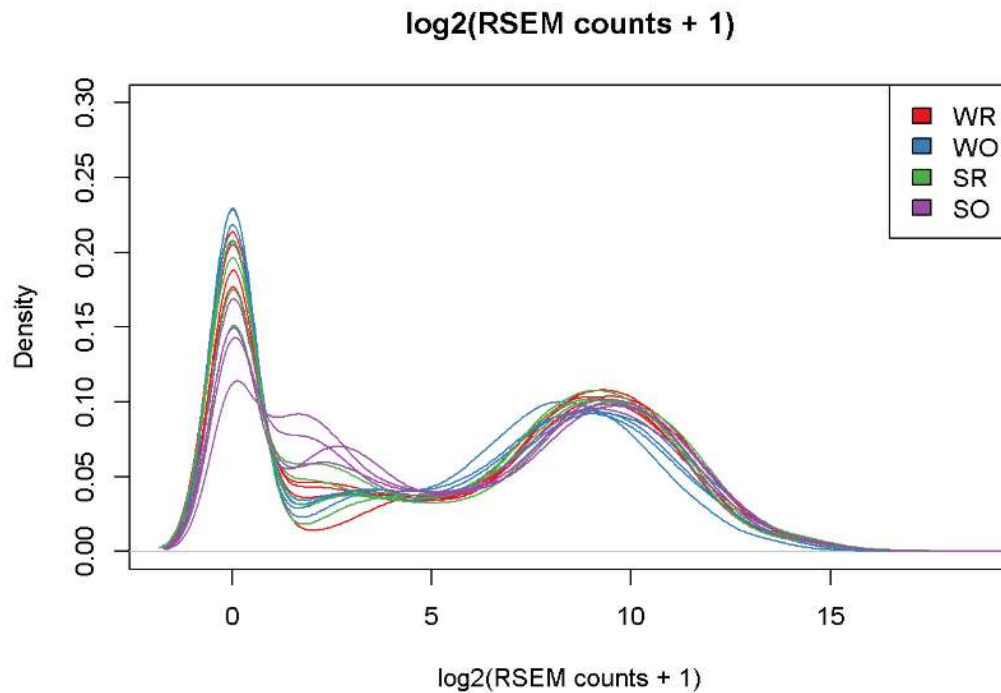
```
presentCount = apply(counts3, 1, function(a) sum(a>0))

qplot(presentCount, geom="histogram", main = "Number of Samples Present", xlab = "Number of Samples Present (count > 0) in Feature", binwidth = 1) + geom_vline(xintercept=(2/3)*ncol(counts3), col="red", linetype="dotted") + ylab("Number of Transcripts")
```



- Feature-level QC
- At the VERY least, you want to remove genes (isoforms) from your dataset that have 0 counts across ALL samples
- Detection Above BackGround (DABG)
- Debated topic

Visualizing Data- Density Plots Example 1



- Density plot for each individual sample
- Colored by group
- Big Spike around 0
 - DABG 0 counts in all samples
- Larger hump later on

Visualizing Data- Density Plots Example 1

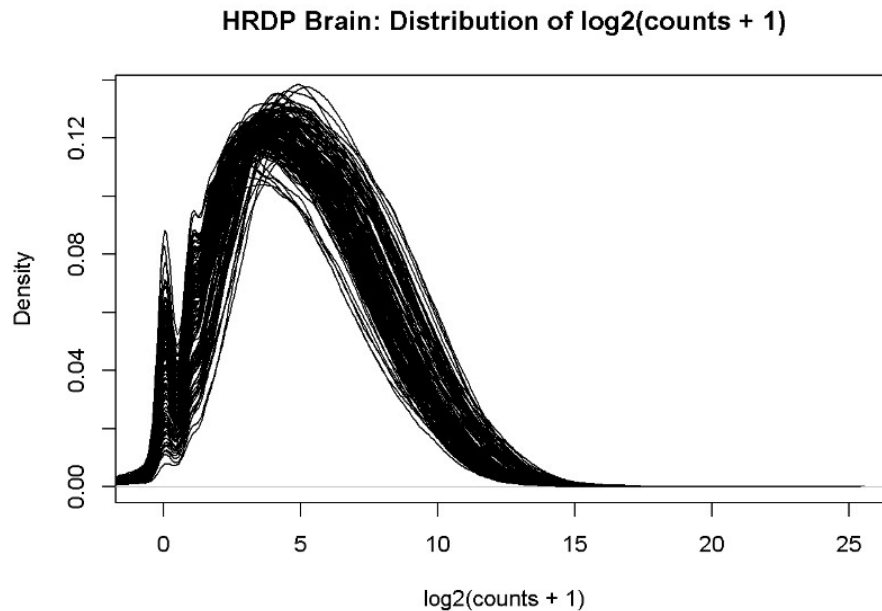
Code

```
library(RColorBrewer)
```

```
colorsWant = brewer.pal(9, "Set1")
#Visualizing distributions of samples
par(mfrow=c(1,1))
plot(density(log2(counts2[,1]+1)), main="log2(RSEM counts + 1)", xlab="log2(RSEM counts + 1)", ylim=c(0,0.3), col=colorsWant[1])
for(i in 2:4) {lines (density(log2(counts2[,i]+1)), col=colorsWant[1])}
for(i in 5:8) {lines (density(log2(counts2[,i]+1)), col=colorsWant[2])}
for(i in 9:12) {lines (density(log2(counts2[,i]+1)), col=colorsWant[3])}
for(i in 13:16) {lines (density(log2(counts2[,i]+1)), col=colorsWant[4])}
legend("topright", c("WR", "WO", "SR", "SO"), fill=colorsWant)
```

