Multivariate Analysis

04/12/2022 (Week 13)

Jingjing Yang, PhD

Assistant Professor of Human Genetics

Jingjing.yang@emory.edu

Outline

- Data preparation and visualization
- Multiple testing
- Differential gene expression analysis using edgeR

Multivariate Data is Common in Biomedical Researches

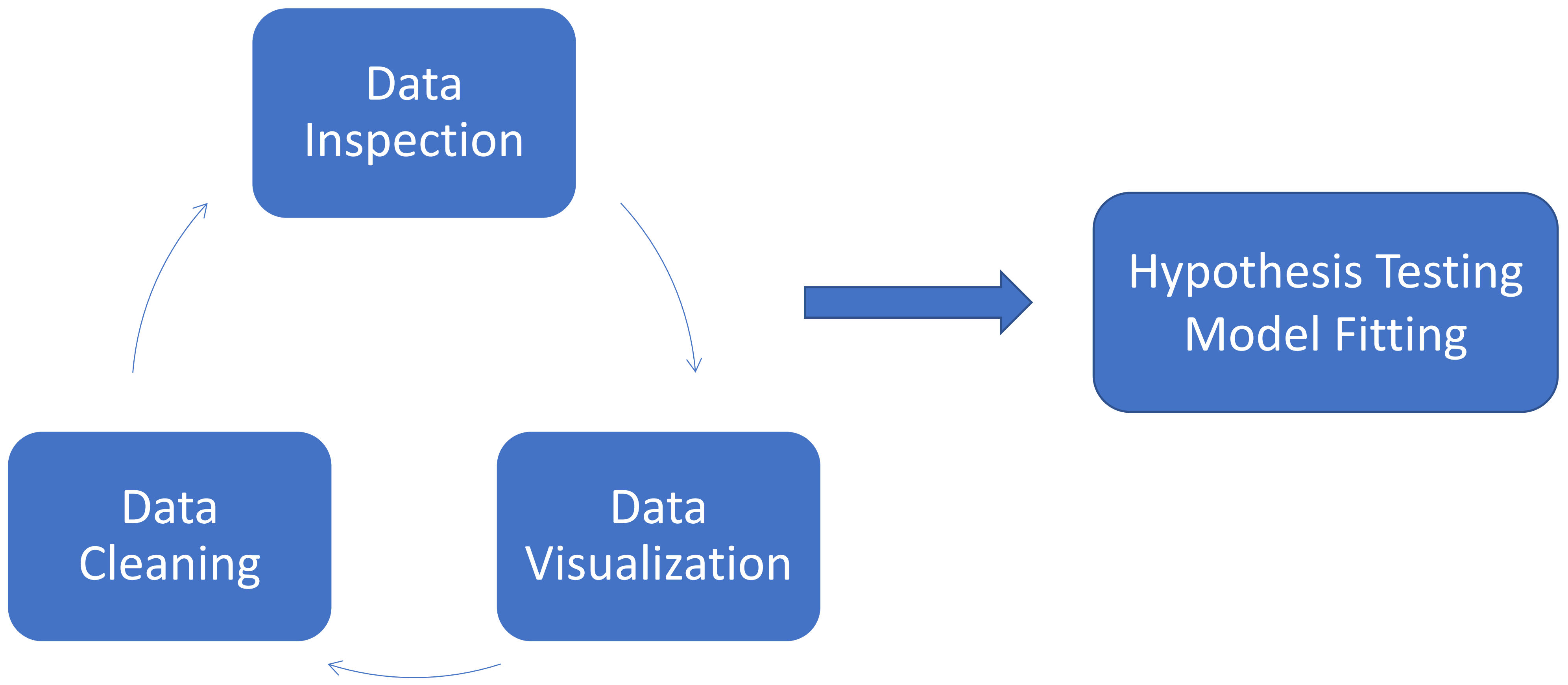
- Multiple variables are generally collected in biomedical projects
  - Demographic variables such as Age, Gender, BMI, Height, etc.
  - Multiple traits to characterize animal behaviors
  - Multiple genes
- Hypothesis test to identify significant association
  - Multiple testing
  - Account for confounding covariates
- Differential gene expression analysis

Visualize Multivariate/High-dimensional Data

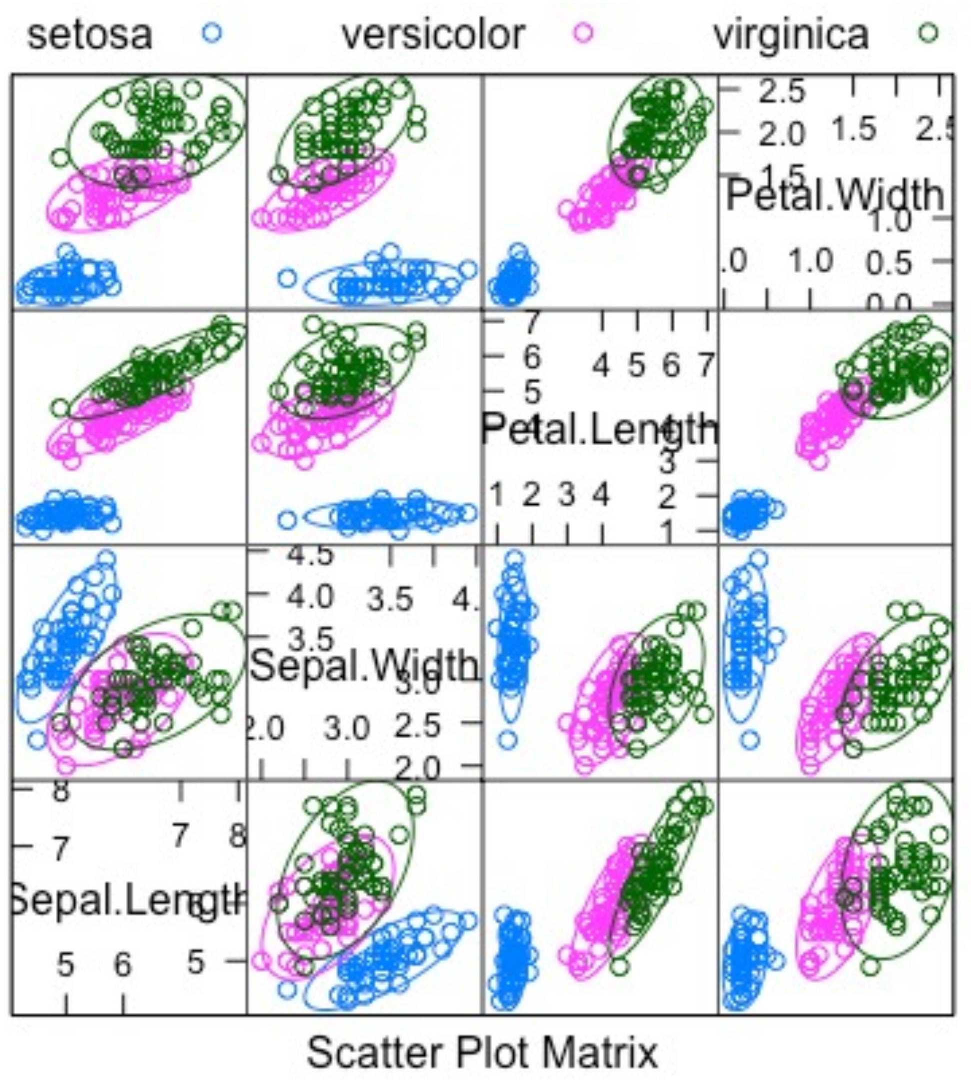
- Learned skills
  - Histogram
  - Scatter plot
  - Heatmap

- Skills to be introduced in this lecture
  - Principal component analysis
  - UMAP

