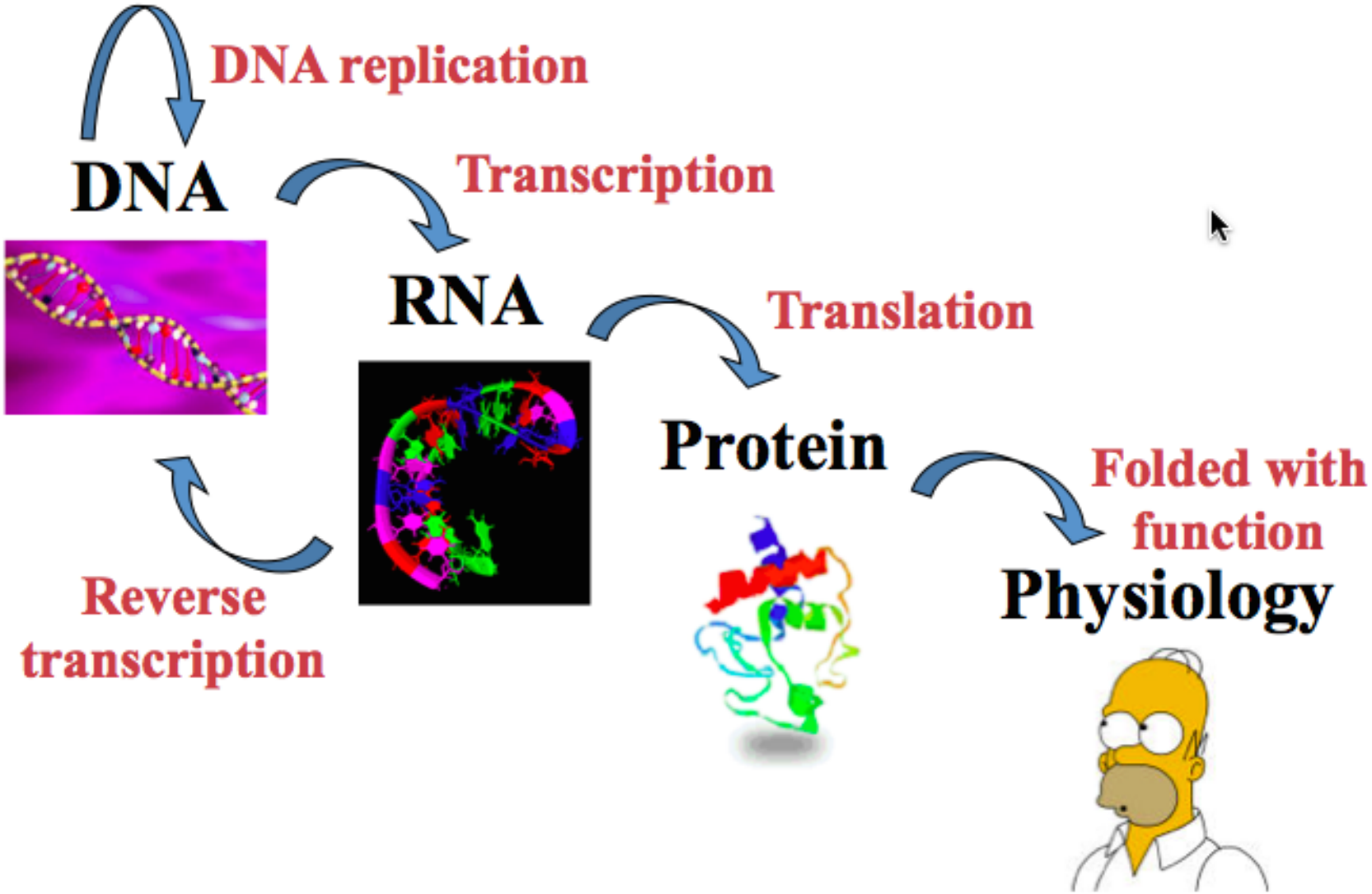
Bioinformatics Application: RNA-seq Data Analysis in R

Instructor: Mary Yang, PhD

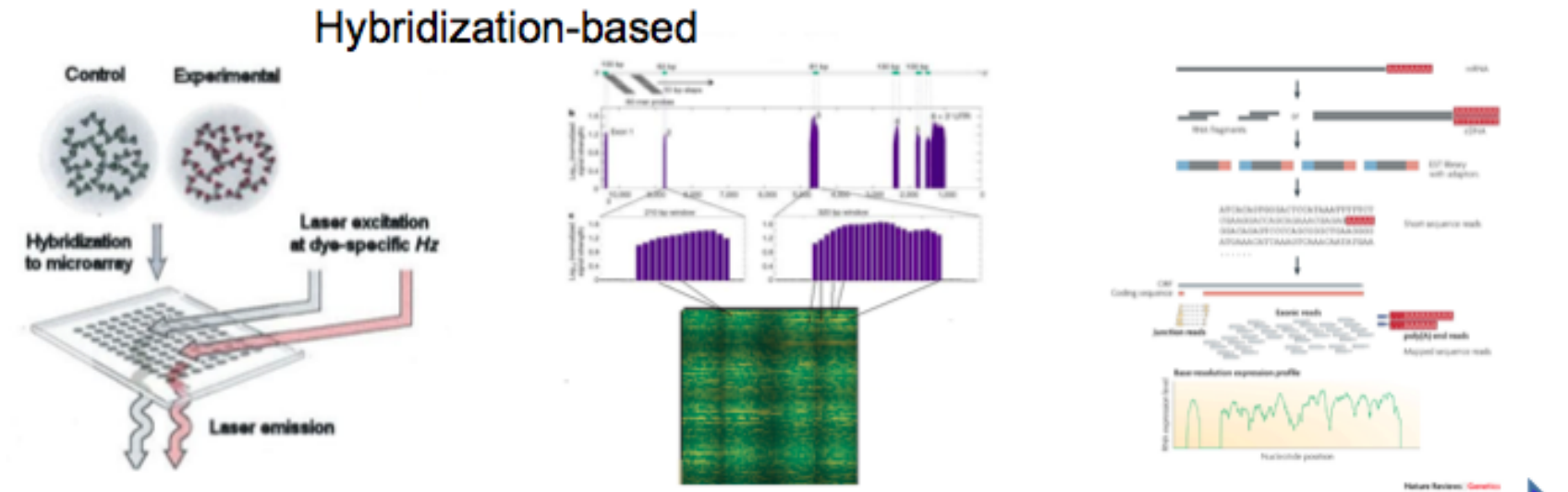
Graduate Assistant: Dan Li and Yifan Zhang