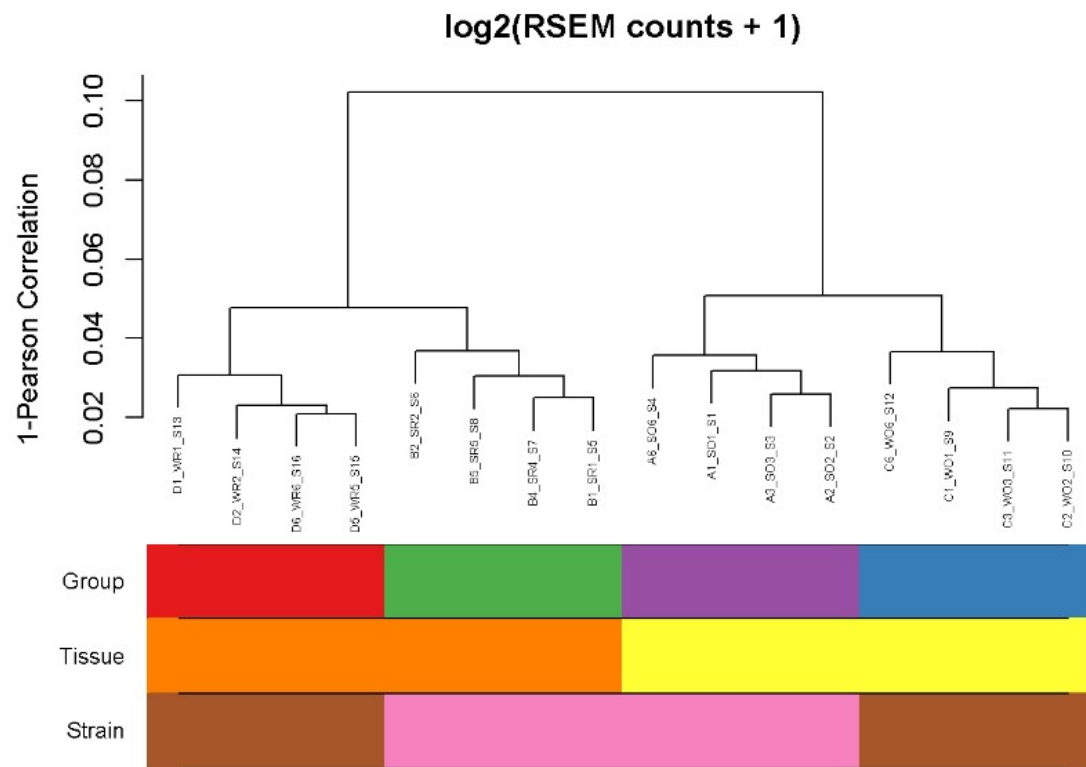
Visualizing Data- Density Plots Example 2

```
plot(density(log2(counts5[,1]+1)), main="HRDP Brain: Distribution of log2(counts + 1)", xlab="log2(counts + 1)")  
for(i in 2:ncol(counts5)) {lines (density(log2(counts5[,i])))}
```



- Still have a somewhat same shape
- Harsher feature selection on DABG calls
- Smaller spike near 0
- Larger hump later on

Visualizing Data – Dendrogram Example 1



- y-axis is the distance (or dissimilarity)
 - 1-correlation
- Clustering by group really well
 - This is an animal model
- Tissue explains the main distance
 - First break at 0.1
- Strain explains the second most distance
 - Second breaks around 0.05
- Great tool for experiments with small number samples

Visualizing Data- Dendrogram Example 1

Code

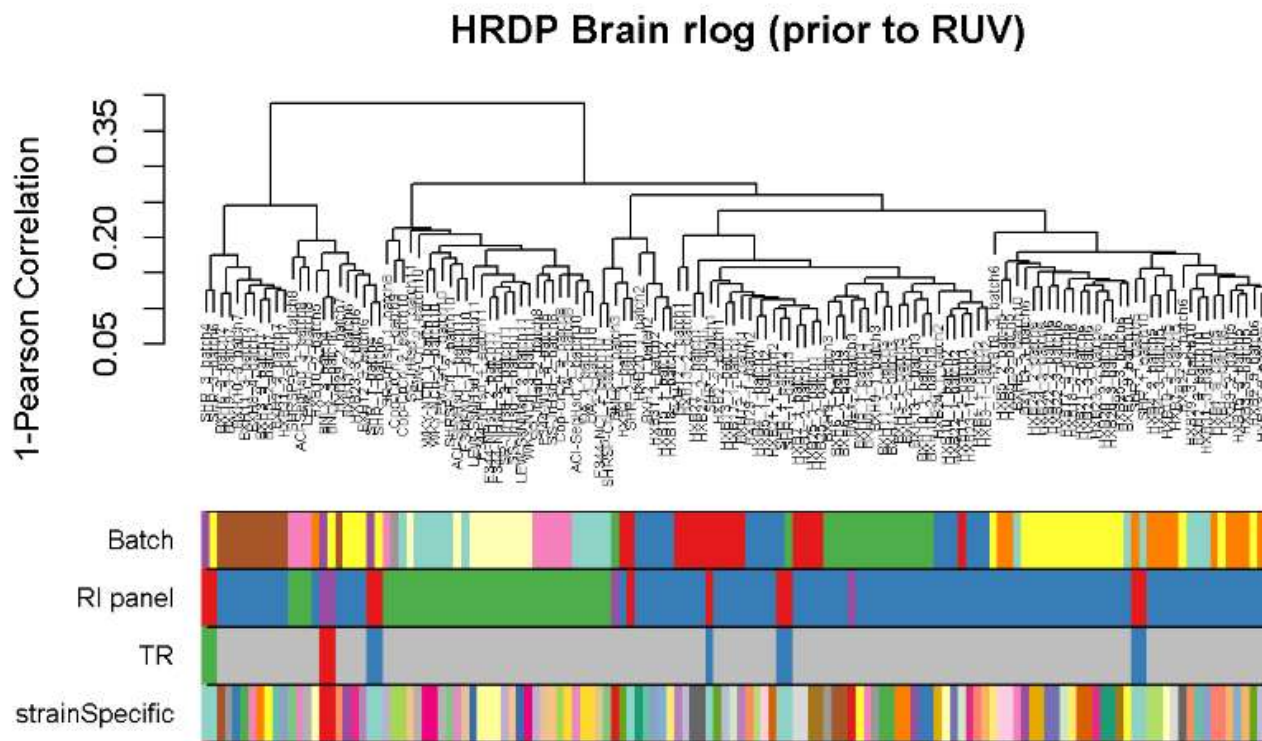
library(WGCNA)

```
counts.hclust = hclust(as.dist(1-cor(log2(counts2+1), method="pearson")))
group.hclust = substr(sapply(strsplit(counts.hclust$labels, split="_", fixed=TRUE), "[", 2), 1, 2)

colors4plot=c()
for(i in 1:length(group.hclust)){
  if(group.hclust[i]=="WR"){colors4plot[i]=colorsWant[1]}
  if(group.hclust[i]=="WO"){colors4plot[i]=colorsWant[2]}
  if(group.hclust[i]=="SR"){colors4plot[i]=colorsWant[3]}
  if(group.hclust[i]=="SO"){colors4plot[i]=colorsWant[4]}
}
colors4plot=as.matrix(colors4plot)
colnames(colors4plot) = "Group"
...
par(mfrow=c(1,1))
plotDendroAndColors(dendro=counts.hclust, colors=cbind(colors4plot, tissues4plot, strain4plot), main=
"log2(RSEM counts + 1)", ylab="1-Pearson Correlation",cex.dendroLabels = 0.45)
```