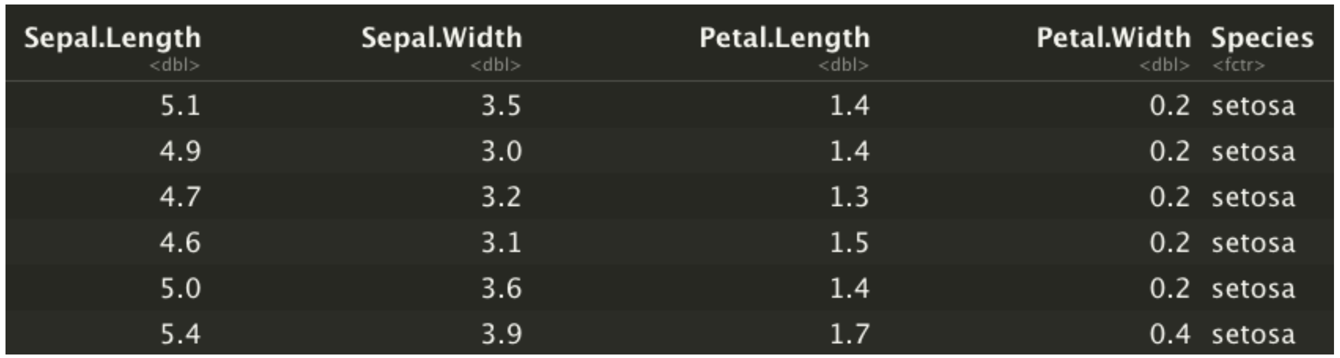
Data Analysis Guidelines



Example of Multivariate/High-dimensional Data



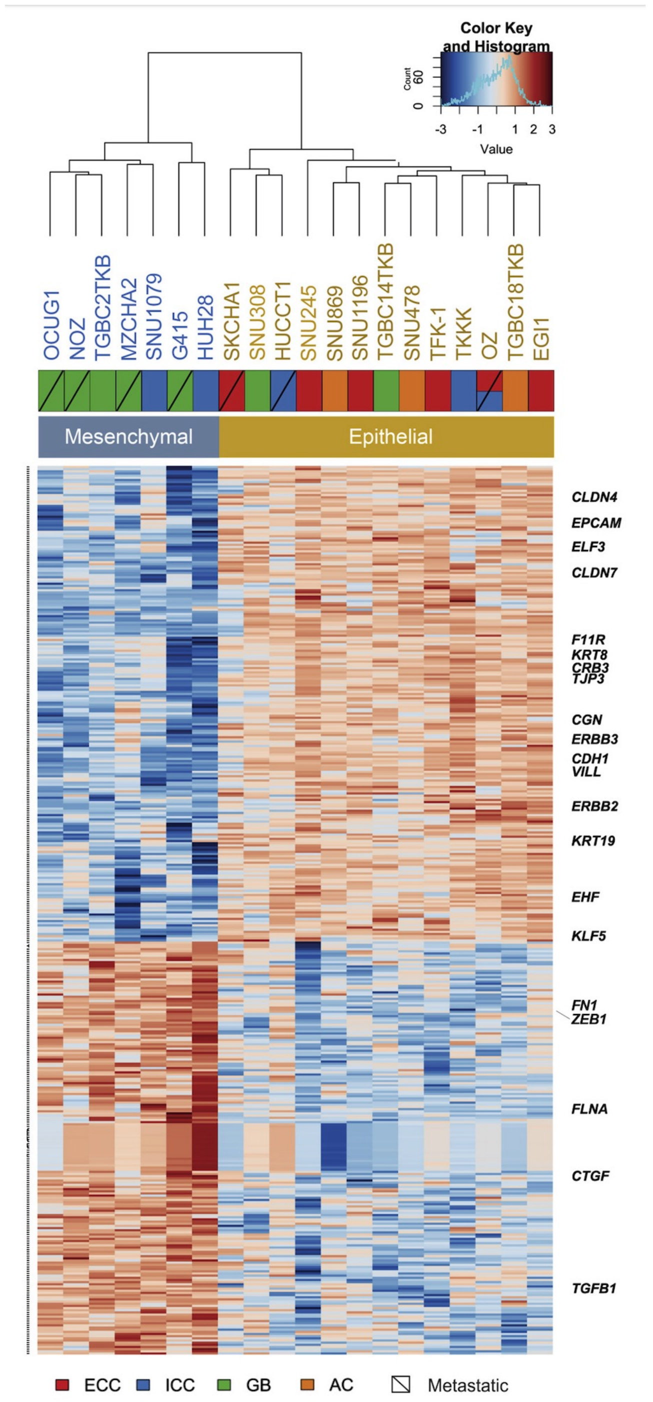
Famous “iris” data set



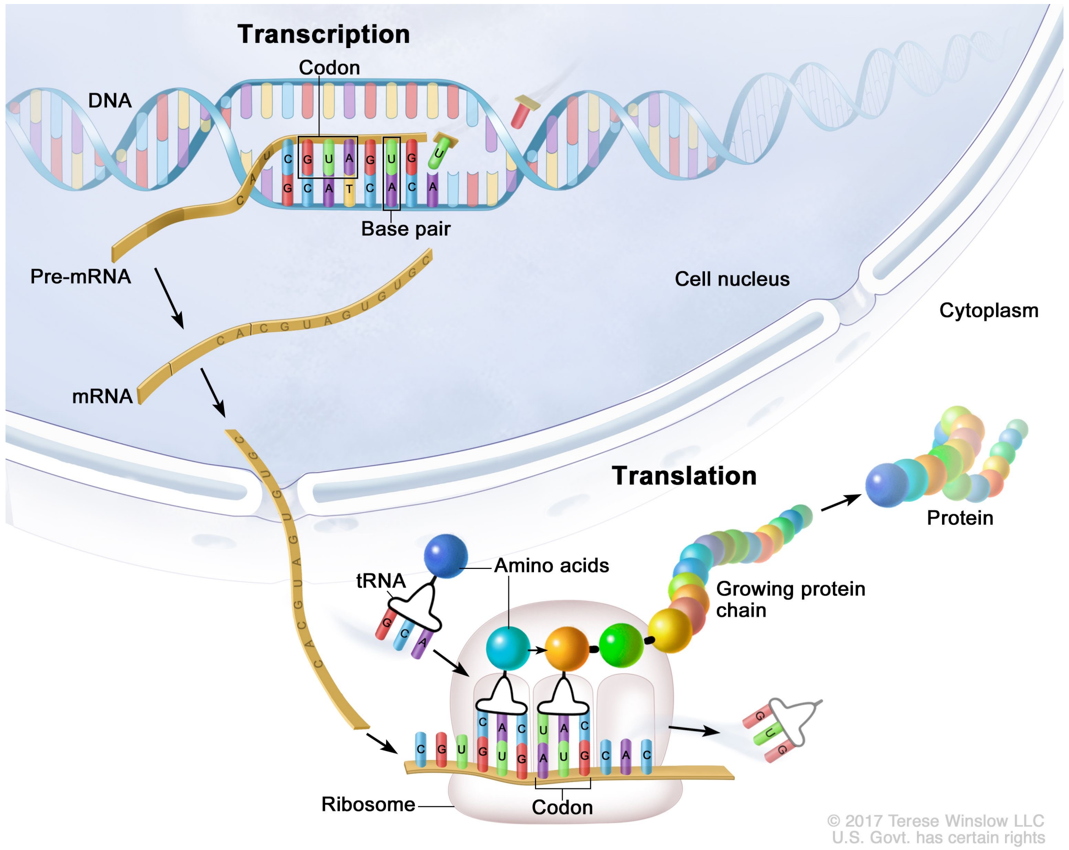
Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fctr>
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

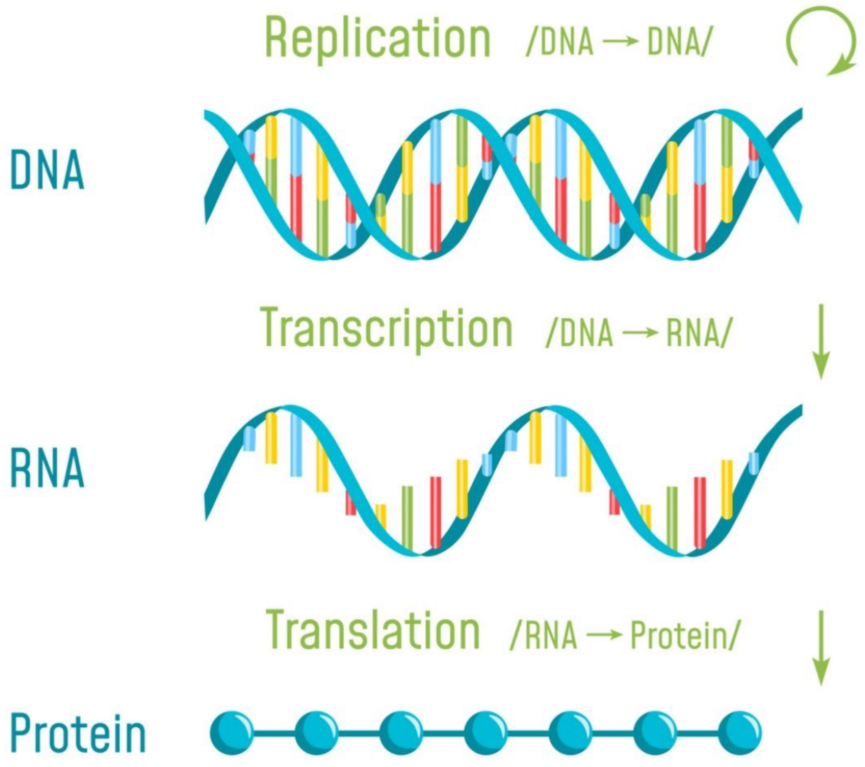
Example of Multivariate/High-dimensional Data

RNA Sequencing data :
Gene expression levels for ~20K human genes



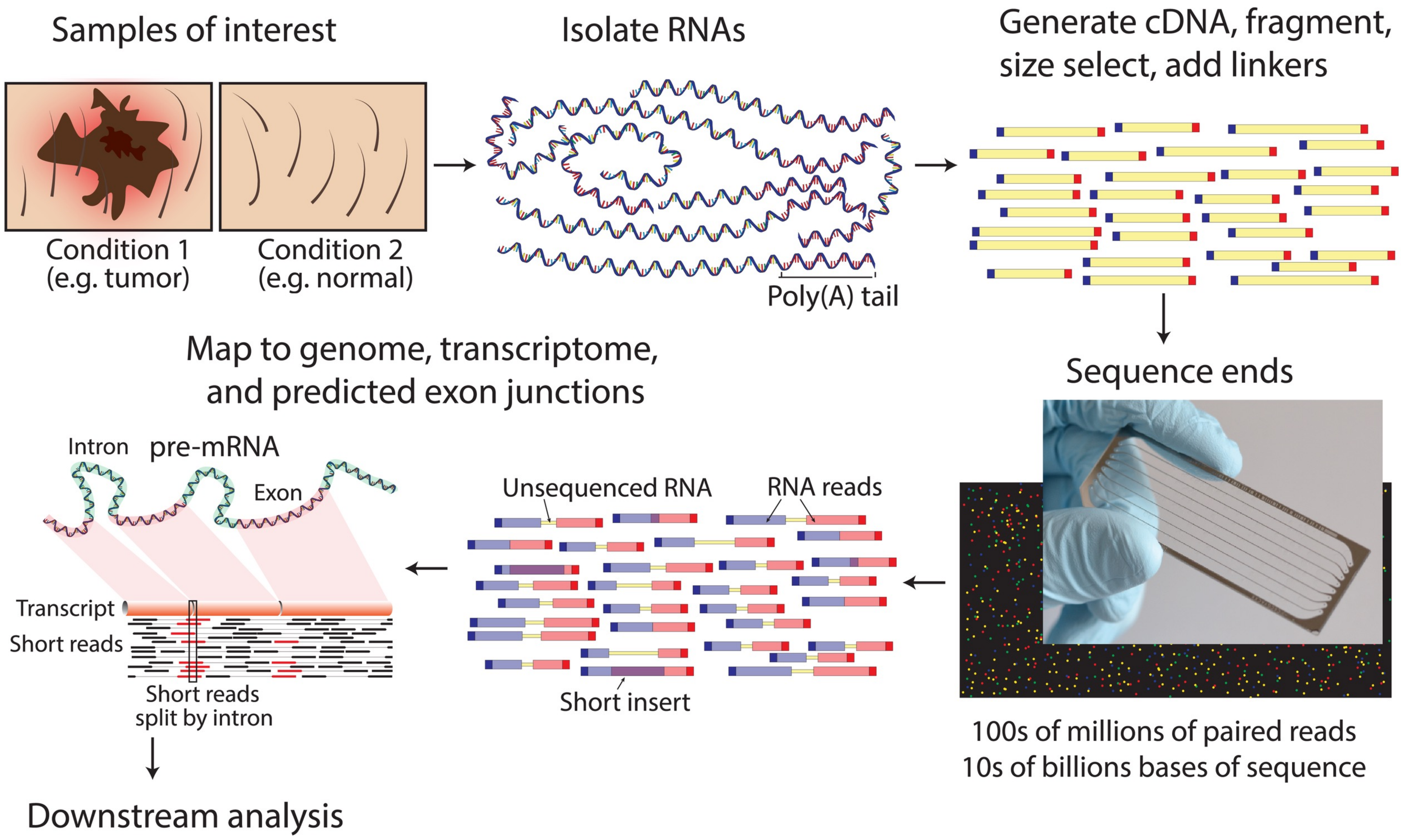
Transcription





https://www.thoughtco.com/dna-transcription-373398

Profile Gene Expression Levels by RNA-sequencing


Samples of interest
Condition 1 (e.g. tumor)
Condition 2 (e.g. normal)
Isolate RNAs
Poly(A) tail
Generate cDNA, fragment, size select, add linkers
Sequence ends
100s of millions of paired reads
10s of billions bases of sequence
Unsequenced RNA
RNA reads
Short insert
Map to genome, transcriptome, and predicted exon junctions
Intron
pre-mRNA
Exon
Transcript
Short reads
Short reads split by intron
Downstream analysis

RNA-Seq Data

- Gene expression Quantitative Traits
  - Profiled by RNA sequencing (RNA-seq)
  - CPM (Counts Per Million) per gene
    - Count up all the read counts in a sample (library size) and divide this number by 1,000,000. This is your “per million” scaling factor.
    - Divide the read counts per gene by the “per million” scaling factor. This gives you CPM.

- 20K ~ 25K genes in human genome

- Bulk RNA-seq ; Single Cell RNA-seq

Example RNA-Seq Data

- Example RNA-Seq data from: David K. Lau et. al., 2019. Genomic Profiling of Biliary Tract Cancer Cell Lines Reveals Molecular Subtypes and Actionable Drug Targets. PMID: 31731200 ; DOI: 10.1016/j.isci.2019.10.044

- Samples from 20 biliary track cancer cell lines were profiled for gene expression data by RNA sequencing

- 24222 genes in the raw data

Inspecting Raw RNA-Seq Data

```r
head(RNAseq_dt)
```

```
##      RefSeq EGI1 G415 HUCCT1 HUH28 MZCHA2  NOZ OCUG1   OZ SKCHA1 SNU1079
## 1 NM_000014    1    0     10    25   8935    1     0    1      3       0
## 2 NM_000015    4    0      1     0      5    0     0    0      2       3
## 3 NM_000017  120  154    132    46    240  163    47  188    195     293
## 4 NM_000019  444 1246    467   426    350 1245   470  286    783     843
## 5 NM_000020    0  250      1     0     85    4     0    0      0      39
## 6 NM_000021 2373 1989   2648   796   1083  958   933 1539   2454    1555
##   SNU1196 SNU245 SNU308 SNU478 SNU869 TFK1 TGBC14TKB TGBC18TKB TGBC2TKB TKKK
## 1       6      0      0     58      2    0         3         1      185    0
## 2       1      2      1      1      0    2         4         1        2    5
## 3     411    497    211    160    161  369       354       212       99  344
## 4     418    175    586    854    712  740       751       604      451 1034
## 5       2      0      2      2      5    4        27         1        0    0
## 6    1618   1539   1421   1282   1525 2151      1158      1590      804 1339
```