August 3, 2017

Central Dogma of Molecular Biology



The evolution of transcriptomics



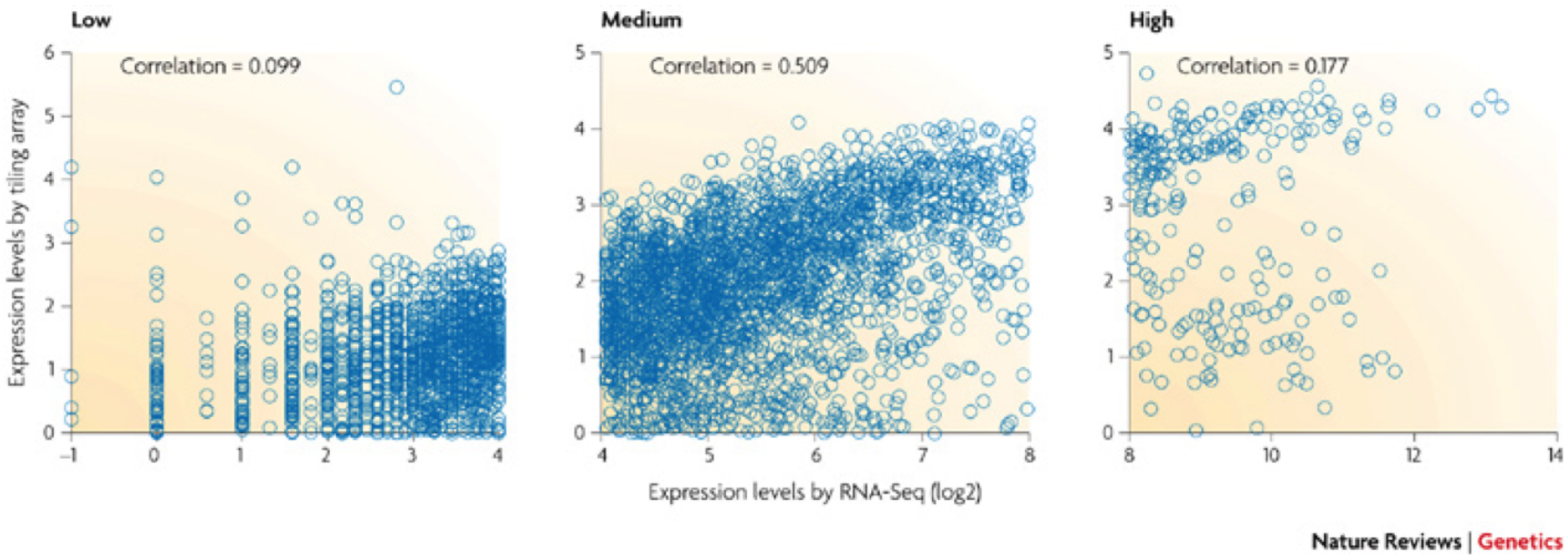
RNA-seq is still a technology under active development

1995 P. Brown, et. al. Gene expression profiling using spotted cDNA microarray: expression levels of known genes

2002 Affymetrix, whole genome expression profiling using tiling array: identifying and profiling novel genes and splicing variants

2008 many groups, mRNA-seq: direct sequencing of mRNAs using next generation sequencing techniques (NGS)

RNA-seq and Microarray agree fairly well only for genes with medium levels of expression

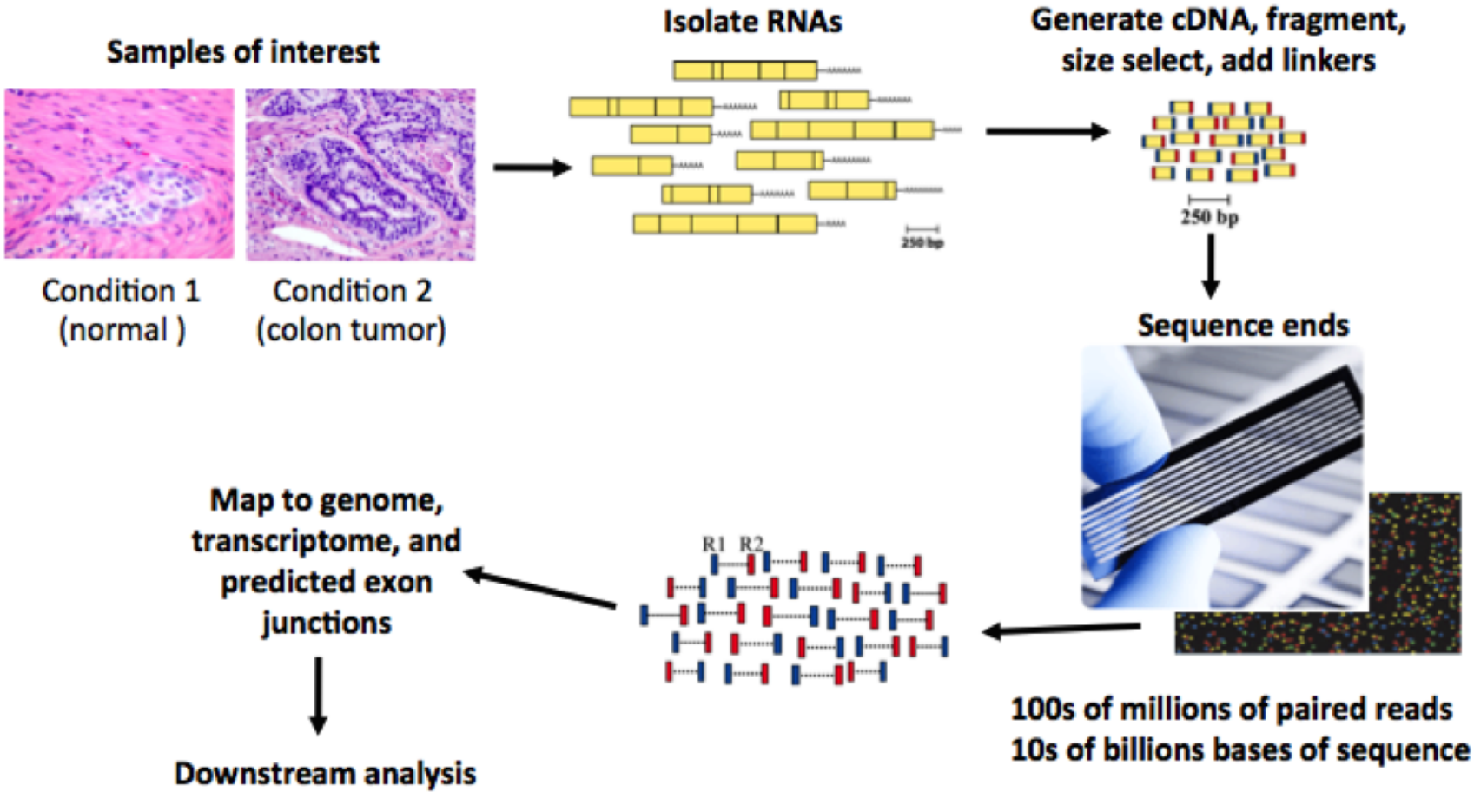


Correlation is very low for genes with either low or high expression levels.

What is RNA-seq?