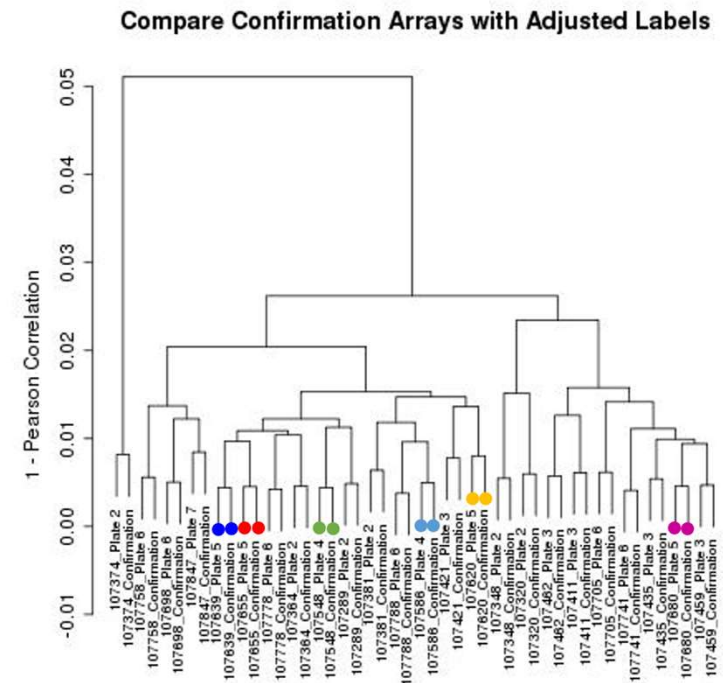
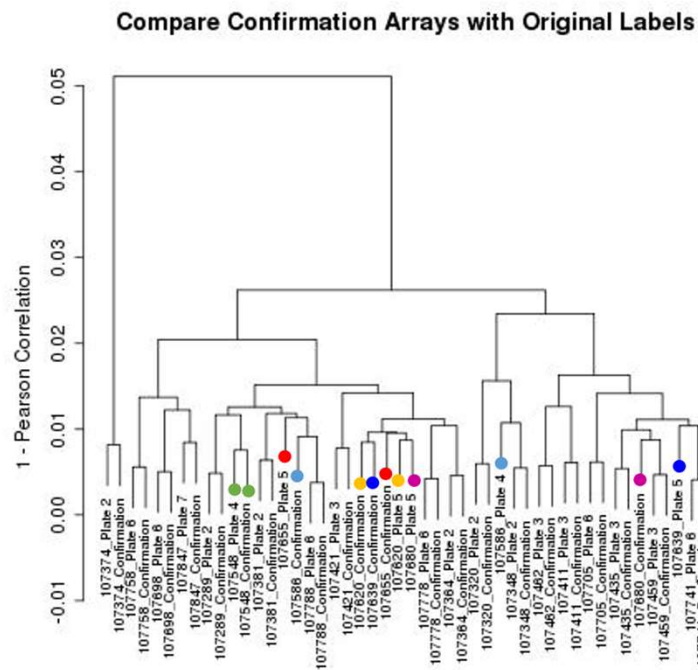
Sample name format: A1_SO1_S1

Visualizing Data – Dendrogram Example 2



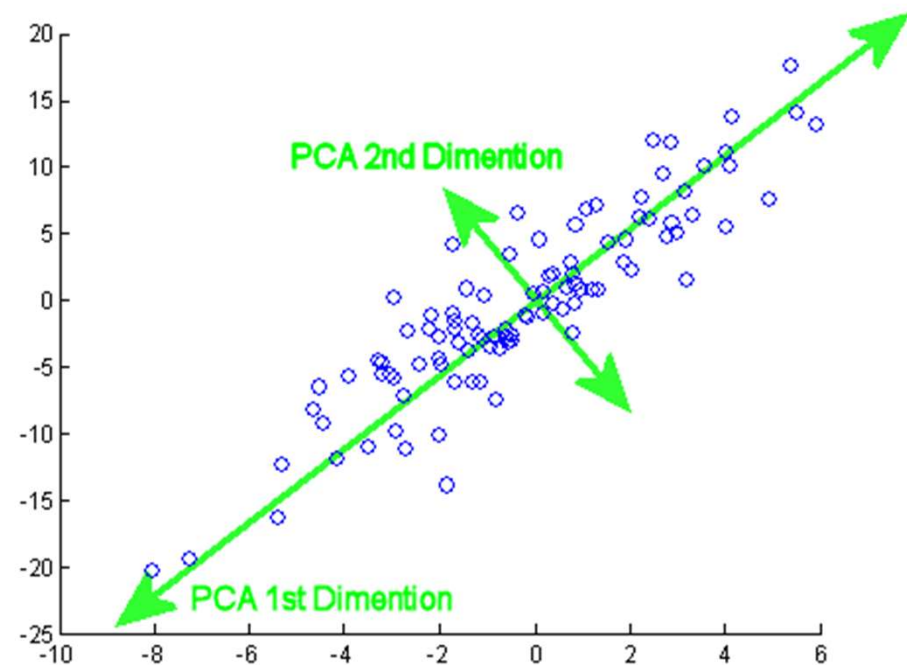
- Take away from this is there are some major batch effects we need to deal with
- Good to identify sample outliers
- Hard to actually see grouping with some many samples

Visualizing Data – Dendrogram Example 3



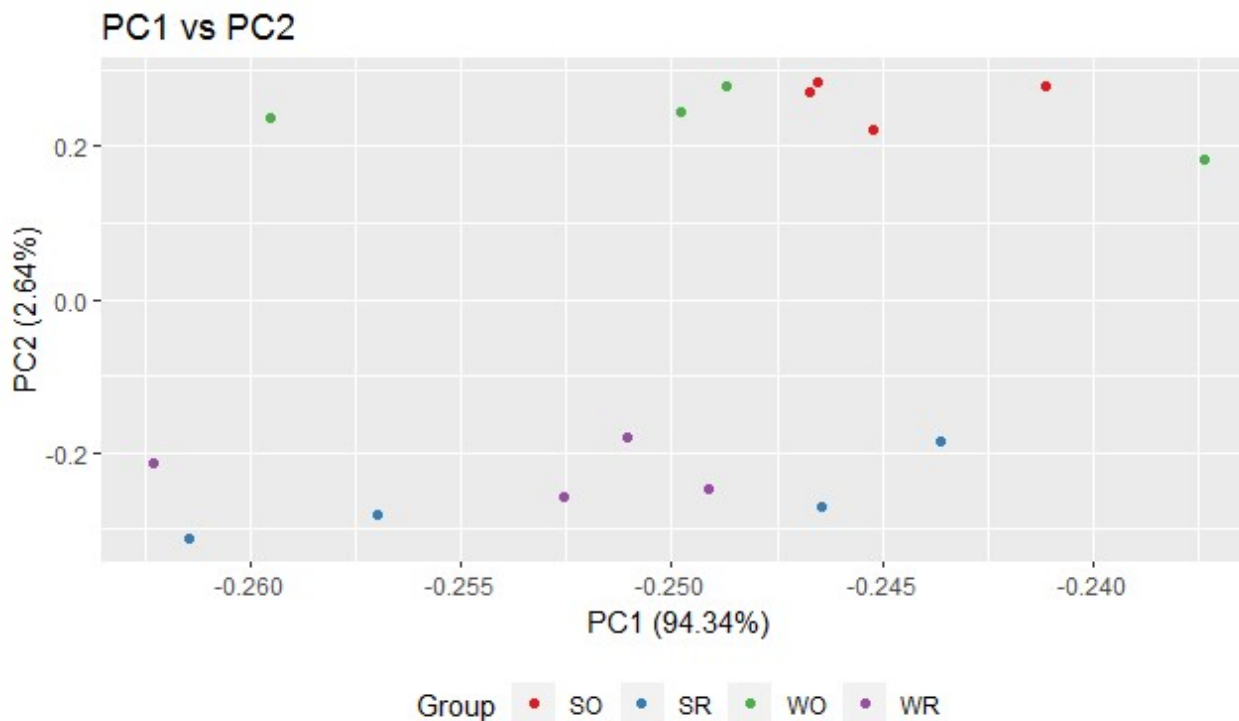
Principal Component Analysis (PCA)

- Good way to summarize your data
- Linear combinations of data which explain the most variance
- PC1 explains the most variance in your data, PC2 the second most
- Easy example with 2D space, what about 20K dimensions!