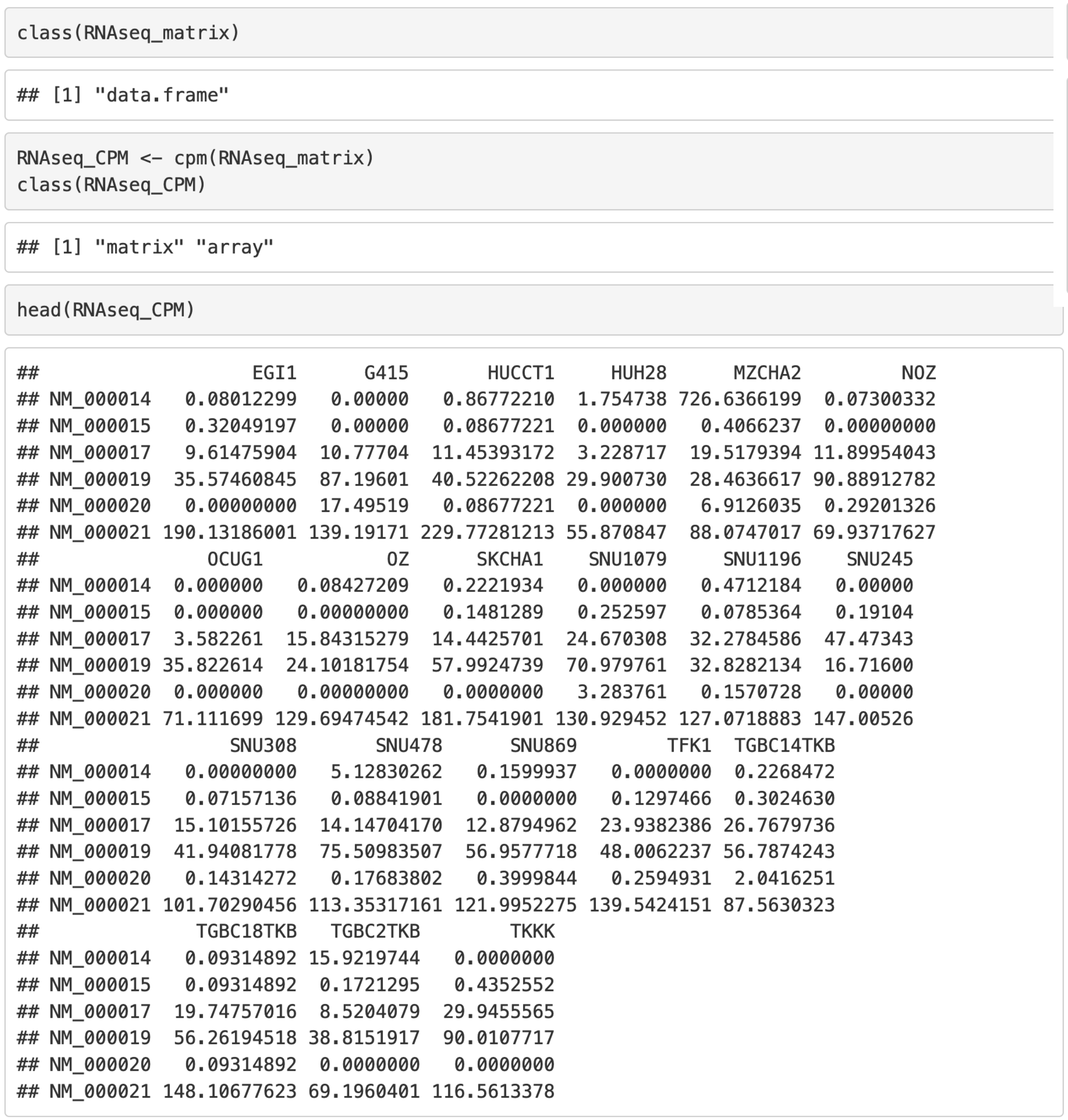
Normalize Raw RNA-Seq Data

```
apply(RNAseq_matrix, 2, sum)
```

```
##      EGI1     G415   HUCCT1    HUH28   MZCHA2      NOZ    OCUG1       OZ
## 12480812 14289644 11524427 14247144 12296380 13698008 13120204 11866325
##    SKCHA1  SNU1079  SNU1196   SNU245   SNU308   SNU478   SNU869     TFK1
## 13501752 11876625 12732950 10469013 13972069 11309785 12500489 15414668
## TGBC14TKB TGBC18TKB  TGBC2TKB     TKKK
## 13224759 10735498 11619162 11487514
```

- Summarizing read counts per column/sample gives us the **library size**. The total number of mapped read counts per sample.
- Various library sizes make the raw read counts per gene are not comparable across all samples/cell-lines.
- Need to **Normalize** read counts to Counts Per Million (CPM)

Get RNA-Seq Data in CPM



```
class(RNAseq_matrix)
```

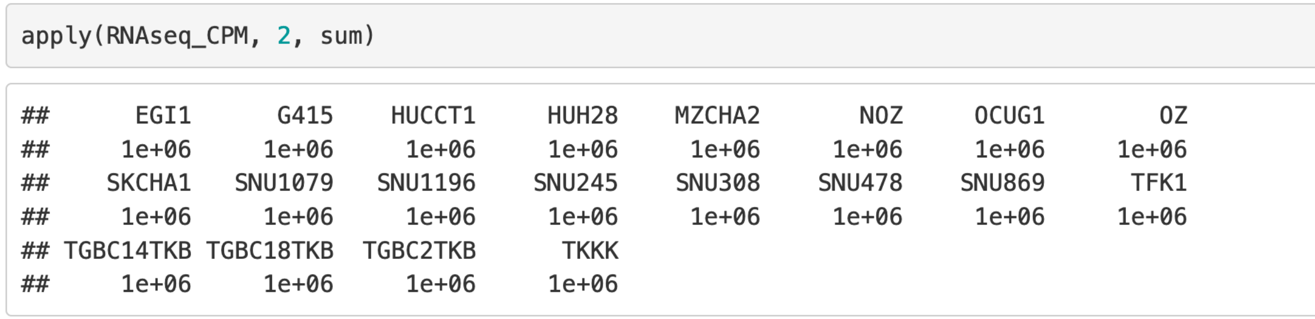
```
## [1] "data.frame"
```

```
RNAseq_CPM <- cpm(RNAseq_matrix)
class(RNAseq_CPM)
```

```
## [1] "matrix" "array"
```

```
head(RNAseq_CPM)
```

```
##                  EGI1      G415       HUCCT1     HUH28      MZCHA2         NOZ
## NM_000014   0.08012299   0.00000   0.86772210  1.754738 726.6366199  0.07300332
## NM_000015   0.32049197   0.00000   0.08677221  0.000000   0.4066237  0.00000000
## NM_000017   9.61475904  10.77704  11.45393172  3.228717  19.5179394 11.89954043
## NM_000019  35.57460845  87.19601  40.52262208 29.900730  28.4636617 90.88912782
## NM_000020   0.00000000  17.49519   0.08677221  0.000000   6.9126035  0.29201326
## NM_000021 190.13186001 139.19171 229.77281213 55.870847  88.0747017 69.93717627
##              OCUG1           OZ      SKCHA1    SNU1079     SNU1196    SNU245
## NM_000014  0.000000   0.08427209   0.2221934   0.000000   0.4712184   0.00000
## NM_000015  0.000000   0.00000000   0.1481289   0.252597   0.0785364   0.19104
## NM_000017  3.582261  15.84315279  14.4425701  24.670308  32.2784586  47.47343
## NM_000019 35.822614  24.10181754  57.9924739  70.979761  32.8282134  16.71600
## NM_000020  0.000000   0.00000000   0.0000000   3.283761   0.1570728   0.00000
## NM_000021 71.111699 129.69474542 181.7541901 130.929452 127.0718883 147.00526
##                 SNU308       SNU478      SNU869        TFK1  TGBC14TKB
## NM_000014   0.00000000   5.12830262   0.1599937   0.0000000  0.2268472
## NM_000015   0.07157136   0.08841901   0.0000000   0.1297466  0.3024630
## NM_000017  15.10155726  14.14704170  12.8794962  23.9382386 26.7679736
## NM_000019  41.94081778  75.50983507  56.9577718  48.0062237 56.7874243
## NM_000020   0.14314272   0.17683802   0.3999844   0.2594931  2.0416251
## NM_000021 101.70290456 113.35317161 121.9952275 139.5424151 87.5630323
##              TGBC18TKB   TGBC2TKB        TKKK
## NM_000014   0.09314892 15.9219744   0.0000000
## NM_000015   0.09314892  0.1721295   0.4352552
## NM_000017  19.74757016  8.5204079  29.9455565
## NM_000019  56.26194518 38.8151917  90.0107717
## NM_000020   0.09314892  0.0000000   0.0000000
## NM_000021 148.10677623 69.1960401 116.5613378
```

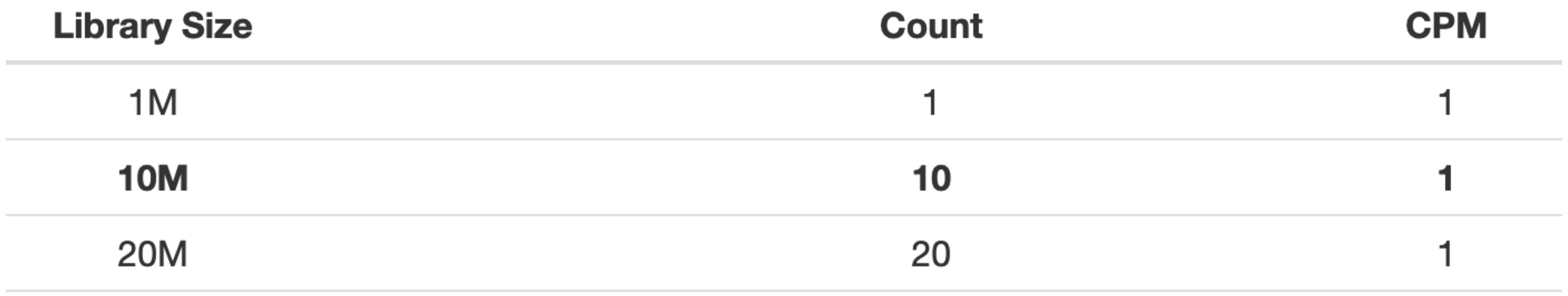


```
apply(RNAseq_CPM, 2, sum)
```

```
##      EGI1      G415    HUCCT1     HUH28    MZCHA2       NOZ     OCUG1        OZ
##     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06
##    SKCHA1   SNU1079   SNU1196    SNU245    SNU308    SNU478    SNU869      TFK1
##     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06
## TGBC14TKB TGBC18TKB  TGBC2TKB      TKKK
##     1e+06     1e+06     1e+06     1e+06
```

Data Cleaning : filtering out genes with low CPM

- Low read counts are more likely to add noises.
- As a general rule, a good threshold can be chosen for a CPM value that corresponds to 10 raw read counts.



Library Size	Count	CPM
1M	1	1
10M	**10**	**1**
20M	20	1

Data Cleaning : filtering out genes with low CPM

```r
thresh <- RNAseq_CPM > 1
class(thresh)
```

```
## [1] "matrix" "array"
```

```r
head(thresh)
```

```
##            EGI1  G415 HUCCT1 HUH28 MZCHA2   NOZ OCUG1    OZ SKCHA1 SNU1079
## NM_000014 FALSE FALSE  FALSE  TRUE   TRUE FALSE FALSE FALSE  FALSE   FALSE
## NM_000015 FALSE FALSE  FALSE FALSE  FALSE FALSE FALSE FALSE  FALSE   FALSE
## NM_000017  TRUE  TRUE   TRUE  TRUE   TRUE  TRUE  TRUE  TRUE   TRUE    TRUE
## NM_000019  TRUE  TRUE   TRUE  TRUE   TRUE  TRUE  TRUE  TRUE   TRUE    TRUE
## NM_000020 FALSE  TRUE  FALSE FALSE   TRUE FALSE FALSE FALSE  FALSE    TRUE
## NM_000021  TRUE  TRUE   TRUE  TRUE   TRUE  TRUE  TRUE  TRUE   TRUE    TRUE
##           SNU1196 SNU245 SNU308 SNU478 SNU869  TFK1 TGBC14TKB TGBC18TKB
## NM_000014   FALSE  FALSE  FALSE   TRUE  FALSE FALSE     FALSE     FALSE
## NM_000015   FALSE  FALSE  FALSE  FALSE  FALSE FALSE     FALSE     FALSE
## NM_000017    TRUE   TRUE   TRUE   TRUE   TRUE  TRUE      TRUE      TRUE
## NM_000019    TRUE   TRUE   TRUE   TRUE   TRUE  TRUE      TRUE      TRUE
## NM_000020   FALSE  FALSE  FALSE  FALSE  FALSE FALSE      TRUE     FALSE
## NM_000021    TRUE   TRUE   TRUE   TRUE   TRUE  TRUE      TRUE      TRUE
##           TGBC2TKB  TKKK
## NM_000014     TRUE FALSE
## NM_000015    FALSE FALSE
## NM_000017     TRUE  TRUE
## NM_000019     TRUE  TRUE
## NM_000020    FALSE FALSE
## NM_000021     TRUE  TRUE
```

```r
RNAseq_CPM.keep <- RNAseq_CPM[keep,]
class(RNAseq_CPM.keep)
```

```
## [1] "matrix" "array"
```

```r
dim(RNAseq_CPM.keep)
```

```
## [1] 10034    20
```