- RNA-Seq is an approach to characterization and quantification transcriptomes that uses next generation sequencing technologies.
- RNA-seq does not rely on prior knowledge of gene structures. It can be used to
  - identify novel transcripts
  - detect alternative splicing
  - profile the expression levels of known transcripts
  - detect single nucleotide polymorphisms (SNP)
  - non-coding RNA
  - gene fusion

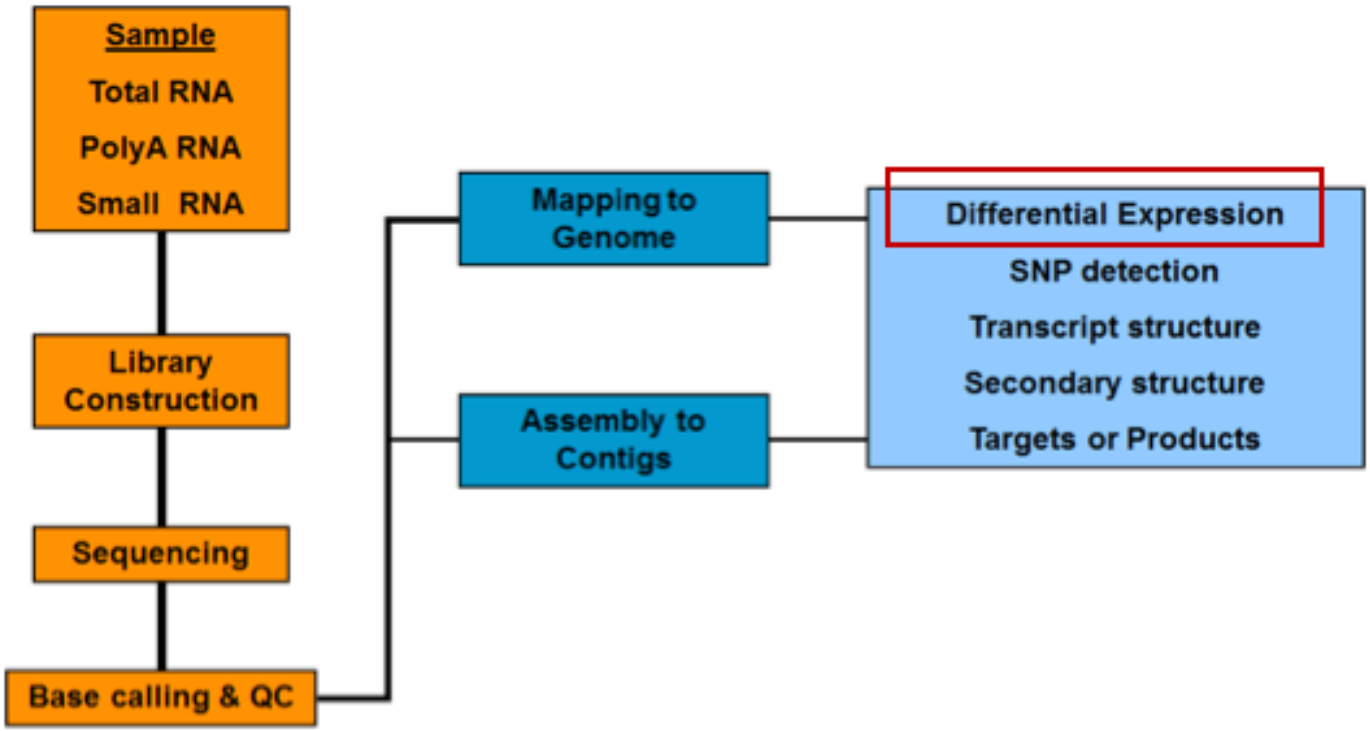
RNA-seq


Samples of interest
Condition 1
(normal)
Condition 2
(colon tumor)
Isolate RNAs
250 bp
Generate cDNA, fragment,
size select, add linkers
250 bp
Sequence ends
100s of millions of paired reads
10s of billions bases of sequence
R1 R2
Map to genome,
transcriptome, and
predicted exon
junctions
Downstream analysis

RNA-seq

- Methods capable of giving a "snapshot" of RNA expression of all genes
- Can be used as diagnostic profile
  - Example: cancer diagnosis
- Can show how RNA levels change during development, after exposure to stimulus, during cell cycle, etc.
- Can help us start to understand how whole systems function