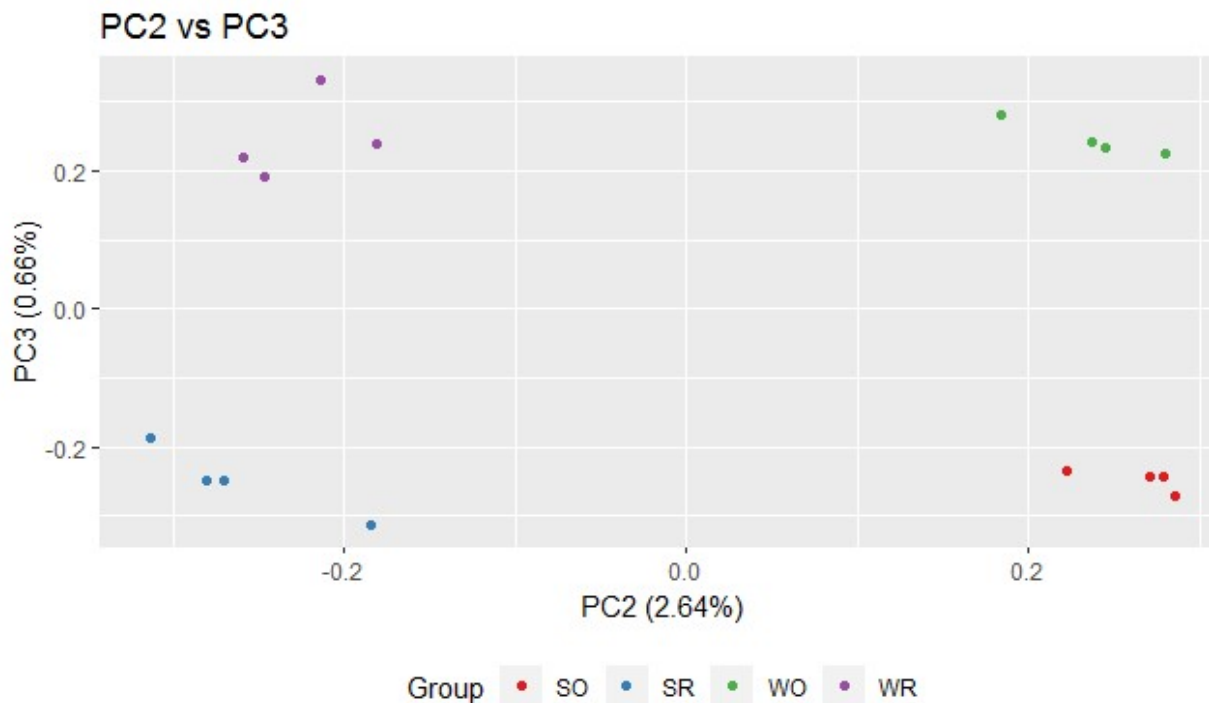
Source: stackExchange: making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

Visualizing Data – PC Plots Example 1



- # points = # samples
- This is the same data from the 16 sample dendrogram
- Large separation from tissue type on PC2
- For a homogeneous design (controlled genetic background), you do see a large % variance explained by PC1

Visualizing Data – PC Plots Example 1 Cont.



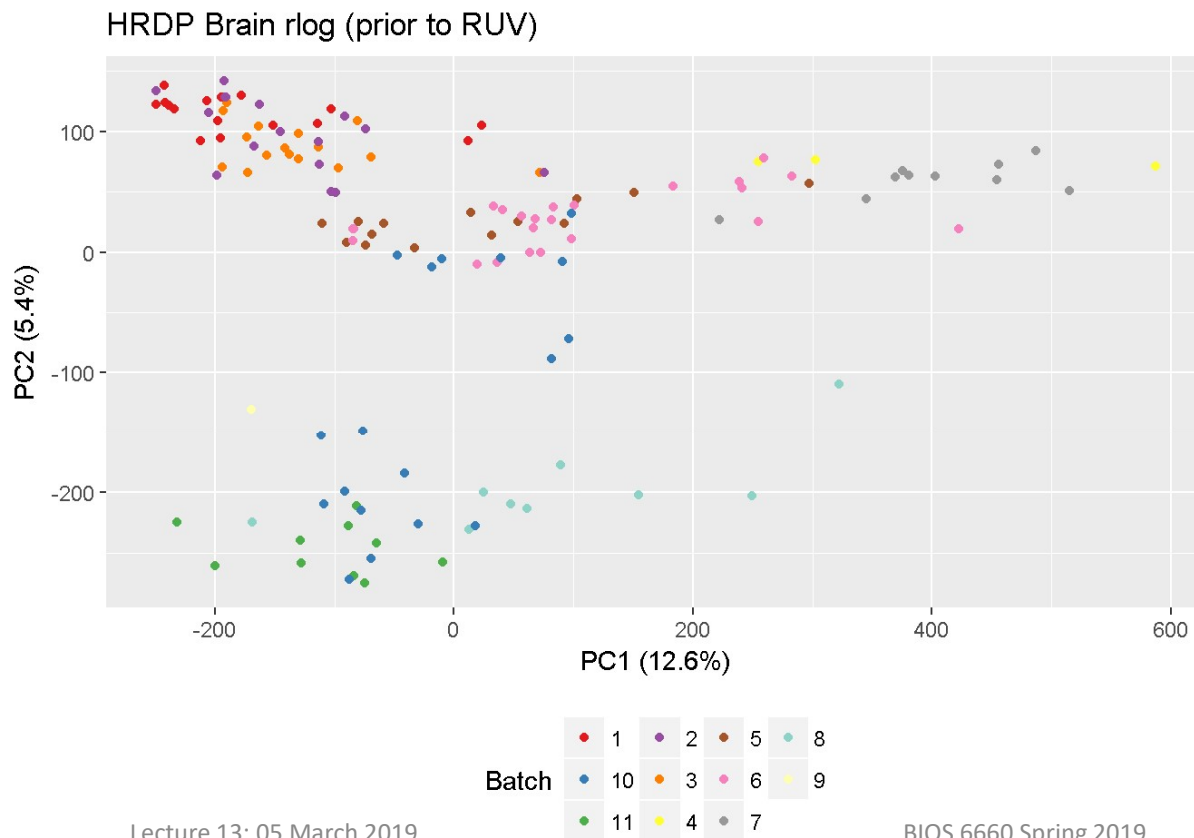
- Here you are seeing great separation of groups.
- You normally don't see this on PC1
- PC2 describes tissue differences (the 2nd letter in group)
- PC3 describes the strain differences (the 1st letter in group)