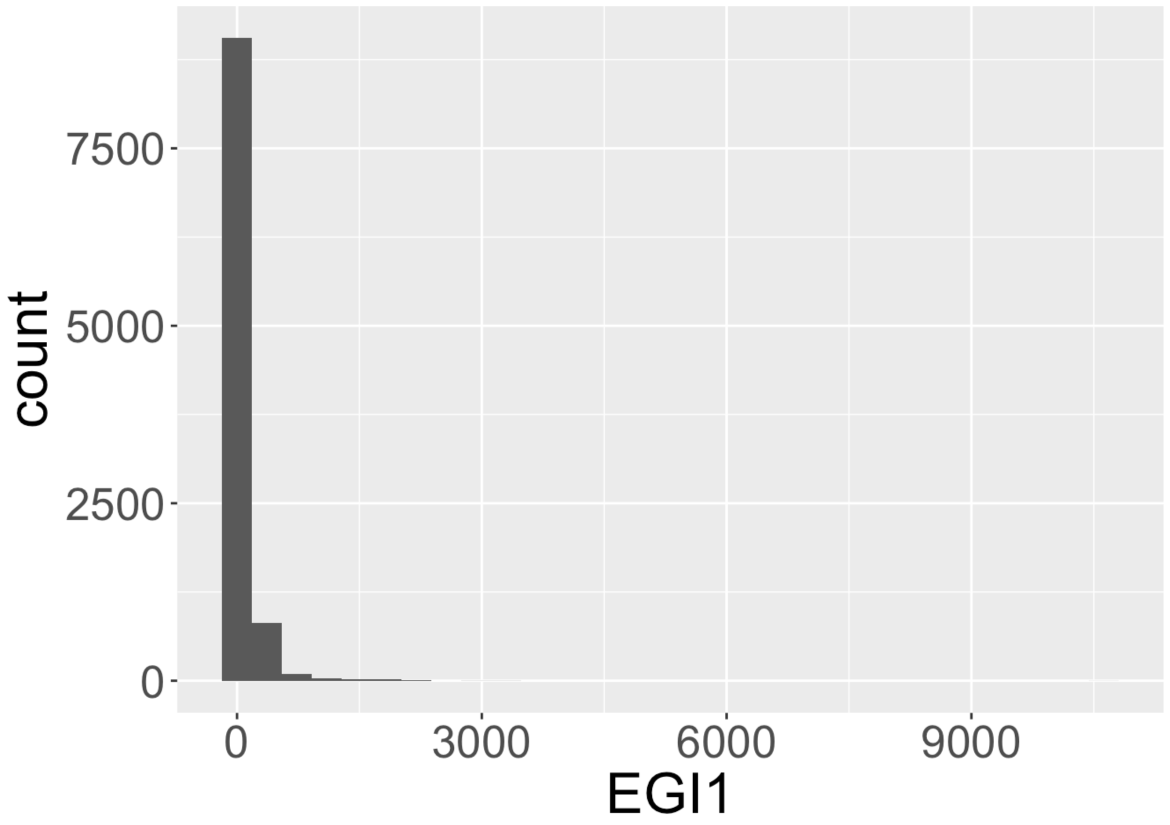
```r
head(RNAseq_CPM.keep)
```

```
##                 EGI1       G415    HUCCT1     HUH28    MZCHA2      NOZ
## NM_000017   9.614759  10.777035  11.45393  3.228717  19.51794 11.89954
## NM_000019  35.574608  87.196014  40.52262 29.900730  28.46366 90.88913
## NM_000021 190.131860 139.191711 229.77281 55.870847  88.07470 69.93718
## NM_000026  43.747154  77.958555  52.49719 48.430759  53.18638 89.79408
## NM_000027  17.226443   7.557921  21.17242 17.687756  19.19264 14.67367
## NM_000028  49.836501  52.975428  32.10572 58.116911 114.26127 13.87063
##                OCUG1        OZ    SKCHA1   SNU1079   SNU1196    SNU245    SNU308
## NM_000017  3.582261  15.84315  14.44257  24.67031  32.27846  47.47343  15.10156
## NM_000019 35.822614  24.10182  57.99247  70.97976  32.82821  16.71600  41.94082
## NM_000021 71.111699 129.69475 181.75419 130.92945 127.07189 147.00526 101.70290
## NM_000026 71.797664  48.96208 108.57850  67.44340  78.77200  48.33311  44.37424
## NM_000027 35.060430  29.15814  47.77158  26.01749  13.82241  13.18176  12.31027
## NM_000028 46.035870  35.05719  46.51248  34.35319  26.38823  34.86480 646.71882
##               SNU478    SNU869      TFK1 TGBC14TKB TGBC18TKB  TGBC2TKB      TKKK
## NM_000017  14.14704  12.87950  23.93824  26.76797  19.74757  8.520408  29.94556
## NM_000019  75.50984  56.95777  48.00622  56.78742  56.26195 38.815192  90.01077
## NM_000021 113.35317 121.99523 139.54242  87.56303 148.10678 69.196040 116.56134
## NM_000026 138.19891 108.95574  68.18181  39.62265  44.89778 49.745412  35.69092
## NM_000027  18.56799  14.87942  22.57590  14.14014  59.14956 46.561017  17.58431
## NM_000028  65.87216  39.75844  72.52832  57.24112  28.13097 89.507316  54.58100
```

Data Visualization : Histogram plot per sample

```
ggplot(data.frame(RNAseq_CPM.keep), aes(x = EGI1)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

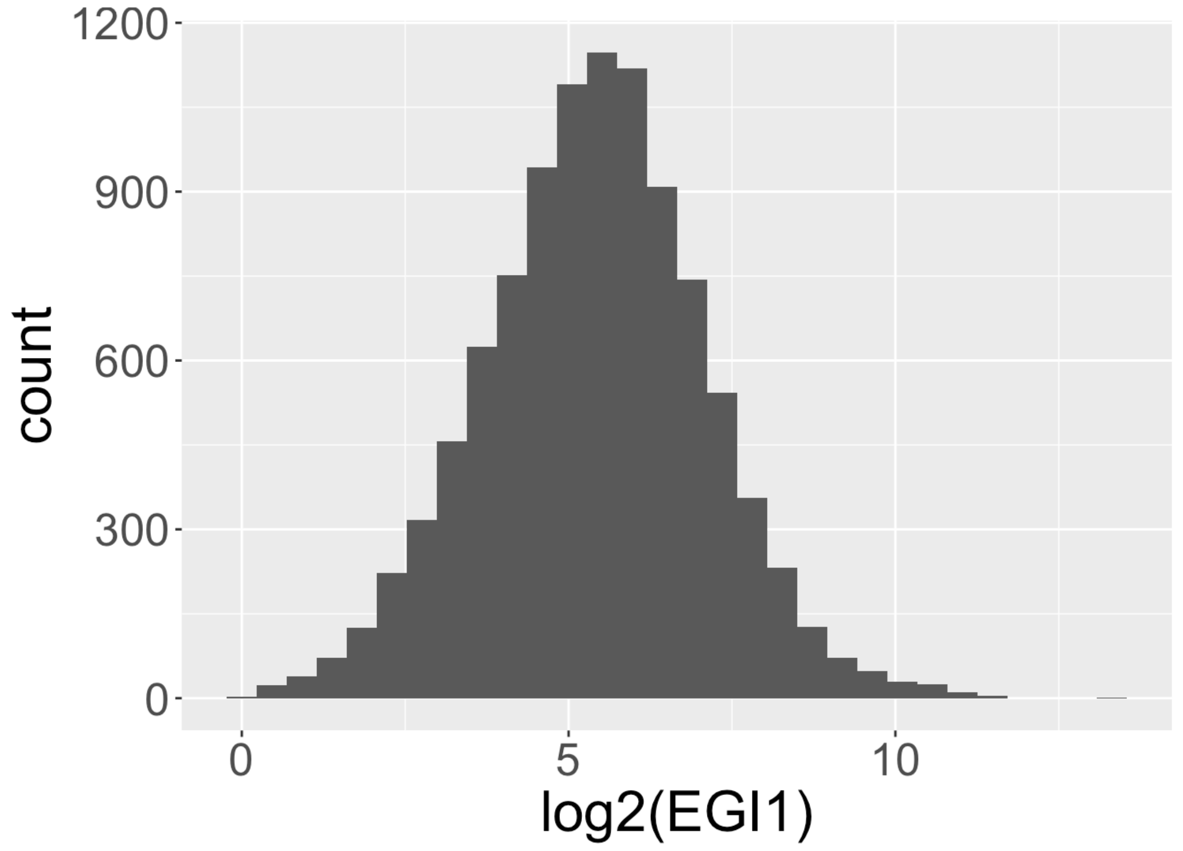


Normally distributed?

Data Transformation: Log2

```
ggplot(data.frame(RNAseq_CPM.keep), aes(x = log2(EGI1) )) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Normally distributed?

Other Data Transformation

- Standardization
  - Center variables to have mean 0
  - Scale variable variances to 1
- Square root
- Nature log
- Log10
- Inverse-normalize

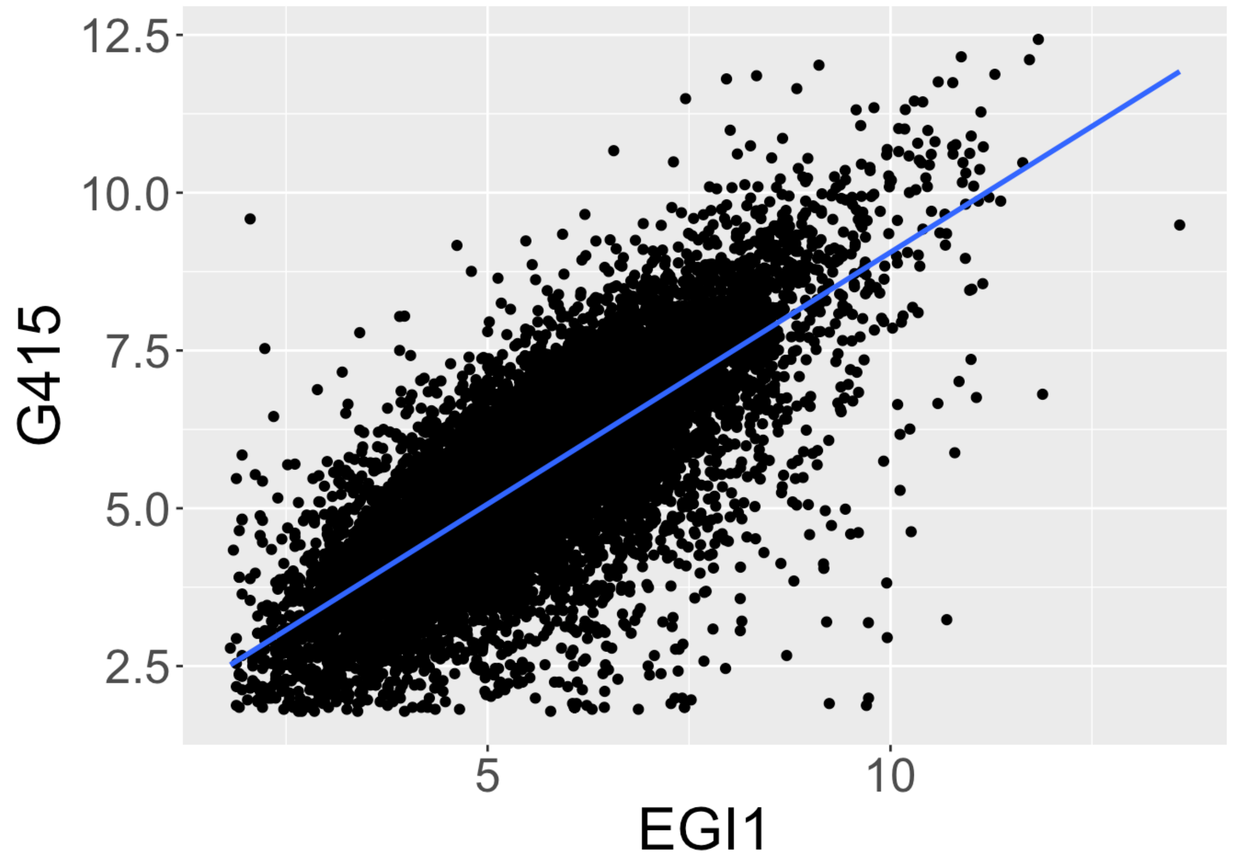
Question

- Why do we like normally distributed data?