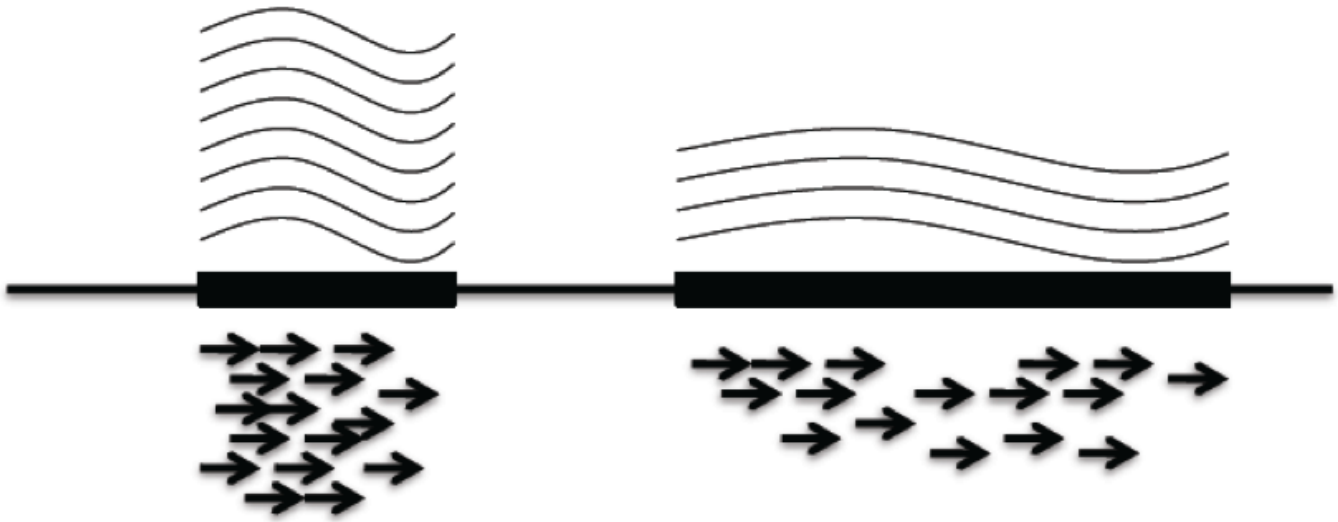
RNA-seq work flow



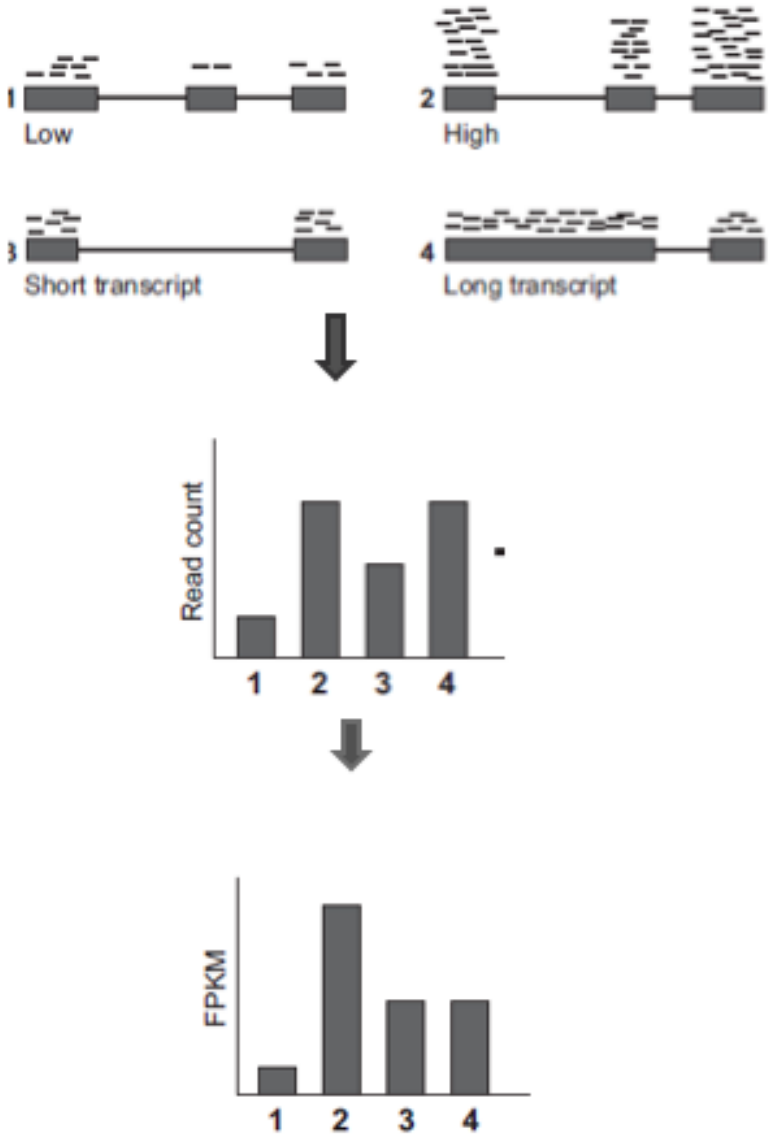
RNA-seq: gene expression

Quantify gene expression from RNA-seq data

- Read count is linearly related to the abundance of the target transcript
- Count the number of reads fall in the transcripts



RNA-seq: Normalization



Garber et.al *Nature Methods*. Nature 2010

RNA-seq: Normalization

- A quantification of gene expression level
  - RPKM (Reads Per Kilo-base exon model per Million mapped reads)

$$RPKM = \frac{C}{\frac{N}{10^6} * \frac{L}{10^3}} \quad (1)$$

`C: total reads falls into the gene region`
`N: total reads`
`L:length of the gene`

- FPKM (Fragments Per Kilo-base exon model per Million mapped reads for paired-end reads)
- Normalization: An attempt to exclude systematic variation by statistical methods

Source of variation in RNA-seq data

Systematic variation in RNA-seq experiments

- Between−sample difference
  - Larger library sizes result in higher counts for the entire sample
  - RNA composition
- Within−sample gene−specific effects
  - Gene length
  - GC−contents

RNA seq data normalization

- Total read count normalization (TC)
  - Assumption: read counts are proportional to expression levels
- RPKM, FPKM
  - Assumption: read counts are proportional to expression level and gene length.
- Upper Quartile normalization (UQ)
  - Assumption: read counts are proportional to expression level, and total read count is strongly dependent on highly expressed transcripts.
- TMM (Robinson and Oshlack, 2010). Trimmed Mean of M values
  - Assumption: majority of transcripts are not differentially expressed.

Differential expression

- Two experimental conditions
  - Treated versus untreated
- Two distinct phenotypes
  - Tumor versus normal tissue

Assessment of differential expression

- Fold change:
  - How large is the expression difference found?
- P−value:
  - How sure are we that a true difference exists?

Count-based methods (R packages)

- DESeq: based on negative binomial distribution
- edgeR: use an overdispersed Poisson model
- baySeq: use an empirical Bayes approach
- TSPM: use a two-stage poisson model

edgeR

- edgeR is a Bioconductor package that performs differential gene expression analysis using count data under a negative binomial model.
- The software works on a table of integer read counts, with rows corresponding to genes and columns to independent libraries.
- The counts represent the total number of reads aligning to each gene.
- The methods used in edgeR do not support FPKM, RPKM or other types of data that are not counts.

Install edgeR

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite("edgeR")
> library(edgeR)
```

Lung cancer RNA-seq data set from TCGA

- A lung cancer RNAseq data example
  - 56 normal
  - 56 tumor
- Goal: Identify the differentially expressed genes between normal and tumor samples.

Reading data

```
#Change directory to the lung cancer directory
> setwd("~/Mary_Yang/Workshop/Topic 4/Data")

> geneCount = read.csv ("Lung_Cancer_GeneCount.csv")
> dim (geneCount)
[1] 3000 113
> geneCount[1:10, 1:8]
```

```
> geneCount[1:10, 1:8]
   geneSymbol normal normal.1 normal.2 normal.3 normal.4 normal.5 normal.6
1        RGS22    302      155       88       42      150       41      596
2         HPSE    210      184      400      339      407      131      457
3        BCAS4    296      251      316      275      369      256      444
4     TNFRSF8     39       68      112       55      120       74       40
5      CLEC3A      0        0        0        0        0        0        1
6        LRWD1    483      449      329      331      708      374      612
7         PAX6    105      119      117      105      153      130      119
8         NEK1    520      519      595      603      792      378      643
9      RPS6KB2    753     1228      827      846     1428      959     1262
10      ANKRD9    720      964      795     1070     1122      573      697
```

Data processing

- Filter out the nonexpressed genes.
  - For simplicity, we consider only the genes with an average read count of 10 or more.

```
> means <- rowMeans(geneCount[,-1])
> filter <- means >= 10
> table(filter)
filter
FALSE TRUE
  530 2470