Visualizing Data – PC Plots Example1 Code

```
library(ggplot2)
library(RColorBrewer)
#perform pca
pca.results = prcomp(log2(counts2+1))
#get results you want
propEx = summary(pca.results)$importance #proportion var explained
toPlot = data.frame(pca.results$rotation[,1:2]) #PC1 and PC2
toPlot$group = substr(sapply(strsplit(rownames(toPlot), split="_", fixed=TRUE), "[[", 2), 1, 2)

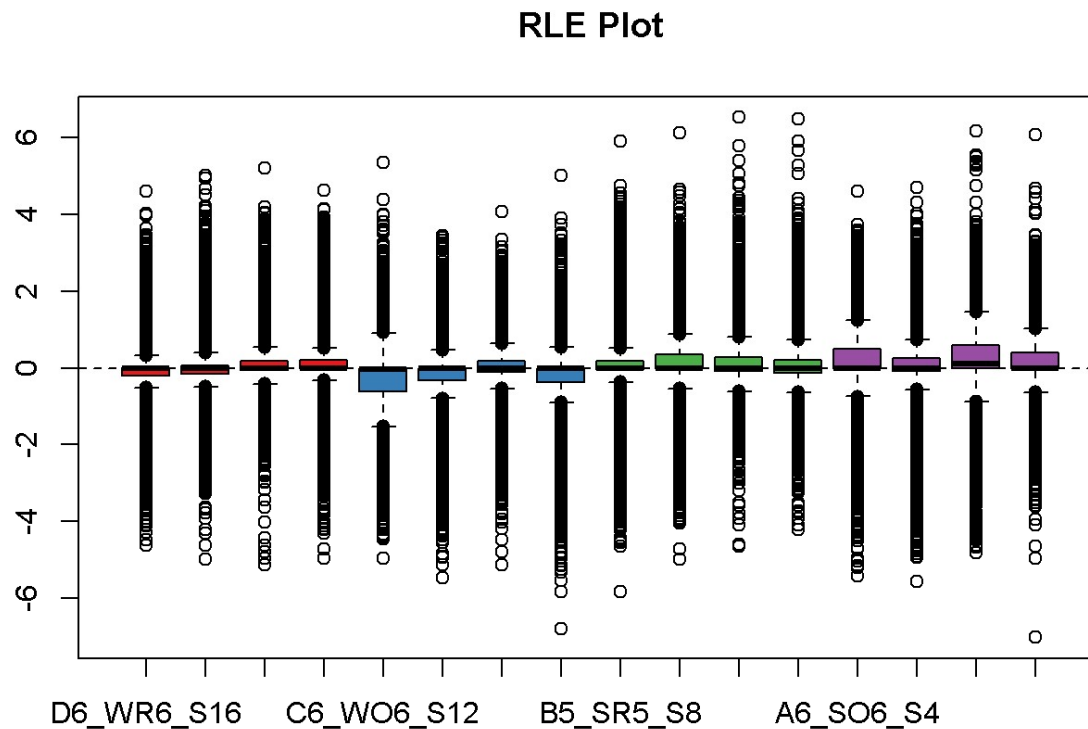
#make plot
colors= brewer.pal(9,"Set1") #define colors
orig = ggplot(toPlot, aes(x=PC1, y=PC2, color=as.character(group))) + geom_point() +
  scale_color_manual(values=colors[c(1:4)]) +
  xlab(paste("PC1 (", round(100*propEx[2,"PC1"], 2), "%)", sep="")) +
  ylab(paste("PC2 (", round(100*propEx[2,"PC2"], 2), "%)", sep="")) +
  ggtitle("PC1 vs PC2") +
  #theme(legend.position="none")
  labs(color="Group")+
  theme(legend.position="bottom")
orig
```

Visualizing Data – PC Plots Example 2



- Lot more samples
- Batch effects are clearly an issue
- PC1 explains lower % variance
 - Find this is the case the more complex the study design

Visualizing Data – RLE Plots



- Relative Log Expression
- For each gene, calculation the median logged expression
- These boxplots are the differences from the median value
- Originally looked at for microarrays, but the theory still stands
- Want the median RLE near 0
- ASSUMING # up-regulated genes = # down-regulated genes
 - Majority of genes DO NOT CHANGE in the system you are questioning

Visualizing Data – RLE Plot Code

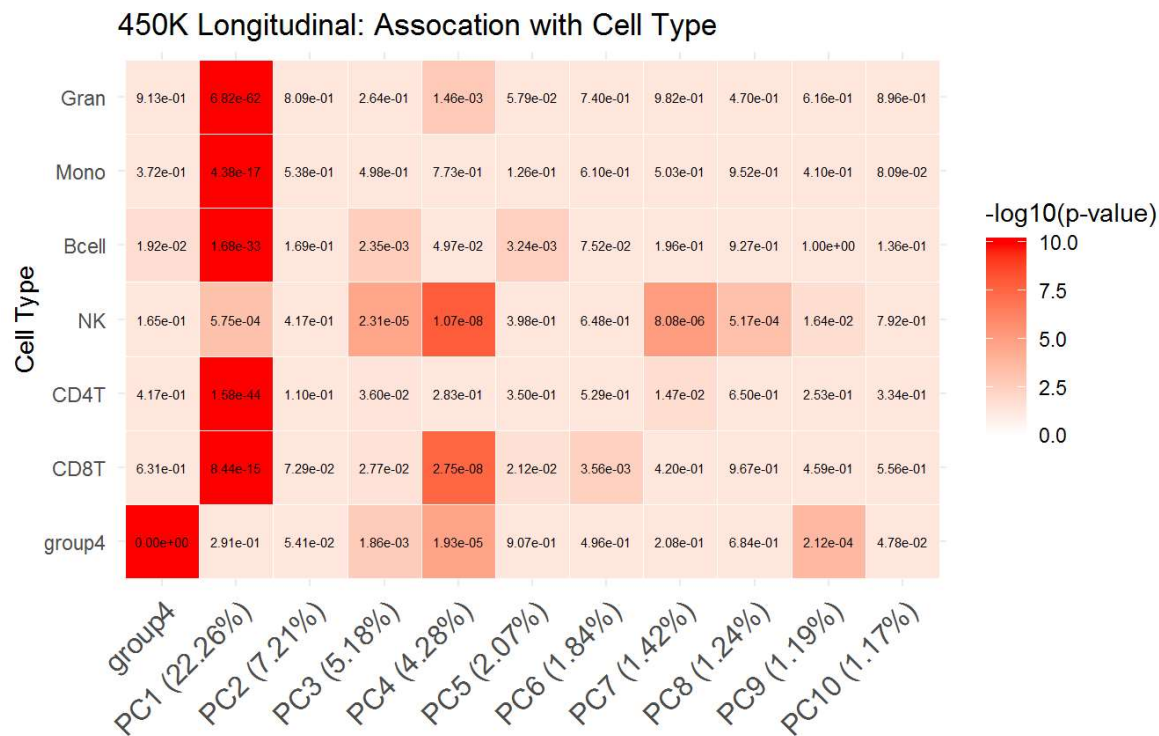
```
library(EDASeq)
```

```
library(RColorBrewer)
```

```
x = substr(sapply(strsplit(colnames(counts2), split="_", fixed=TRUE), "[", 2), 1, 2)

x.col=c()
for(i in 1:length(x)){
  if(x[i]=="WR"){x.col[i]=colorsWant[1]}
  if(x[i]=="WO"){x.col[i]=colorsWant[2]}
  if(x[i]=="SR"){x.col[i]=colorsWant[3]}
  if(x[i]=="SO"){x.col[i]=colorsWant[4]}
}
set = newSeqExpressionSet(as.matrix(round(counts2)), phenoData=data.frame(x, row.names=colnames(counts2)))
par(mfrow=c(1,1))
plotRLE(set, colLabel=x, main="RLE Plot", col=x.col)
```