Data Visualization: Correlation between cell lines

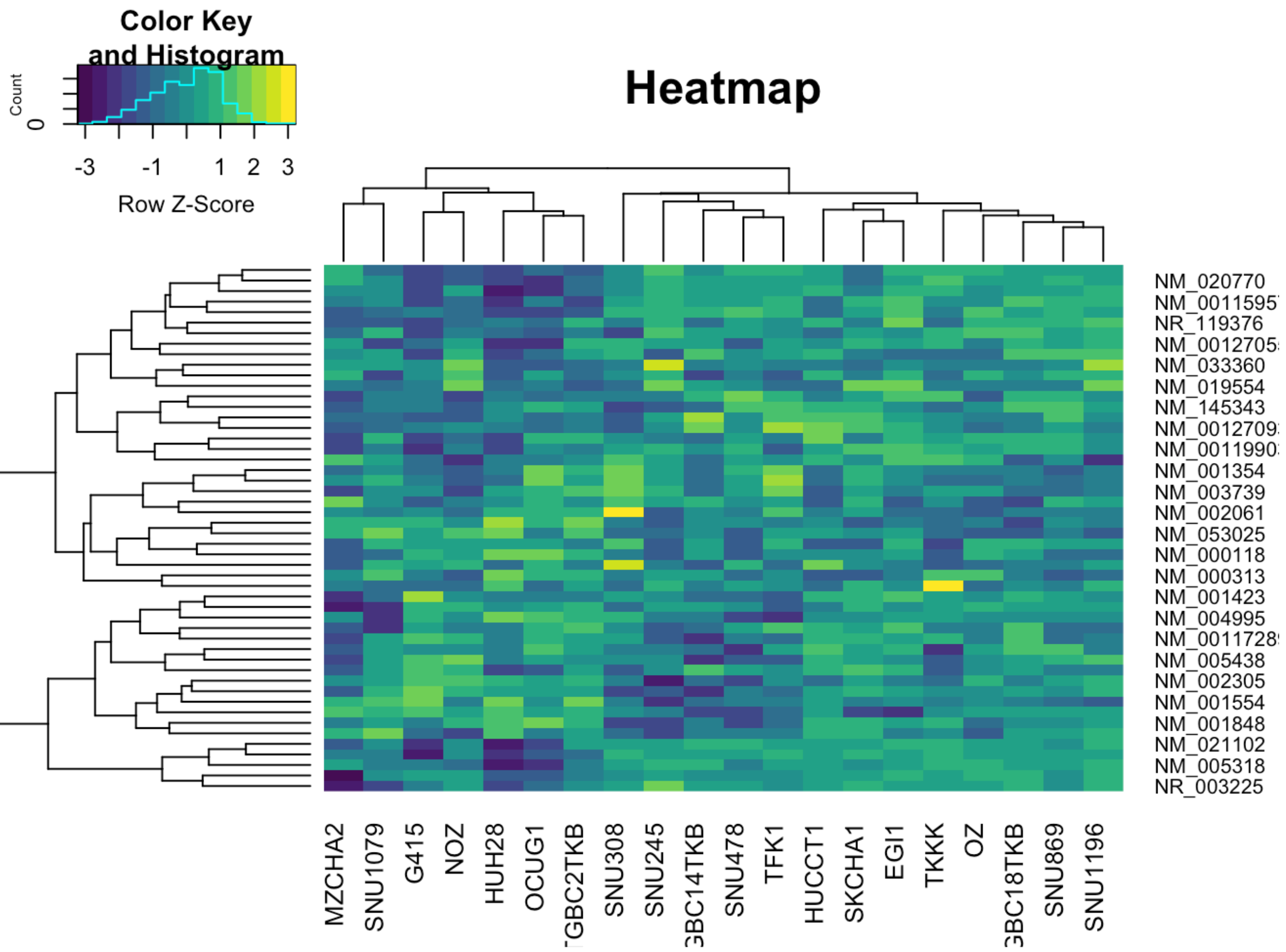
```
ggplot(data.frame(RNAseq_CPM.keep.log2), aes(x = EGI1, y = G415)) + geom_point() +
    geom_smooth(method = "lm", se = FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Data Visualization: Heatmap for highly variable genes

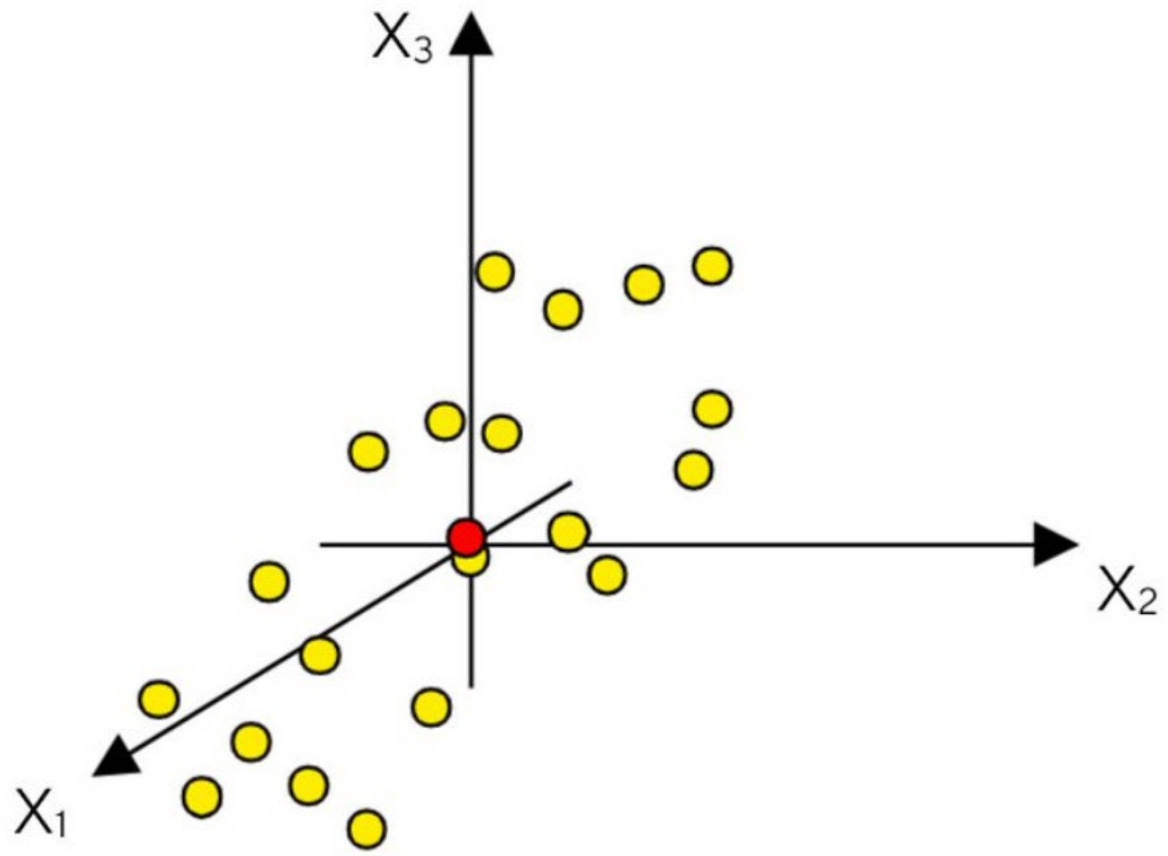
```
hvlcpm <- RNAseq_CPM.keep.log2[select_genes, ]
gplots::heatmap.2(hvlcpm,
          col=viridis,
          trace="none",
          main="Heatmap",
          scale="row")
```

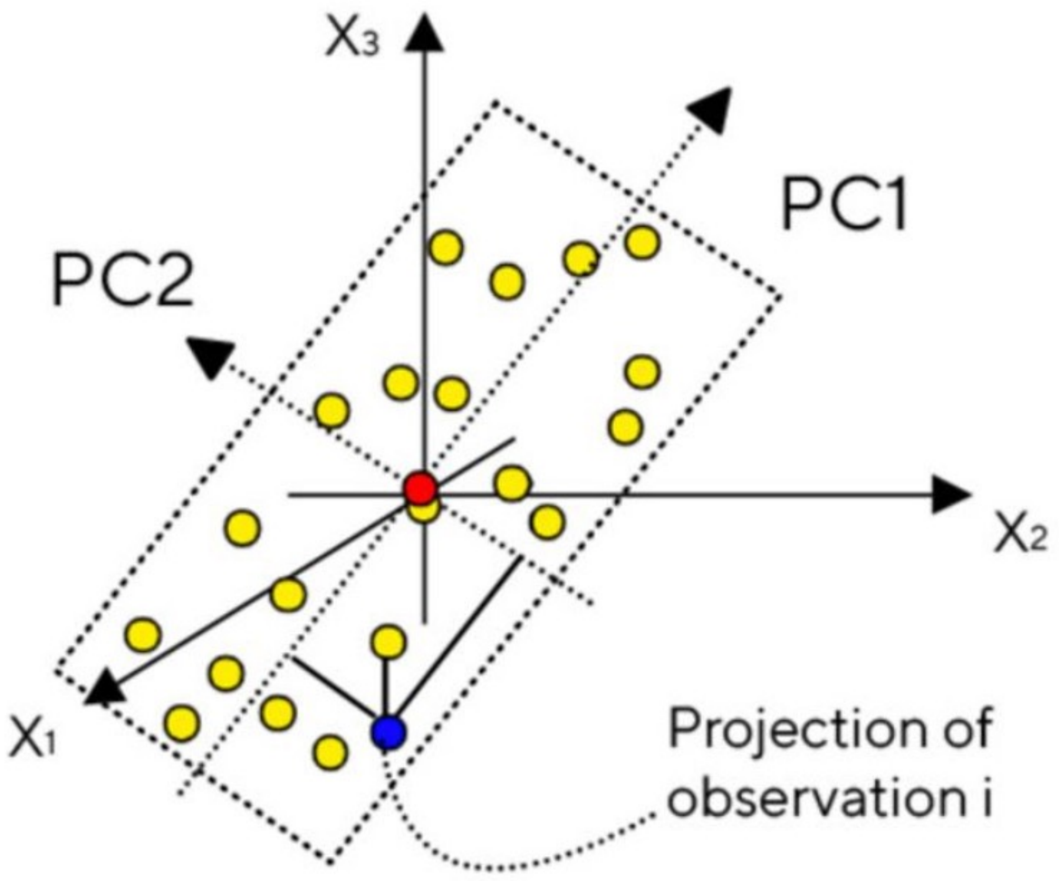


How to visualize sample relationship using all gene expression data?

- 10034 genes left after filtering out low expressed ones
- Still high dimensional data
- Project high dimensional data to two dimensions (dimension reduction), and then a scatter plot will work.
- How?

Principal Component Analysis





Principal Component Analysis

- Find orthogonal loading vectors which explain data variation from the largest to the smallest
- Project original data matrix to loading vectors to obtain Principal Components (PCs)
- Top 2 loading vectors defines a model plane that explain most data variation
- Top 2 PCs can be plotted in a scatter plot to visualize samples

Principal Components Analysis (PCA)

- Consider data matrix $X_{n\times \mathrm{p}}$, with n samples and p variables
- Center and standardize columns in $X_{n\times \mathrm{p}}$ -> $Z_{n\times p}$
- PCA project original genotype data matrix to a new coordinate system such that the PC1 explains the most data variance, and then PC2, …
  - Compute the $n\times n$ variance-covariance matrix for all samples as $\Sigma_{n\times n} = ZZ^T/(n-1)$
  - Conduct eigenvalue decomposition of Σ, by R function eign().
  - Eigen vectors would be loading vectors (w_k, length p, k=1, 2, …) for PC1, PC2, ….
  - Principle components (PCs) are given by: $\mathrm{Z}w_k$

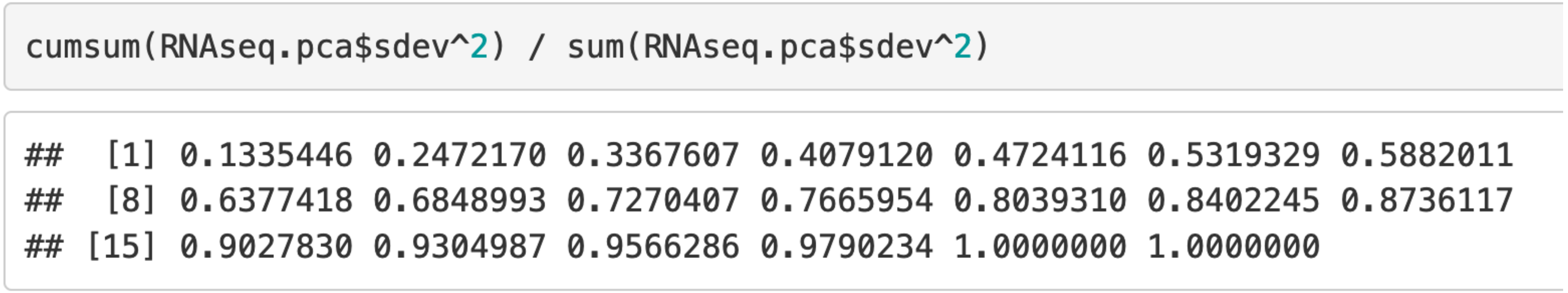
Principal Components Analysis (PCA)

```r
RNAseq.pca <- prcomp(t(RNAseq_CPM.keep.log2), retx = TRUE, scale. = TRUE)
str(RNAseq.pca)
```