#Exclude the genes with an average count less than 10,
# and delete the first column which is for gene name
> geneCountHigh <- geneCount[filter,-1]
> dim(geneCountHigh)
[1] 2470 112
```

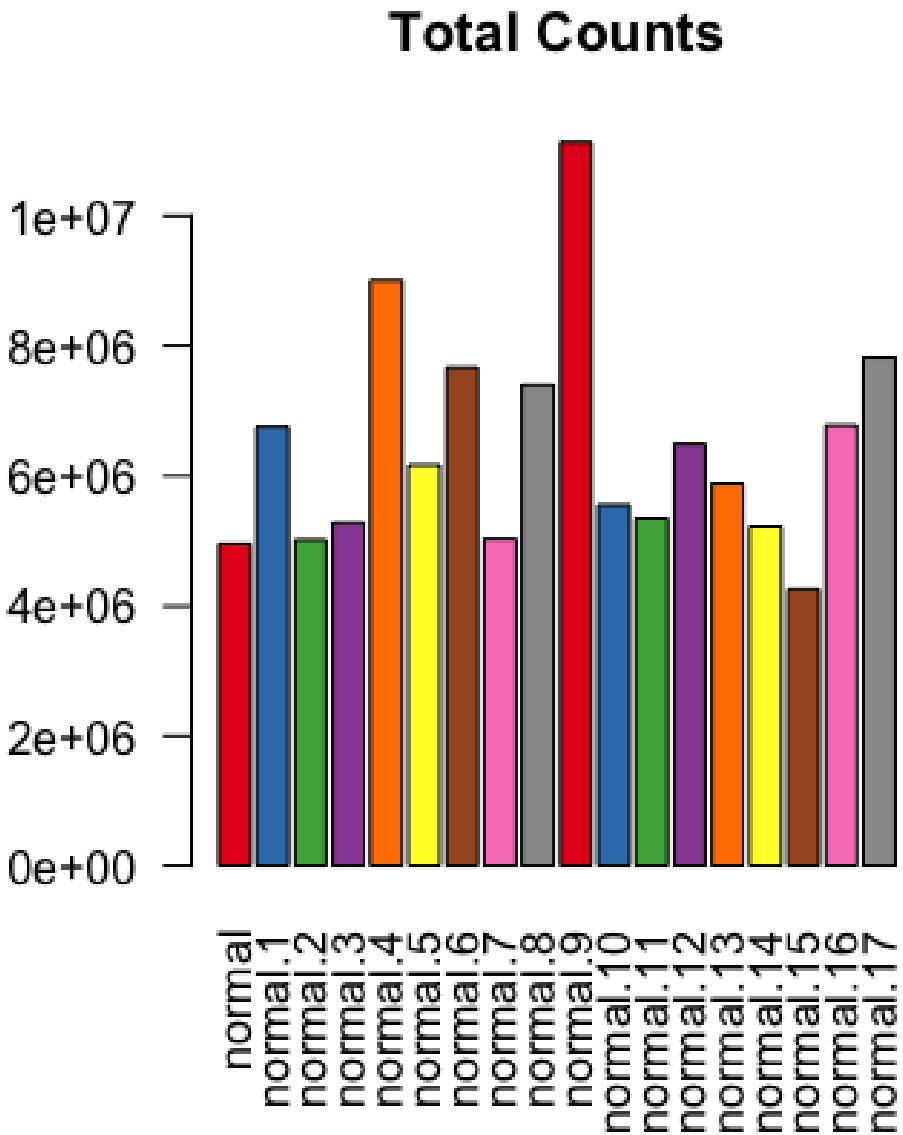
Library sizes

- One of the main characteristics of RNA-seq data: the sequencing depths or library size are varied
- We can visualize the total number of mapped reads to known genes with the barplot() function.
- Further, to check for systematic effects we can color-code the plot by different biological or technical variables.

Total count of each sample

```
> library(RColorBrewer)
> colors <- brewer.pal(9, "Set1")

#We test the first 18 samples only.
> totCounts <- colSums(geneCountHigh[,1:18])
> barplot(totCounts, las=2, col=colors, main = "Total Counts")
```

Total count of each sample

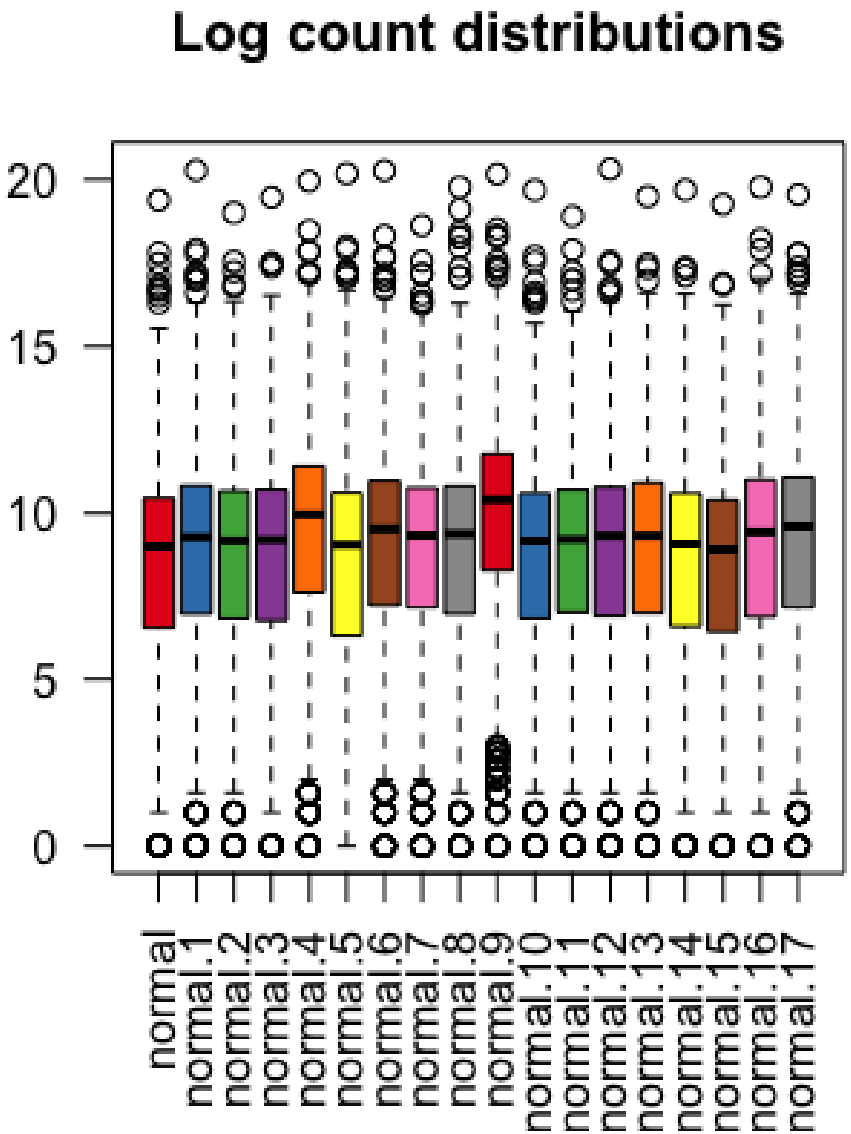


The count distribution of each sample

- The **boxplot()** function provides an easy way to visualize the difference in distribution between each experiment.

```
> boxplot(log2(geneCountHigh[,1:18] + 1), las=2, col=colors, main
    = "Log count distributions")
```

The count distribution of each sample



Building the edgeR object

- **DGEList()** is the function that converts the count matrix into an edgeR object.
- **DGEList()** creates a object from a table of counts (rows=features, columns=samples), **group** indicator for each column, library size (optional) and a table of feature annotation (optional).
- In addition to the counts, we need to group the samples according to the variable of interest in our experiment. Here, we compare the normal and tumor samples.

The types of samples

DGEList(counts,lib.size, norm.factors, samples, group = NULL ...)