Visualizing Data – Diagnostic Heatmap



Visualizing Data – Diagnostic Heatmap Code

```
#toPlot is a dataframe with 4 columns: PC, Cell Type, estimate and pvalue
toPlot$minus.log10pval[which(toPlot$pvalue>0.05)]=1.3
toPlot$pvalsToPrint = formatC(toPlot$pvalue, format = "e", digits = 2)

ggheatmap <- ggplot(toPlot, aes(PC, CellType, fill = minus.log10pval))+
  geom_tile(color = "white")+
  ggtitle("450K Longitudinal: Association with Cell Type") +
  scale_fill_gradient2(low = "white", high = "red", mid="white", na.value = "red",
    midpoint = 0.05, limit = c(0,10), space = "Lab",
    name="-log10(p-value)") +
  theme_minimal()+ # minimal theme
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
    size = 12, hjust = 1))+
  xlab("") +
  scale_x_discrete(labels=c("group4", paste("PC", c(1:10), " (", round(eig_pc_proportion[1:10], 2), "%)", sep=""))) +
  ylab("Cell Type") +
  coord_fixed()
# Print the heatmap
ggheatmap + geom_text(label = toPlot$pvalsToPrint, size=2)
```


Relative Abundance

- RNA-Seq is a **relative abundance** measurement technology

Gene	Sample 1 Absolute Abundance	Sample 1 Relative Abundance	Sample 2 Absolute Abundance	Sample 2 Relative Abundance
A	20	10%	20	5%
B	20	10%	20	5%
C	20	10%	20	5%
D	20	10%	20	5%
E	20	10%	20	5%
F	100	50%	300	75%
TOTAL 200 counts			TOTAL 400 counts	

Library Sizes Matter

Normalization Methods

- Need to do this to make samples comparable across features and samples
- Common Methods:
 - RPKM/FPKM
 - TPM
 - Quantile
 - TMM (EDASeq)
 - DESeq Median of Ratios (geometric mean & scaling factor)
 - RUV

RPKM/FPKM

- Reads Per Kilobase per Million or Fragments PKM
- One of the 1st normalization methods
- Adjusts for
 - Sequencing depth (or library size)
 - Length of gene (longer gene more reads)
- 1. Count up total number reads in sample and divide by a million (per million scaling factor)
- 2. Divide read counts by “per million” scaling factor. Normalizing for sequencing depth (RPM, reads per million)
- 3. Divide RPM by length of gene in Kb, this give you RPKM
- FPKM is for paired-end reads, really so the 2 reads from a single fragment of RNA is not counted twice.

TPM

- Transcripts Per Million
- Very similar to RPKM/FPKM but order of operations are switched
- Adjusts for length of gene first, then sequencing depth
 1. Divide counts by length of gene in Kb (RPK)
 2. Count up all the RPK values in a sample and divide this number by 1 million (this is the per million scaling factor).
 3. Divide the RPK values by “per million” scaling factor, this gives you TPM.
- Preferred over RPKM/FPKM since using TPM the total sum of gene TPM will be the same across all samples (i.e. really have the same denominator)
- RSEM outputs TPM

RPKM vs TPM example

Counts Matrix

Gene	Rep 1	Rep 2	Rep 3
A (2kb)	10	12	30
B (4kb)	20	25	60
C (1kb)	5	8	15
D (10kb)	0	0	1
TOTAL	35	45	106

Note: In this sample I used per 10 instead of per 1 million just to make the numbers easier to look at

Source: RNASeq blog

Your total is the sample across samples, so it's easier to compare across samples

RPKM Matrix

Gene	Rep 1	Rep 2	Rep 3
A (2kb)	1.43	1.33	1.42
B (4kb)	1.43	1.39	1.42
C (1kb)	1.43	1.78	1.42
D (10kb)	0	0	0.009
TOTAL	4.29	4.5	4.269

TPM Matrix

Gene	Rep 1	Rep 2	Rep 3
A (2kb)	3.33	2.96	3.326
B (4kb)	3.33	3.09	3.326
C (1kb)	3.33	3.95	3.326
D (10kb)	0	0	0.02
TOTAL	10	10	10