```
## List of 5
##  $ sdev    : num [1:20] 36.6 33.8 30 26.7 25.4 ...
##  $ rotation: num [1:10034, 1:20] -0.00983 0.00658 -0.00757 0.01461 -0.00949 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:10034] "NM_000017" "NM_000019" "NM_000021" "NM_000026" ...
##   .. ..$ : chr [1:20] "PC1" "PC2" "PC3" "PC4" ...
##  $ center  : Named num [1:10034] 4.3 5.79 7.08 6.19 4.7 ...
##   ..- attr(*, "names")= chr [1:10034] "NM_000017" "NM_000019" "NM_000021" "NM_000026" ...
##  $ scale   : Named num [1:10034] 0.803 0.619 0.515 0.507 0.687 ...
##   ..- attr(*, "names")= chr [1:10034] "NM_000017" "NM_000019" "NM_000021" "NM_000026" ...
##  $ x       : num [1:20, 1:20] -31.3 58.6 -11.9 14 -17.7 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:20] "EGI1" "G415" "HUCCT1" "HUH28" ...
##   .. ..$ : chr [1:20] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"
```

Principal Components Analysis (PCA)

- Decide the number of top PCs to use in the analysis
  - Select the number of PCs such that a certain percentage of total data variation would be explained, e.g., 95%



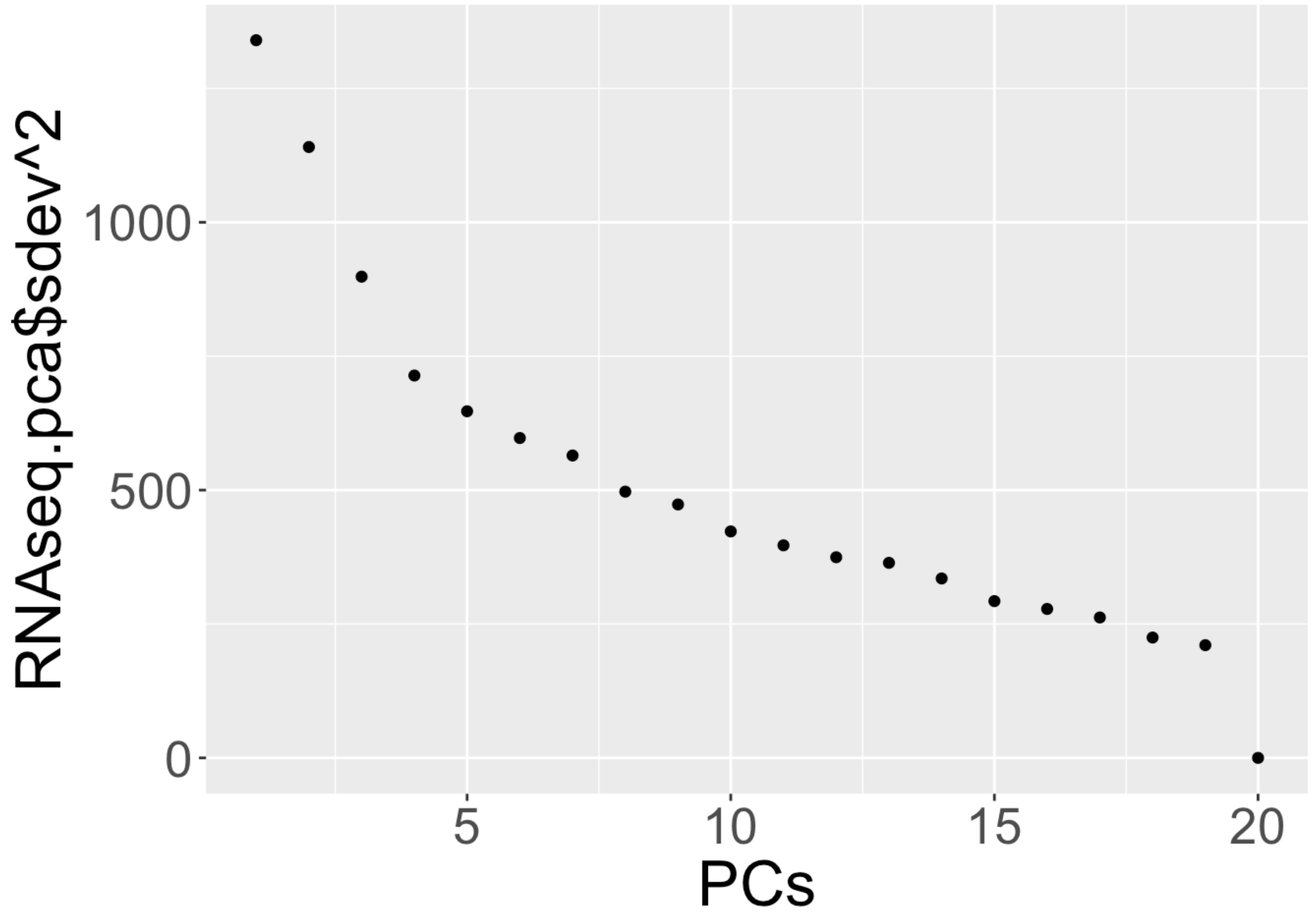
```r
cumsum(RNAseq.pca$sdev^2) / sum(RNAseq.pca$sdev^2)
```

```
##  [1] 0.1335446 0.2472170 0.3367607 0.4079120 0.4724116 0.5319329 0.5882011
##  [8] 0.6377418 0.6848993 0.7270407 0.7665954 0.8039310 0.8402245 0.8736117
## [15] 0.9027830 0.9304987 0.9566286 0.9790234 1.0000000 1.0000000
```

Principal Components Analysis (PCA)

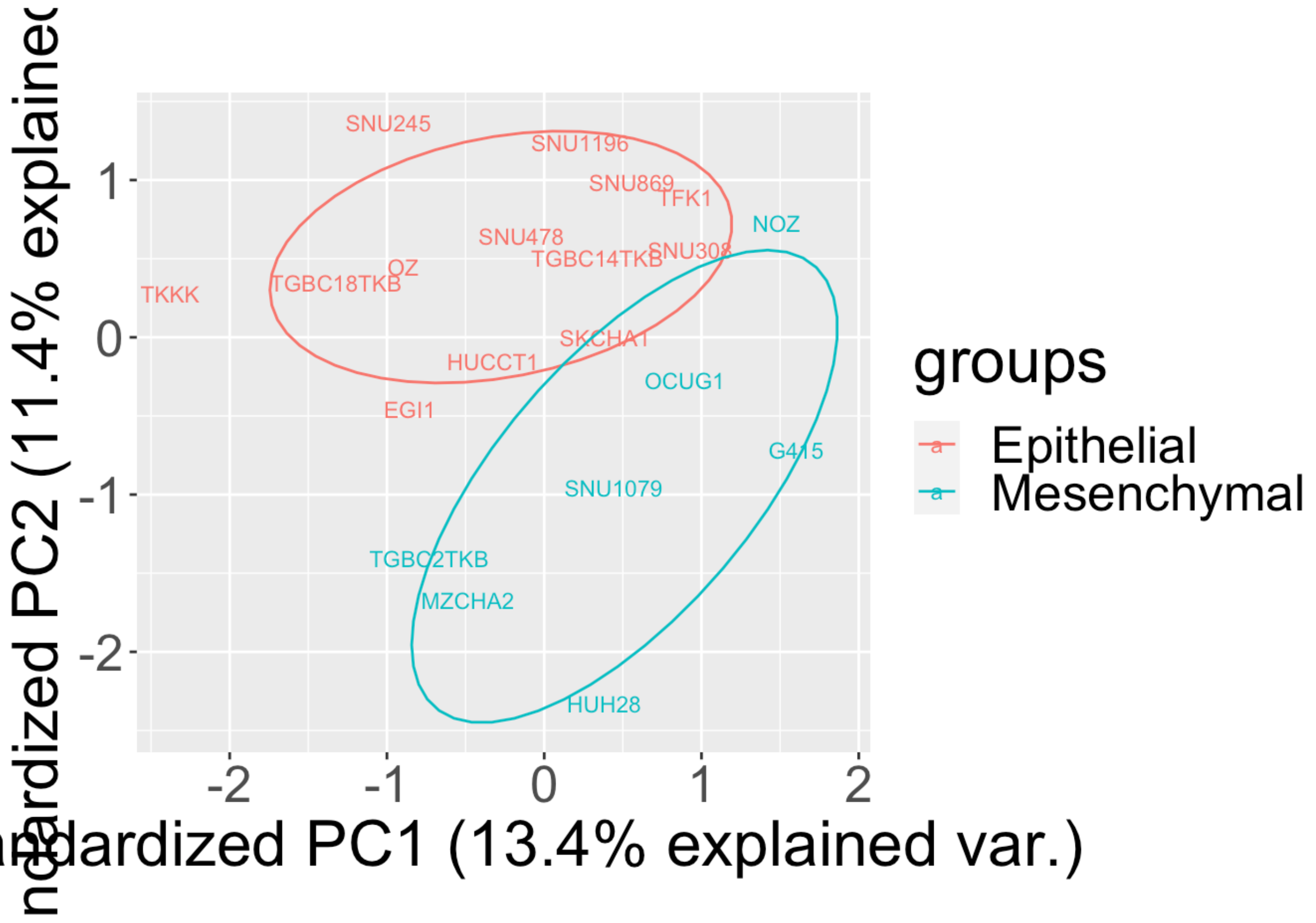
- Decide the number of top PCs to use in the analysis
  - Select top K eigenvectors (w_k) whose corresponding eigenvalues are significantly large (e.g., 5 or 10) by a scree plot

```
# Variation explained by PCs
qplot(1:20, RNAseq.pca$sdev^2) + labs(x = "PCs")
```

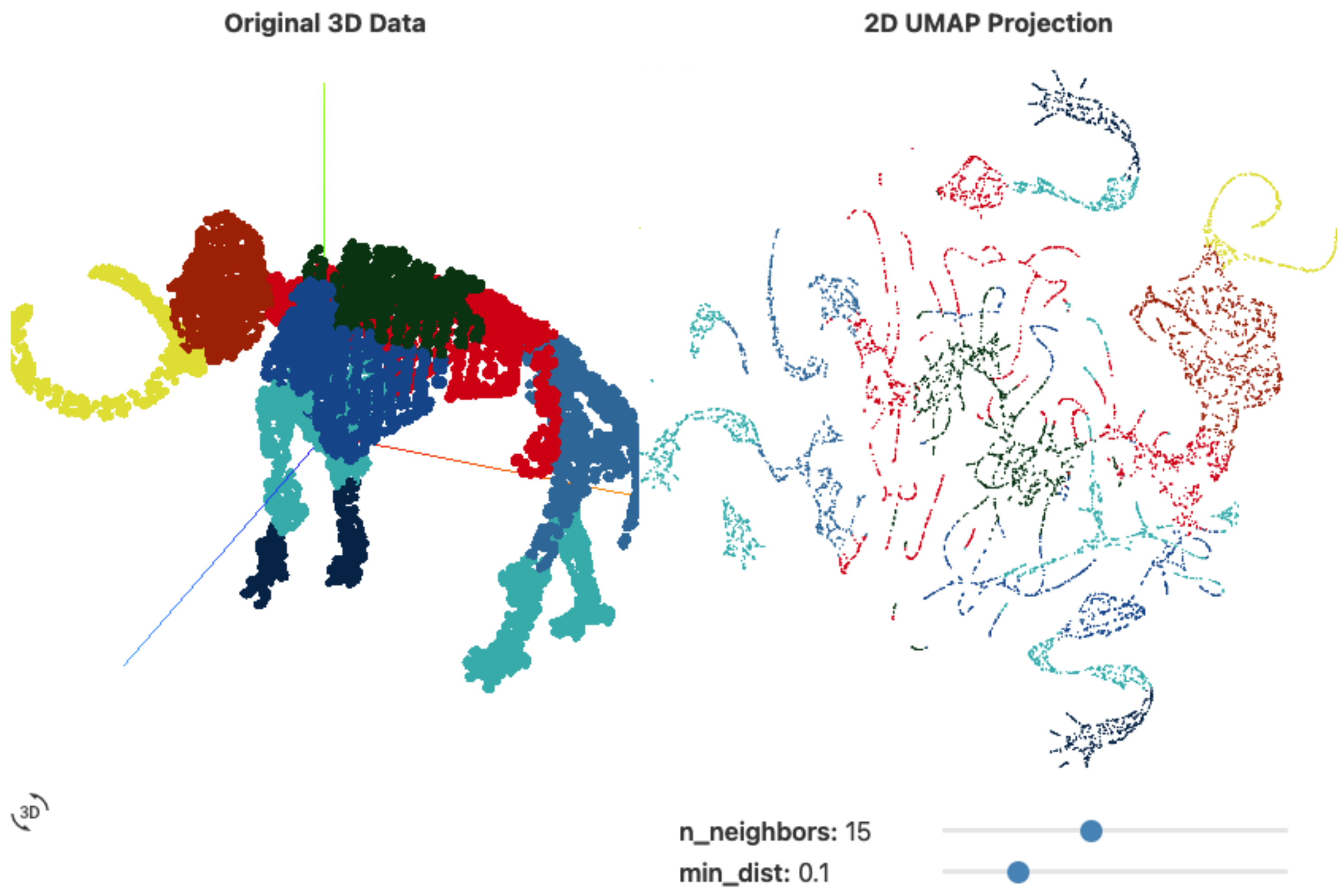


Visualize Top 2 PCs