```
counts:       numeric matrix of read counts.
group:        vector or factor giving the experimental group/condition
              for each sample
lib.size :    numeric vector giving the total count (sequence depth)
              for each library.
norm.factors: numeric vector of normalization factors that modify
              the library sizes.
```

The groups of samples

```
> colnames (geneCountHigh)
  [1] "normal"    "normal.1"  "normal.2"  "normal.3"  "normal.4"
  [6] "normal.5"  "normal.6"  "normal.7"  "normal.8"  "normal.9"
 [11] "normal.10" "normal.11" "normal.12" "normal.13" "normal.14"
 [16] "normal.15" "normal.16" "normal.17" "normal.18" "normal.19"
 [21] "normal.20" "normal.21" "normal.22" "normal.23" "normal.24"
 [26] "normal.25" "normal.26" "normal.27" "normal.28" "normal.29"
 [31] "normal.30" "normal.31" "normal.32" "normal.33" "normal.34"
 [36] "normal.35" "normal.36" "normal.37" "normal.38" "normal.39"
 [41] "normal.40" "normal.41" "normal.42" "normal.43" "normal.44"
 [46] "normal.45" "normal.46" "normal.47" "normal.48" "normal.49"
 [51] "normal.50" "normal.51" "normal.52" "normal.53" "normal.54"
 [56] "normal.55" "tumor"     "tumor.1"   "tumor.2"   "tumor.3"
 [61] "tumor.4"   "tumor.5"   "tumor.6"   "tumor.7"   "tumor.8"
 [66] "tumor.9"   "tumor.10"  "tumor.11"  "tumor.12"  "tumor.13"
 [71] "tumor.14"  "tumor.15"  "tumor.16"  "tumor.17"  "tumor.18"
 [76] "tumor.19"  "tumor.20"  "tumor.21"  "tumor.22"  "tumor.23"
 [81] "tumor.24"  "tumor.25"  "tumor.26"  "tumor.27"  "tumor.28"
 [86] "tumor.29"  "tumor.30"  "tumor.31"  "tumor.32"  "tumor.33"
 [91] "tumor.34"  "tumor.35"  "tumor.36"  "tumor.37"  "tumor.38"
 [96] "tumor.39"  "tumor.40"  "tumor.41"  "tumor.42"  "tumor.43"
[101] "tumor.44"  "tumor.45"  "tumor.46"  "tumor.47"  "tumor.48"
[106] "tumor.49"  "tumor.50"  "tumor.51"  "tumor.52"  "tumor.53"
[111] "tumor.54"  "tumor.55"
```

Building the edgeR object

```
> counts = geneCountHigh
> group <- c(rep("normal", 56), rep("tumor", 56))
> cds <- DGEList(counts, group = group)
> class (cds)
[1] "DGEList"
attr(,"package")
[1] "edgeR"

#We can then see the elements that the object contains by using
    names() function
> names(cds)
[1] "counts" "samples"
```

Accessing the elements of an R object

```
> class (cds$counts)
[1] "matrix"
> class (cds$samples)
[1] "data.frame"
```

```
> head(cds$counts[,1:5])
  normal normal.1 normal.2 normal.3 normal.4
1    302      155       88       42      150
2    210      184      400      339      407
3    296      251      316      275      369
4     39       68      112       55      120
6    483      449      329      331      708
7    105      119      117      105      153
```

```
> head(cds$samples)
           group lib.size norm.factors
normal    normal  4959458            1
normal.1  normal  6758098            1
normal.2  normal  5022197            1
normal.3  normal  5277385            1
normal.4  normal  9007296            1
normal.5  normal  6168404            1
```

Normalization

- Using TMM normalization to account for compositional difference between the libraries

```
#Calculate normalization factors to scale the raw library sizes
> cds <- calcNormFactors(cds)
```

```
 > head(cds$samples)
          group lib.size norm.factors
normal   normal  4959458    0.9051502
normal.1 normal  6758098    0.8496215
normal.2 normal  5022197    1.0444084
normal.3 normal  5277385    0.9892869
normal.4 normal  9007296    0.9742291
normal.5 normal  6168404    0.8026885
```

Normalization

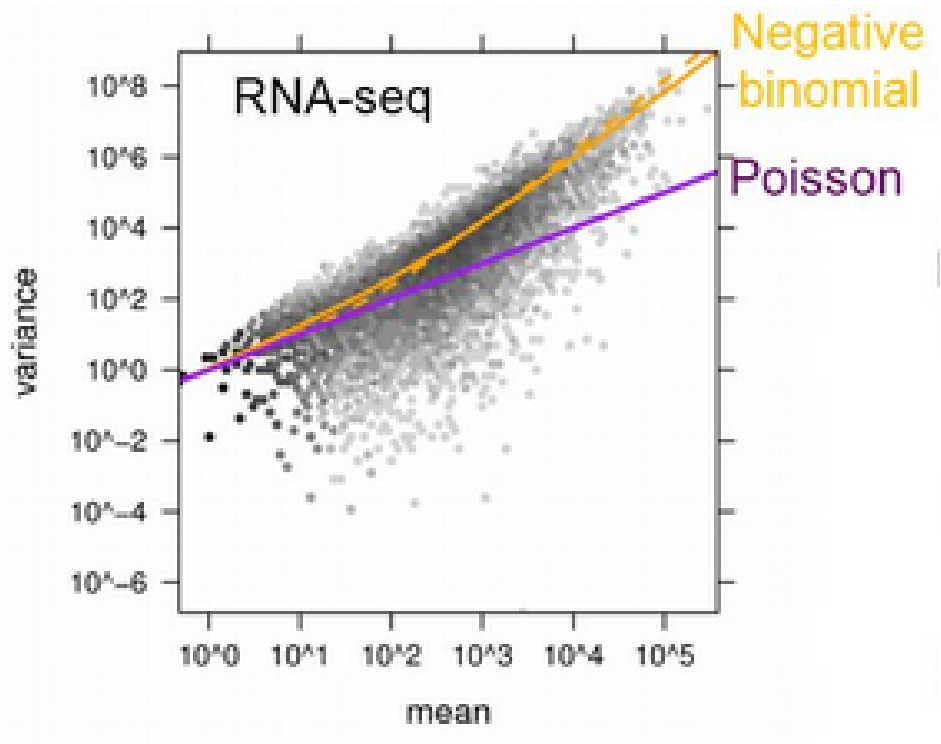
- By default, the function **calcNormFactors** normalize the data using the "weighted trimmed mean of M-values" (TMM) method
- Other options are RLE (relative log expression)and upper-quartile.
- If we want to use the upper-quartile to normalize, we can add an extra argument to the function.

```
#Calculate normalization factors to scale the raw library sizes
> cds <- calcNormFactors(cds, method="upperquartile")
```

```
> head(cds$samples)
          group lib.size norm.factors
normal   normal  4959458    0.9051568
normal.1 normal  6758098    0.8688050
normal.2 normal  5022197    1.0497513
normal.3 normal  5277385    1.0274628
normal.4 normal  9007296    0.9790930
normal.5 normal  6168404    0.8236127
```

Variation

- When assessing differential expression, it is important to model the variability in the data appropriately
- The negative binomial (NB) mode is used as more variation in RNA-seq data than can be accounted for by the Poisson model (called overdispersion).



Variation

- The dispersion parameter parameter is very important as it determines model the variance for each gene is modeled.
- The variance function for each gene is

$$V = mu * (1 + dispersion * mu) \tag{2}$$

where each gene has a distinct value for the mean (mu), which corresponds to the abundance of that gene in the RNA sample

Dispersion

- The dispersion is essential for modelling the variance of each gene
- Under the **common dispersion model** we use the same value for the dispersion when modelling the variance for each gene.
- Under a **tagwise model** we allow for a different value for the dispersion to be used for each gene

Estimate of the common dispersion