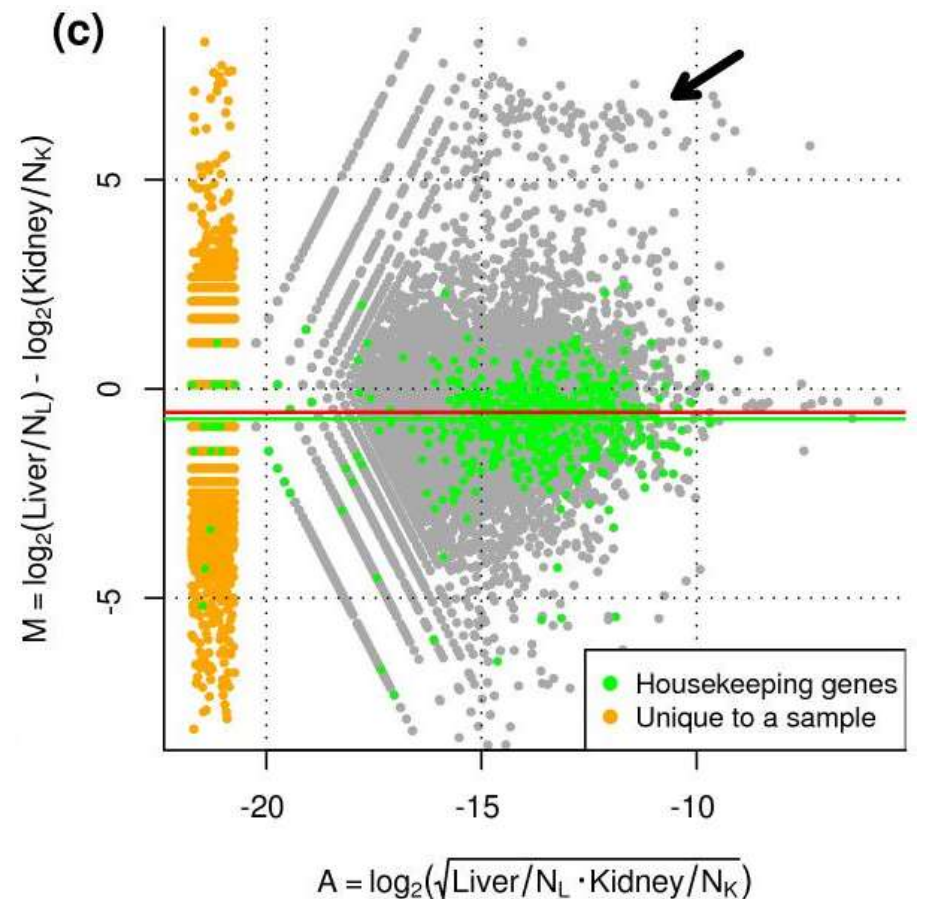
Quantile Normalization

- Making your distributions identical in statistical properties (i.e. the quantiles will be the same)
- Ranking the genes for each sample
- Get a rank expression level then each rank gets a new expression level

Rank	Sample 1	Sample 2	Sample N	Normalized Value
1	0	1	...	0	Mean 1 st row
2	3	2	...	2	Mean 2 nd row
...
G	10002	9999	...	9232	Mean Gth row

TMM (EDASeq)

- Weighted Trimmed Mean of Log Expression Ratios (M values)
- Very similar to a Bland-Altman plot
- Call A (x-axis) the reference sample (geometric mean of 2 samples)
- Red line should be at 0
- Where the line is the normalization factor



DESeq Median of Ratios

1. Normalized Count = count / geometric mean of that gene

Gene	Sample 1	Sample 2	Geometric Mean
A	100	50	70.7
B	300	220	256.9

→

Gene	Sample 1	Sample 2
A	1.41	0.71
B	1.17	0.86

2. Within sample normalization factor by taking the median across all genes.

Sample 1 normalization factor = $\text{median}(c(1.41, 1.17)) = 1.29$

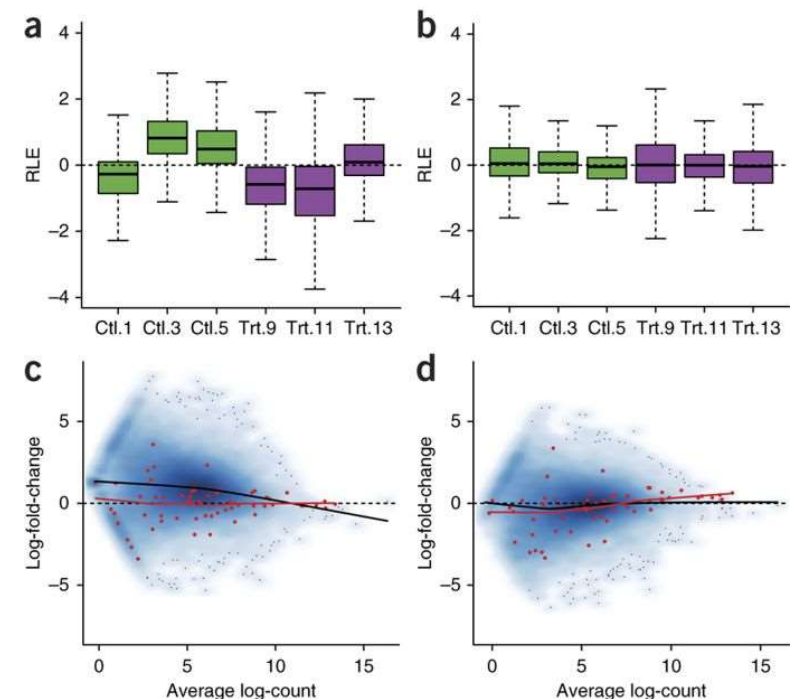
Sample 2 normalization factor = $\text{median}(c(0.71, 0.86)) = 0.785$

3. Final Normalized Matrix

Gene	Sample 1	Sample 2
A	$100/1.29 =$ 77.5	$50/0.785 =$ 63.7
B	$300/1.29 =$ 232.6	$220/0.785 =$ 280.3

RUV

- **R**emove **U**nwanted **V**ariation
- Idea is to use control genes that do not change in your design
 - Spike-ins
 - Biological knowledge
 - Empirical Method
- Uses factor analysis (very similar to PCA)
- Depending on how many control genes you have, you can adjust out for as many factors (k)

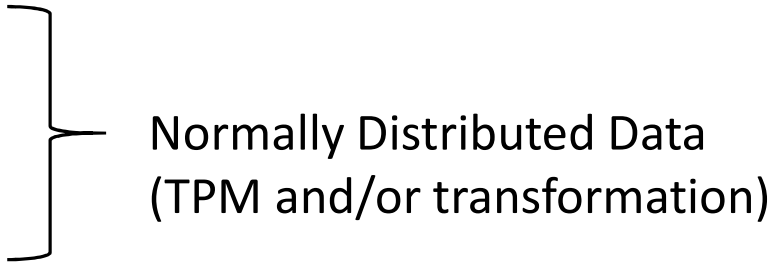


Source: Risso et al, 2014

Normalization Methods Summary

- DO NOT USE RPKM/FPKM
- Preferably keep data as count data and use either DESeq or EDASeq
 - These packages allow you to analyze the data using a GLM with assumption data is distributed with NB distribution
 - However, they don't allow you to have any random effects
- RUV is a newer exciting method
 - Selecting number factors to adjust out (k) is not straight forward
 - Many times used in conjunction with others

Differential Expression

- Basic analysis to find differences in expression between a group(s).
 - Common Analysis Techniques:
 - T-test
 - ANOVA
 - Linear Regression
 - Linear Mixed-Effect Models
 - Generalized Linear Models (GLM)
 - Link function so you can run data that is non-normal in a linear model structure
 - DESeq & EDASeq
- 
- Normally Distributed Data
(TPM and/or transformation)

DESeqDataSet Format

- S4 Type of object
- Stores the input values (raw counts), intermediate calculations (any normalization) and results of an analysis of differential expression (from DESeq or DESeq2)

```
dds2 = DESeqDataSetFromMatrix(countData = as.matrix(round(counts2)),  
                               colData = data.frame(group, tissue, strain),  
                               design = ~ strain*tissue)
```

- If you used HTSeq (python module) to generate counts, you can use `DESeqDataSetFromHTSeqCount()` to generate object

DESeqDataSet Format

```
> dds2
class: DESeqDataSet
dim: 15127 16
metadata(1): version
assays(1): counts
rownames(15127): FBgn0000003 FBgn0000008 ... FBgn0267794 FBgn0267795
rowData names(0):
colnames(16): D6_WR6_S16 D5_WR5_S15 ... A2_S02_S2 A1_S01_S1
colData names(3): group tissue strain
```

```
rawCountMatrix <- counts(dds2)
```

```
metaData <- colData(dds2)
```