```
# Plot PC1 vs. PC2
ggbiplot(RNAseq.pca, groups = cell_group$group_label,
         ellipse = TRUE,
         var.axes = FALSE,
         labels = cell_group$cell_line)
```



UMAP (Uniform Manifold Approximation and Projection)



n_neighbors: 15

min_dist: 0.1

Figure 5: *UMAP projections of a 3D woolly mammoth skeleton (50k points, 10k shown) into 2 dimensions, with various settings for the* `n_neighbors` *and* `min_dist` *parameters.*

UMAP : Dimension Reduction

- Uses local Manifold Approximations and patches together their local fuzzy simplicial set representations to construct a topological representation of the high dimensional data.
- Given some low dimensional representation of the data, a similar process can be used to construct an equivalent topological representation.
- UMAP then optimizes the layout of the data representation in the low dimensional space, to minimize the cross-entropy between the two topological representations.
- https://pair-code.github.io/understanding-umap/

UMAP : Dimension Reduction

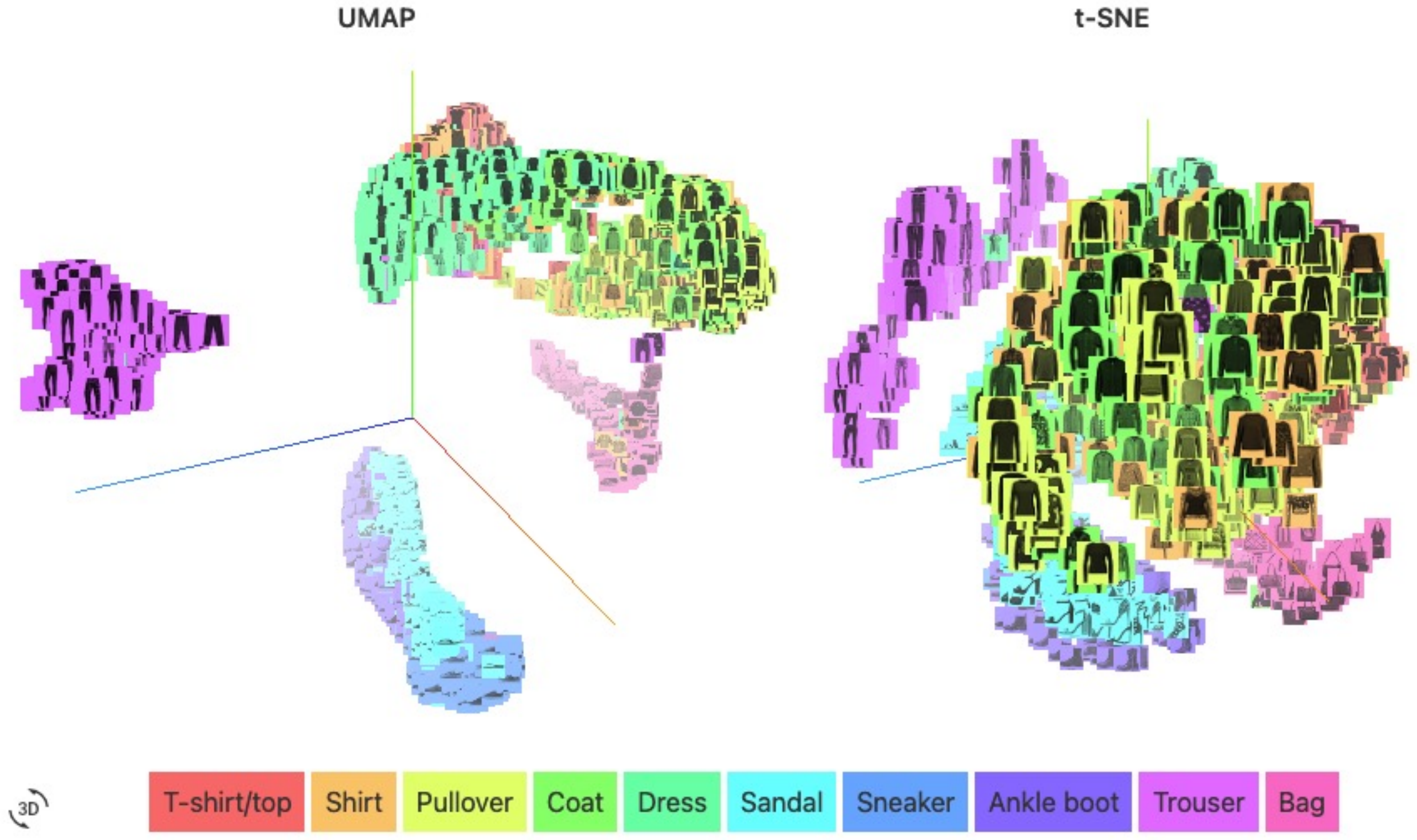


***Figure 2:** Dimensionality reduction applied to the Fashion MNIST dataset. 28x28 images of clothing items in 10 categories are encoded as 784-dimensional vectors and then projected to 3 using UMAP and t-SNE.*

UMAP

```
# Generate UMAP data object
RNAseq.umap = umap(t(RNAseq_CPM.keep.log2))
RNAseq.umap
```

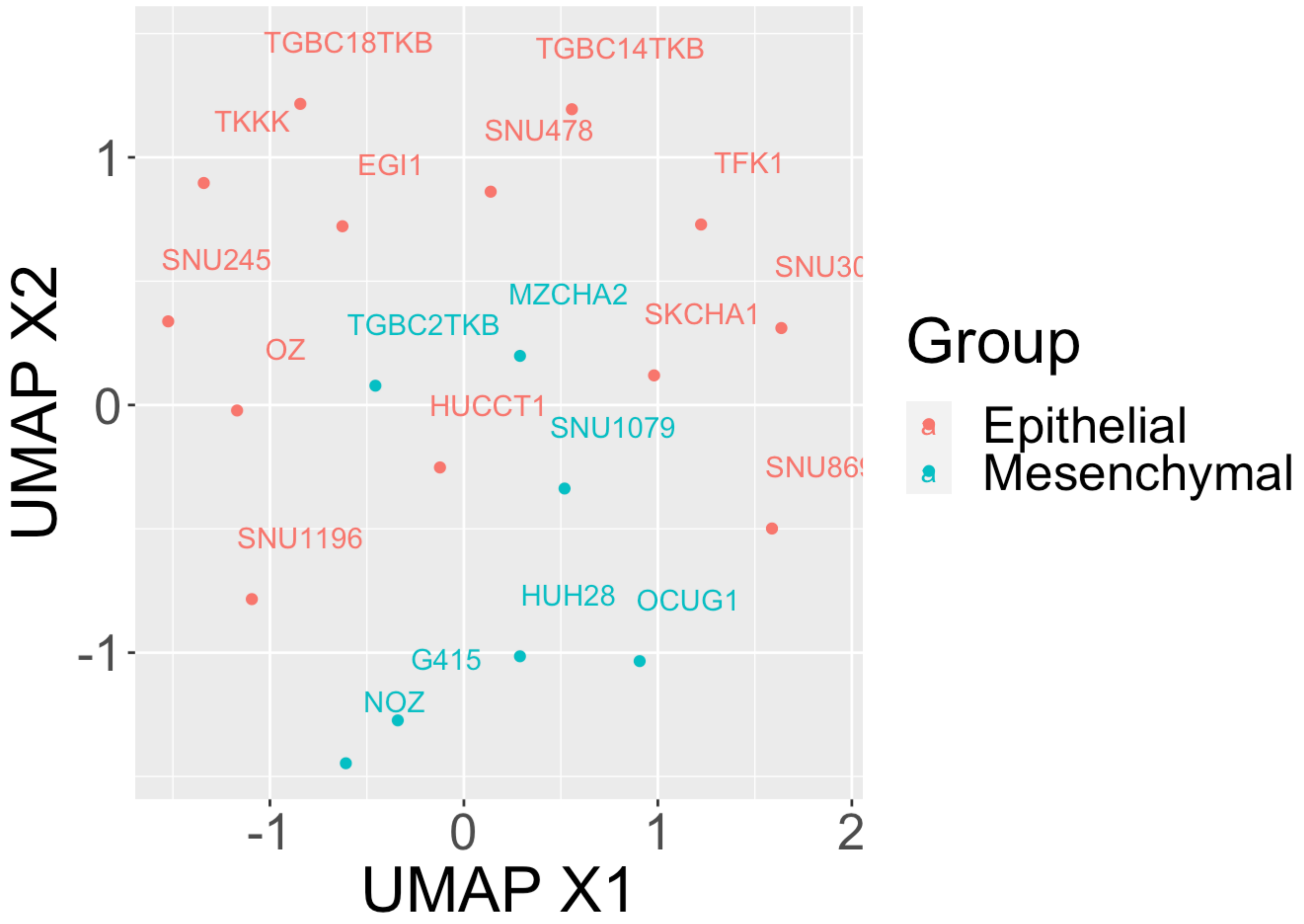
```
## umap embedding of 20 items in 2 dimensions
## object components: layout, data, knn, config
```

```
head(RNAseq.umap$layout)
```

```
##              [,1]       [,2]
## EGI1   -0.6262318  0.7217832
## G415   -0.3408041 -1.2734034
## HUCCT1 -0.1232610 -0.2521879
## HUH28   0.2891026 -1.0146064
## MZCHA2  0.2892878  0.1983404
## NOZ    -0.6085395 -1.4466489
```

UMAP