```
> cds <- estimateCommonDisp(cds)
> names(cds)
[1] "counts" "samples" "common.dispersion"
[4] "pseudo.counts" "pseudo.lib.size" "AveLogCPM"

> cds$common.dispersion
[1] 0.4116277
```

Common dispersion

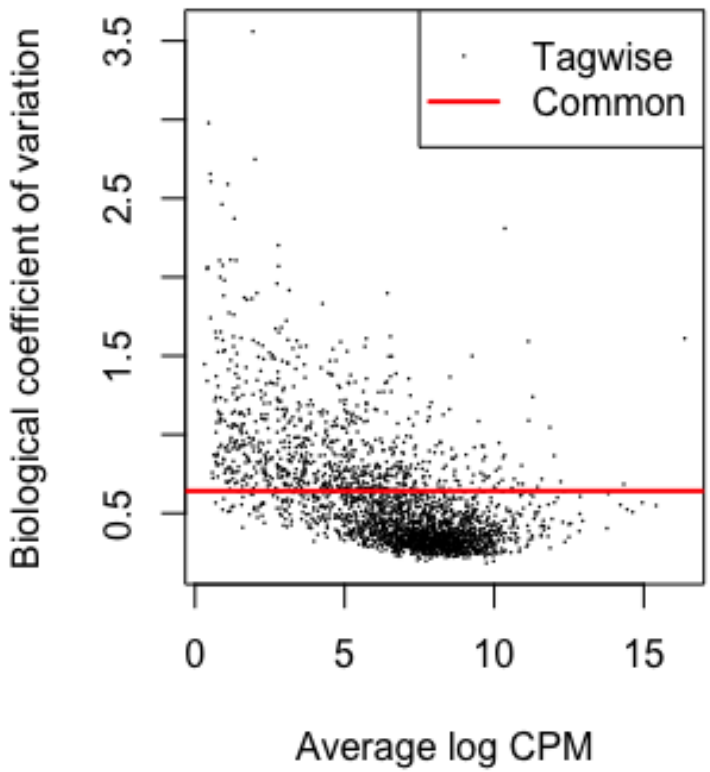
- The square root of the common dispersion gives the coefficient of variation of biological variation (BCV).
- The BCV is the relative variability of expression between biological replicates.
- If you estimate dispersion = 0.41, then sqrt(dispersion) = BCV = 0.64.
    - Means that the expression values vary up and down by 64% between replicates.

Tagwise dispersion

- The way edgeR estimates a tagwise (i.e. gene−wise) dispersion parameter is by "shrinking" the gene−wise dispersions toward a common value (the common dispersion estimated in the previous step).
- Alternatively, one can shrink the gene−wise estimates to a common trend, by estimating a smooth function prior to the shrinkage (using the **estimateTrendedDisp()** function)

Tagwise dispersion

```
> cds <- estimateTagwiseDisp(cds)
> plotBCV(cds)
```



The gene-wise dispersions show a decreasing trend with expression level.

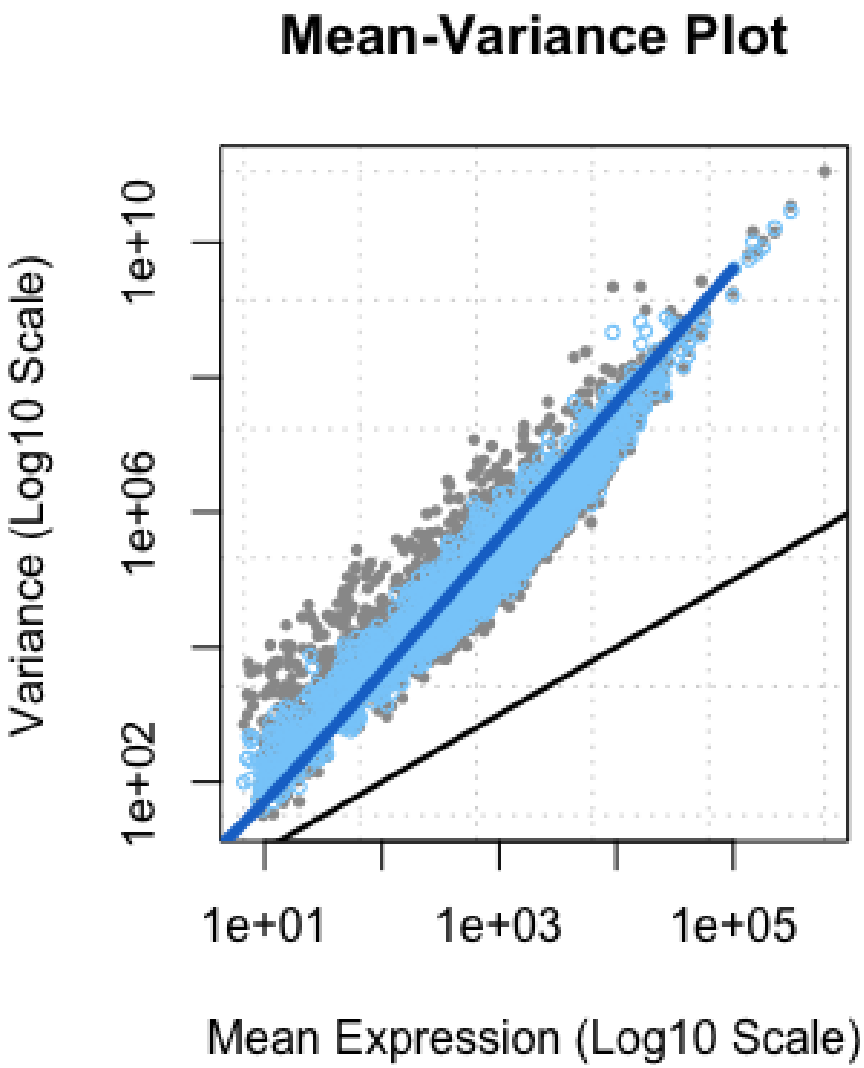
Mean variance plot

Assess how well the estimated dispersion parameters fit the data by plotting the mean-variance relationship.

```
meanVarPlot <- plotMeanVar(cdsG ,show.raw.vars=TRUE, show.tagwise.vars=TRUE
    , show.binned.common.disp.vars=FALSE, show.ave.raw.vars=FALSE, NBline =
    TRUE, nbins = 100, pch = 16,xlab ="Mean Expression (Log10 Scale)", ylab = "
    Variance (Log10 Scale)", main = "Mean-Variance Plot")
```

The plot function outputs the variances which will be stored in the data set *meanVarPlot*

Mean variance plot



- raw variances of the counts (grey dots)
- variances using the tagwise dispersions (light blue dots)
- variances using the common dispersion (solid blue line)
- variance = mean poisson variance (solid black line)

Differential expression analysis

- The function **exactTest** performs pair-wise tests for differential expression between two groups. The important parameter is pair which indicates which two groups should be compared.