DESeq2 Analysis - LRT example

#performs median of ratios normalization

dds2 = estimateSizeFactors(dds2)

#if you want to extract normalized count matrix

normalizedCounts = counts(dds2, normalized=TRUE)

#perform a likelihood ratio test (LRT) to see any strain effect

```
> dds2 = DESeq(dds2, test="LRT", reduced= ~ tissue, fitType="local")  
using pre-existing size factors  
estimating dispersions  
gene-wise dispersion estimates  
mean-dispersion relationship  
final dispersion estimates  
fitting model and testing  
> |
```

DESeq2 Analysis – LRT example

```
> strainEffects = results(dds2, independentFiltering=FALSE)
> head(strainEffects)
log2 fold change (MLE): strainWhiteEyed.tissueRetina
LRT p-value: '~ strain * tissue' vs '~ tissue'
DataFrame with 6 rows and 6 columns
```

	baseMean <numeric>	log2FoldChange <numeric>	lfcSE <numeric>	stat <numeric>
FBgn0000003	699.145888361084	-0.58771314044907	0.693613552000856	2.61212587909614
FBgn0000008	2954.45567305398	-0.895597749628288	0.250017864368627	14.7137213480259
FBgn0000014	8.21095639029486	1.24812287239828	1.96457143573063	2.64892501978943
FBgn0000015	3.79786939660886	4.08364954314244	2.65369361537612	2.26799339087655
FBgn0000017	5671.9724202071	-0.410359966390261	0.274445280382548	4.97547427058328
FBgn0000018	113.196237762821	-0.418801337528481	0.377321859183718	3.18711565071041

	pvalue <numeric>	padj <numeric>
FBgn0000003	0.27088444817034	0.607569563854795
FBgn0000008	0.000638198832163481	0.00746430224491593
FBgn0000014	0.265945863950927	0.601885243945011
FBgn0000015	0.321744767444056	0.67099517415778
FBgn0000017	0.0830977930453044	0.294911094150426
FBgn0000018	0.203201369150486	0.515821492054677

```
> |
```


DESeq2 Analysis – single coefficient example

```
> dds.inter = DESeqDataSetFromMatrix(countData = as.matrix(round(counts2)),  
+                                     colData = data.frame(group, tissue, strain),  
+                                     design = ~ strain*tissue)  
converting counts to integer mode  
Warning message:  
In DESeqDataSet(se, design = design, ignoreRank) :  
  some variables in design formula are characters, converting to factors  
> dds.inter = DESeq(dds.inter)  
estimating size factors  
estimating dispersions  
gene-wise dispersion estimates  
mean-dispersion relationship  
final dispersion estimates  
fitting model and testing  
> interactionEffect = results(dds.inter, name="strainWhiteEyed.tissueRetina",  
                             independentFiltering=FALSE)
```


DESeq2 Analysis – single coefficient example

```
> head(interactionEffect)
log2 fold change (MLE): strainWhiteEyed.tissueRetina
Wald test p-value: strainWhiteEyed.tissueRetina
DataFrame with 6 rows and 6 columns
```

	baseMean <numeric>	log2FoldChange <numeric>	lfcSE <numeric>	stat <numeric>
FBgn0000003	699.145888361084	-0.587558882866976	0.632132296455124	-0.929487207285395
FBgn0000008	2954.45567305398	-0.895529730693888	0.270234151087478	-3.31390287678331
FBgn0000014	8.21095639029486	1.24910089991614	2.02038098532741	0.61825017607445
FBgn0000015	3.79786939660886	4.0786729090029	2.74465670088278	1.48604118966538
FBgn0000017	5671.9724202071	-0.410361872640175	0.293033034643545	-1.40039457714845
FBgn0000018	113.196237762821	-0.418655048790763	0.392255275774135	-1.06730253140516

	pvalue <numeric>	padj <numeric>
FBgn0000003	0.352636651914663	0.920261409774183
FBgn0000008	0.000920034588316088	0.0399724537563497
FBgn0000014	0.536410439428845	0.992603703499596
FBgn0000015	0.137268226396593	0.66180229649985
FBgn0000017	0.16139519302746	0.70510860946658
FBgn0000018	0.28583524538277	0.864727389563077

```
>
```

DESeq2 Analysis – Adjusting for Covariate

```
> dds.cov = DESeqDataSetFromMatrix(countData = as.matrix(round(counts2)),  
+                                colData = data.frame(group, tissue, strain),  
+                                design = ~ tissue + strain)  
converting counts to integer mode  
Warning message:  
In DESeqDataSet(se, design = design, ignoreRank) :  
  some variables in design formula are characters, converting to factors  
> dds.cov = DESeq(dds.cov)  
estimating size factors  
estimating dispersions  
gene-wise dispersion estimates  
mean-dispersion relationship  
final dispersion estimates  
fitting model and testing  
> strain.adjForTissue = results(dds.cov)  
> |
```

DESeq2 Analysis – Adjusting for Covariate

```
> strain.adjForTissue = results(dds.cov)
> head(strain.adjForTissue)
log2 fold change (MLE): strain WhiteEyed vs Sevenless
Wald test p-value: strain WhiteEyed vs Sevenless
DataFrame with 6 rows and 6 columns
```

results() has a name option for selecting coefficient, but default it automatically returns last coefficient in model

	baseMean <numeric>	log2FoldChange <numeric>	lfcSE <numeric>	stat <numeric>
FBgn0000003	699.145888361084	0.481065789920314	0.315813565099327	1.52325879279129
FBgn0000008	2954.45567305398	0.180576506548756	0.173719729223854	1.03947034315294
FBgn0000014	8.21095639029486	-1.51479593858499	0.982522214724989	-1.54174217730943
FBgn0000015	3.79786939660886	-0.0561051226257818	1.37892407109896	-0.0406876084054923
FBgn0000017	5671.9724202071	0.227839409937837	0.152736766573332	1.49171293231775
FBgn0000018	113.196237762821	0.264910424695025	0.19763036163017	1.3404338407818

	pvalue <numeric>	padj <numeric>
FBgn0000003	0.127693972189392	0.353716203604252
FBgn0000008	0.298586043835595	0.567851795440525
FBgn0000014	0.123136258247327	0.346438222750232
FBgn0000015	0.967544940475883	0.987751006971347
FBgn0000017	0.135774417382166	0.365765984149215
FBgn0000018	0.180104339899211	0.430512887063497

References

Robinson MD, Oshlack A. *A scaling normalization method for differential expression analysis of RNA-seq data*. Genome Biol. **2010**;11(3):R25.

Anders S, Huber W. *Differential expression analysis for sequence count data*. Genome Biol. **2010**;11(10):R106.

Risso D, Ngai J, Speed T, Dudoit S (2014). “Normalization of RNA-seq data using factor analysis of control genes or samples.” *Nature Biotechnology*, **32**(9), 896–902.