```
> et <- exactTest(cds, pair = c("normal", "tumor"))
```

- We need to provide gruop of the samples first. For instance,
  - normal versus tumor
  - control vs treated

Differential expression analysis

```
> class (et)
[1] "DGEExact"
attr(,"package")
[1] "edgeR"
> names (et)
[1] "table" "comparison" "genes"
```

Differential expression analysis

```
> et$comparison
[1] "normal" "tumor"
> head(et$table)
       logFC   logCPM       PValue
1 -1.2620601 5.101380 1.928026e-12
2  0.3746115 5.796729 3.320407e-02
3  0.8842329 6.350847 5.712832e-07
4  0.5703152 4.426511 1.246751e-03
6  0.5075784 6.849094 3.877792e-03
7 -1.0192441 4.095199 1.183635e-08
```

Table of the Top Differentially Expressed Tags

- Extracts the top DE tags in a data frame for a given pair of groups, ranked by p-value or absolute log-fold change.

topTags(object, n=10, adjust.method="BH", sort.by="PValue", p.value=1)

```
object:        a DGEExact object (output from exactTest)
n:             scalar, number of tags to display/return
adjust.method: character string stating the method used to adjust p-values
               for multiple testing, passed on to p.adjust
sort.by:        character string, should the top tags be sorted by p-value
               ("PValue"), by absolute log-fold change ("logFC"),
               or not sorted ("none").
p.value:        cutoff value for adjusted p-values.
```

Table of the Top Differentially Expressed Tags

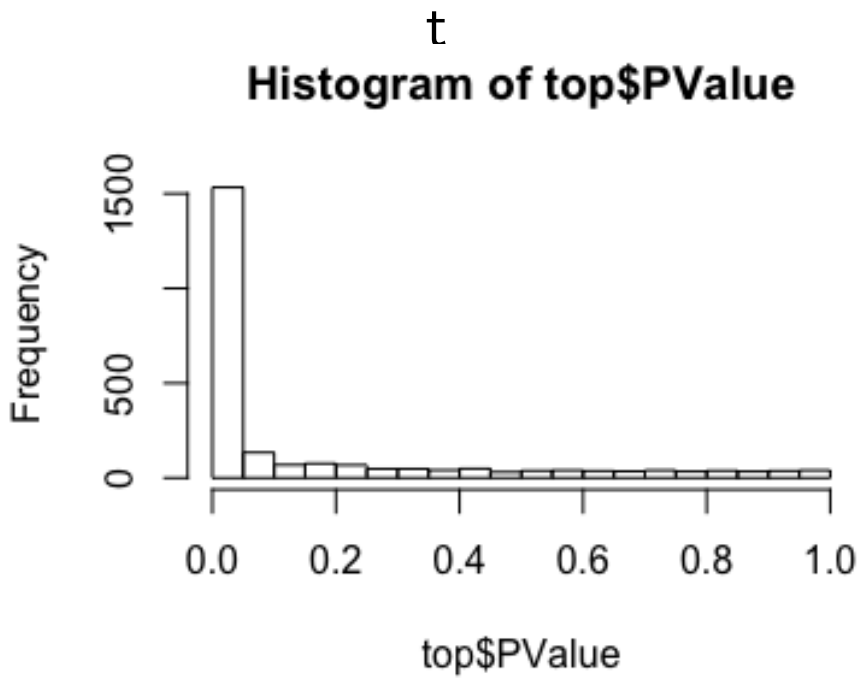
```
> topTags(et)
Comparison of groups:  tumor-normal
        logFC    logCPM        PValue           FDR
772  9.467071 10.356943 4.704334e-308 1.161971e-304
1321 8.114549  6.427723 2.244859e-241 2.772401e-238
2271 6.769685  4.713430 9.260237e-184 7.624262e-181
1336 6.327003  5.723981 4.597782e-173 2.839130e-170
2041 6.105740  8.197687 1.034280e-171 5.109343e-169
2994 6.917303  3.148385 1.590759e-161 6.548625e-159
2551 6.892310  2.791990 3.354682e-154 1.183724e-151
1314 8.772051  1.933639 2.103799e-152 6.495479e-150
1263 5.860897  4.261415 8.651641e-149 2.374395e-146
63   5.216740  7.828899 8.244542e-137 1.879656e-134
```

Interpreting results

- logFC: positive if gene expresses higher in tumor
  - log2(Fold Change) $= \text{log2}(\frac{\text{expression in tumor}}{\text{expression in normal}})$
- logCPM: log2(Counts Per Million)
- FDR: False Discovrty Rate

Extract Differentially Expressed (DE) Genes

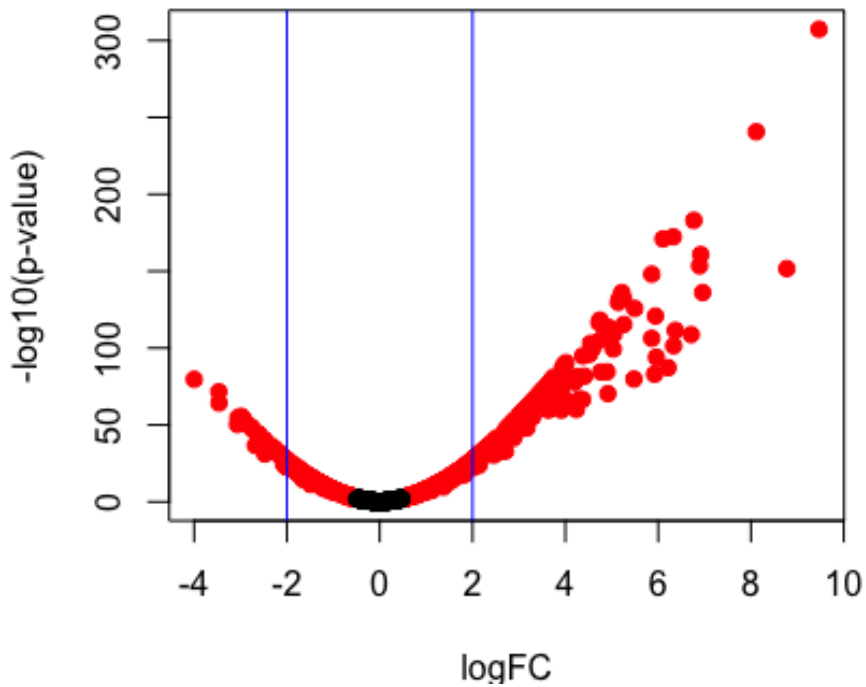
```
> top <- topTags(et, n=nrow(cds$counts))$table
> class (top)
[1] "data.frame"

#Store the gene symbol of the differentially expressed DE) genes
> de <- rownames(top[top$PValue<0.01,])

# generate the distribution of Pvalue
> hist(top$PValue, breaks=20)
```



Extract Differentially Expressed (DE) Genes

- We can use the "volcano plot" to visualize the relationship between log-fold-changes and p-values.

```
> plot(top$logFC,-log10(top$PValue), pch=20, cex=1.5, ylab="-log10(p-value)", xlab="
    logFC", col=as.numeric(rownames(top) %in% de)+1)
> abline(v=c(-2, 2), col="blue")
```



Save the DE results

```
> write.csv(top, file="DEGs_lung_cancer.csv")

> de_sig = top[top$PValue<0.01,]

> write.csv(de_sig, file="DEGs_lung_cancer_significant.csv")
```

Summary: edgeR

Main steps

- Building the edgeR object
- Normalization
- Estimating Dispersion
- Testing for Differentially Expressed (DE)