```r
# Plot UMAP X1 vs. X2
ggplot(data.frame(RNAseq.umap$layout),
       aes(x = X1, y = X2, colour = cell_group$group_label)) +
    geom_point() + labs(x = "UMAP X1", y = "UMAP X2") +
    geom_text(label = rownames(RNAseq.umap$layout), nudge_x = 0.25, nudge_y = 0.25,
    check_overlap = T) + labs(colour = "Group")
```



Differential Gene Expression Analysis

- Test the null hypothesis that gene expression is equally expressed in two groups of samples
- Reject: the gene is a significant differential gene expression that expressed differently between two groups of samples
- Negative binomial distribution is reasonably to be assumed for gene expression read count data
- Generally, the test is conducted based on a negative binomial regression model with gene expression counts as the outcome and group information as the design matrix

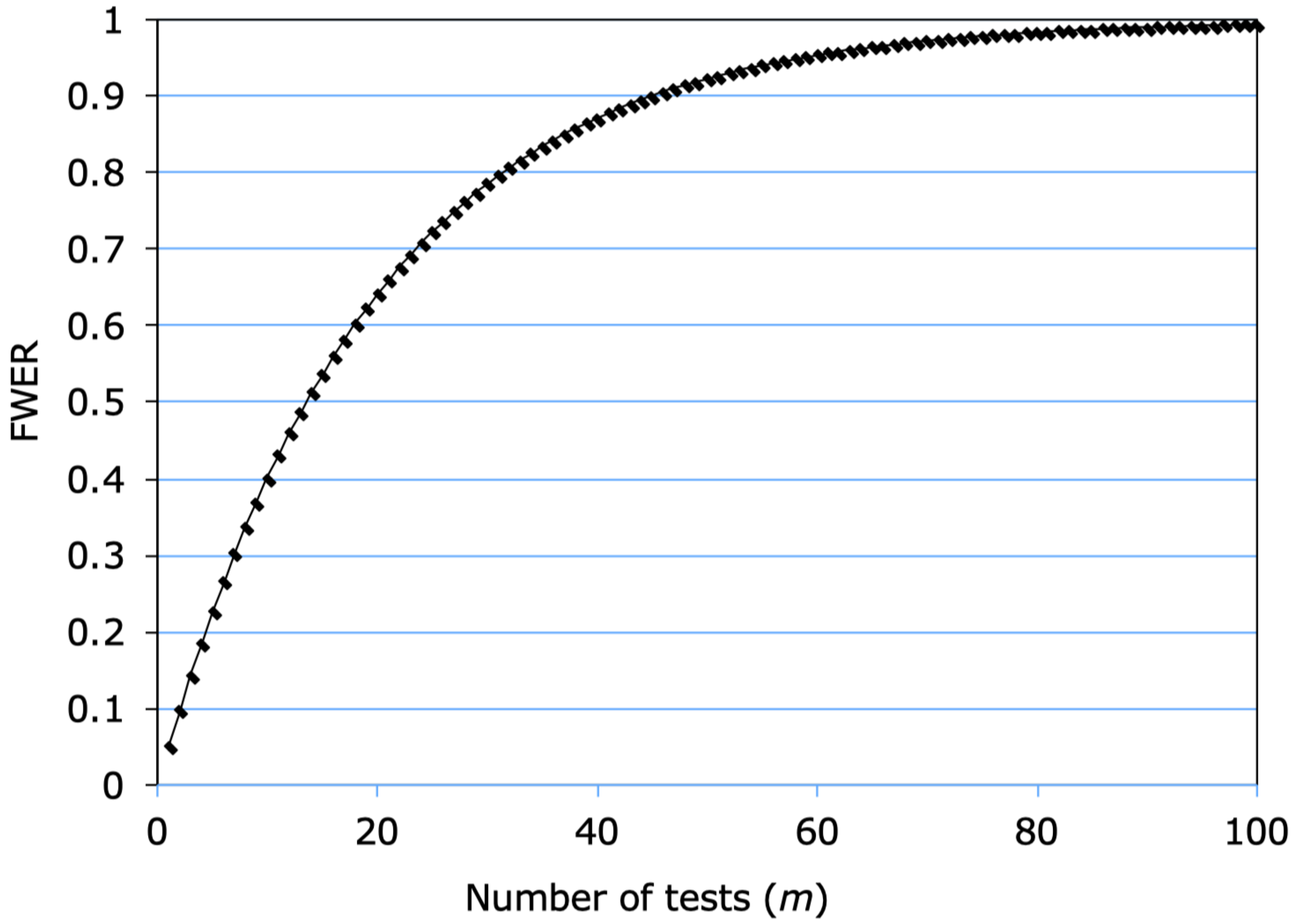
Multiple Testing

- Need to test 10K or 20K genes
- Number of possible false positives with significance threshold = 0.05 under the null hypothesis?
- What is the Family-Wise Error Rate (FWER) for testing 10K genes?
    - Probability of making one or more false discoveries
    - Type I errors when performing multiple hypothesis tests

Consider the case of two association tests

- 2 independent tests performed
  - Test statistics T_1 and $T_{2,}$ *p*-values p_1 and p_2
- What if we reject the null whenever $p < \alpha$?
- If the null hypothesis is actually true in both cases,
  - Probability of rejecting null for test 1 = α
  - Probability of rejecting null for test 2 = α
- What's the probability that both are rejected? α^2
- What's the probability that neither are rejected? $(1- \alpha)^2$
- What's the probability that at least one is rejected? $1-(1- \alpha)^2$
- What if instead of 2 tests, we have *m* tests? $1-(1- \alpha)^m$

Probability of rejecting when null is true if we reject whenever p<.05



What is the solution?

1) State upfront what hypotheses you will test

2) Count the number of tests you will perform

3) **Include a strategy to assess significance while accounting for the number of tests**
   - One strategy: Bonferroni-adjustment of α-level
   - Alternative strategy: Control false discovery rate (FDR)

Bonferroni adjustment of α-level

- Goal: to control the FWER (Family-Wise Error Rate, *a.k.a.* experiment-wide error rate)
  - Probability that we reject the null hypothesis for at least one of the tests that we performed, if all are truly null
- We showed earlier that for m tests, this probability (FWER) = $1 - (1-\alpha)^m$
- So, maybe we could solve for α, to find the α that gives us our desired FWER?

- Bonferroni adjustment: α_{Bonf} = FWER/ m

False Discovery Rate (FDR)

FDR = Expected proportion of rejected hypothesis that were actually true

= $E(V/R)$ for $R > 0$

Counts of hypotheses rejected and not rejected

	Not rejected	Rejected	Total
True null hypotheses	U	V	m_0
False null hypotheses	T	S	$m-m_0$
Total	m-R	R	m

Controlling the FDR: Benjamini-Hochberg (BH) method

- Benjamini-Hochberg procedure controls FDR
    - Order p-values $p_{(1)} \leq p_{(2)} \leq \ldots \leq p_{(m)}$
    - Let $k = \max\{i\}$ such that $p_{(i)} \leq \frac{i}{m} q$ where q is the level at which we desire to control FDR (E(V/R) < q)
        - Compare smallest p to .05/*m*
        - Compare 2nd smallest p to .10/*m*.
        - Take largest *k* such that *k*th smallest $p \leq .05 * k/m$.
        - FDR<.05 for the *k* most significant tests.
        - Reject $H_{(1)} \ldots H_{(k)}$
    - Controlling FDR also controls FWER when all null hypotheses are true

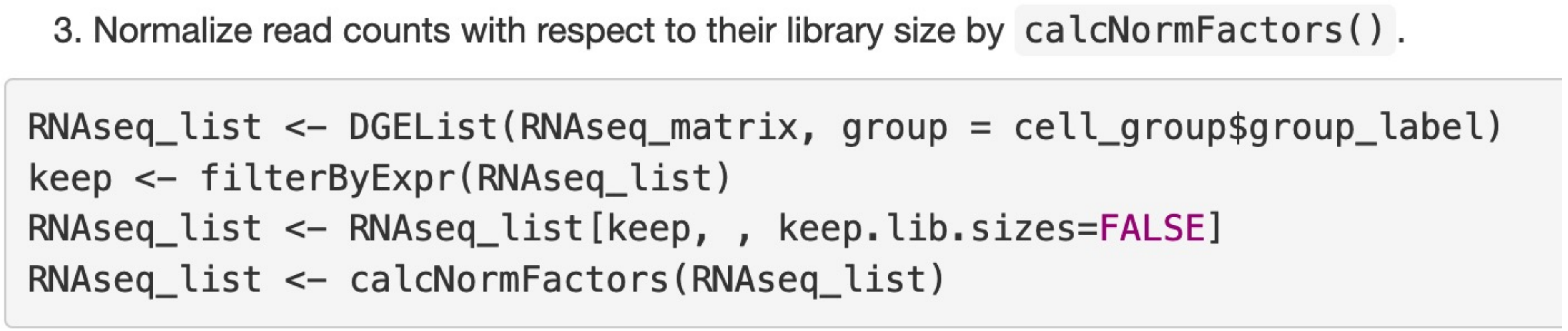
Review: FDR vs. FWER

	Not rejected	Rejected	Total
True null hypotheses	U	V	m_0
False null hypotheses	T	S	$m-m_0$
Total	m-R	R	m

- FDR = E(V/R) = expected proportion of rejected hypotheses that are actually true
- FWER = P(V>0) = probability of rejecting *any* true hypotheses
  - Using a Bonferroni cutoff will control this at your desired level.

EdgeR

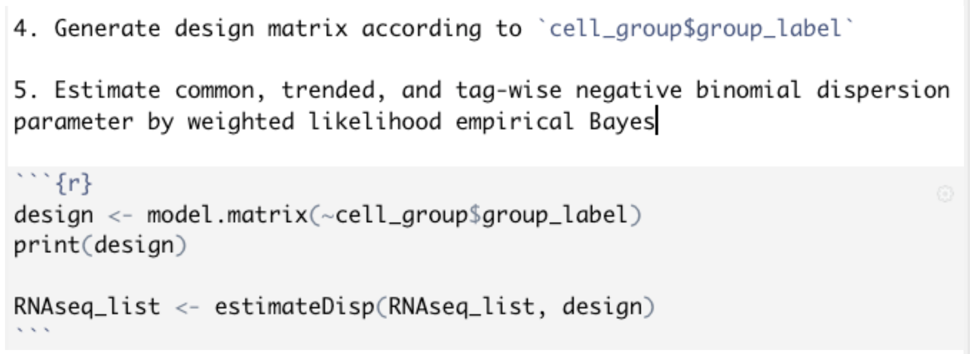
1. Generate a DGEList object from raw RNAseq data RNAseq_matrix.
2. Filter genes using filterByExpr() function, which keeps genes with worthwhile counts in a minimum number of samples (two by default).



3. Normalize read counts with respect to their library size by `calcNormFactors()`.

```r
RNAseq_list <- DGEList(RNAseq_matrix, group = cell_group$group_label)
keep <- filterByExpr(RNAseq_list)
RNAseq_list <- RNAseq_list[keep, , keep.lib.sizes=FALSE]
RNAseq_list <- calcNormFactors(RNAseq_list)
```

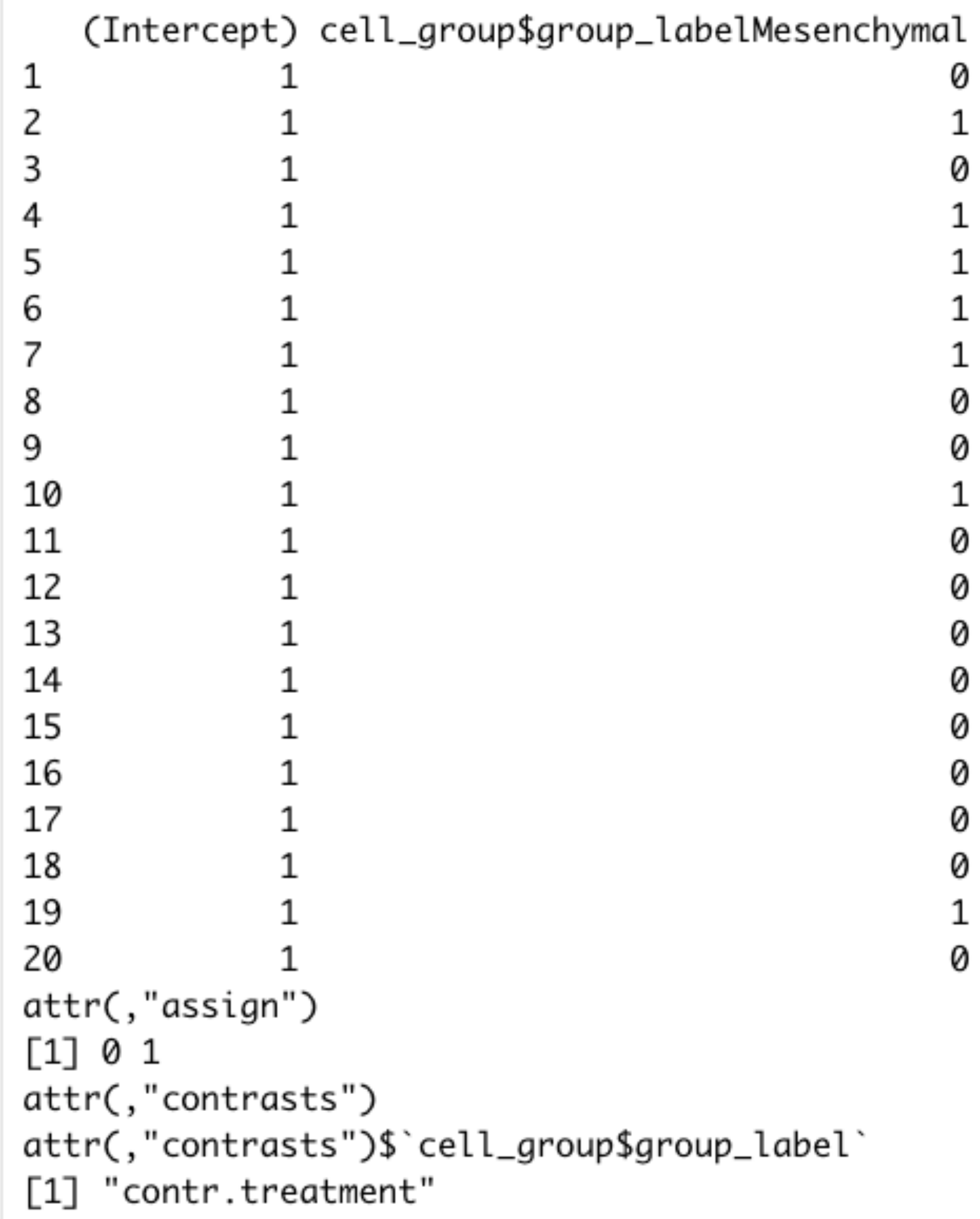
Design Matrix



4. Generate design matrix according to `cell_group$group_label`

5. Estimate common, trended, and tag-wise negative binomial dispersion parameter by weighted likelihood empirical Bayes

~~~
```{r}
design <- model.matrix(~cell_group$group_label)
print(design)

RNAseq_list <- estimateDisp(RNAseq_list, design)
```
~~~



```
   (Intercept) cell_group$group_labelMesenchymal
1            1                                 0
2            1                                 1
3            1                                 0
4            1                                 1
5            1                                 1
6            1                                 1
7            1                                 1
8            1                                 0
9            1                                 0
10           1                                 1
11           1                                 0
12           1                                 0
13           1                                 0
14           1                                 0
15           1                                 0
16           1                                 0
17           1                                 0
18           1                                 0
19           1                                 1
20           1                                 0
attr(,"assign")
[1] 0 1
attr(,"contrasts")
attr(,"contrasts")$`cell_group$group_label`
[1] "contr.treatment"
```

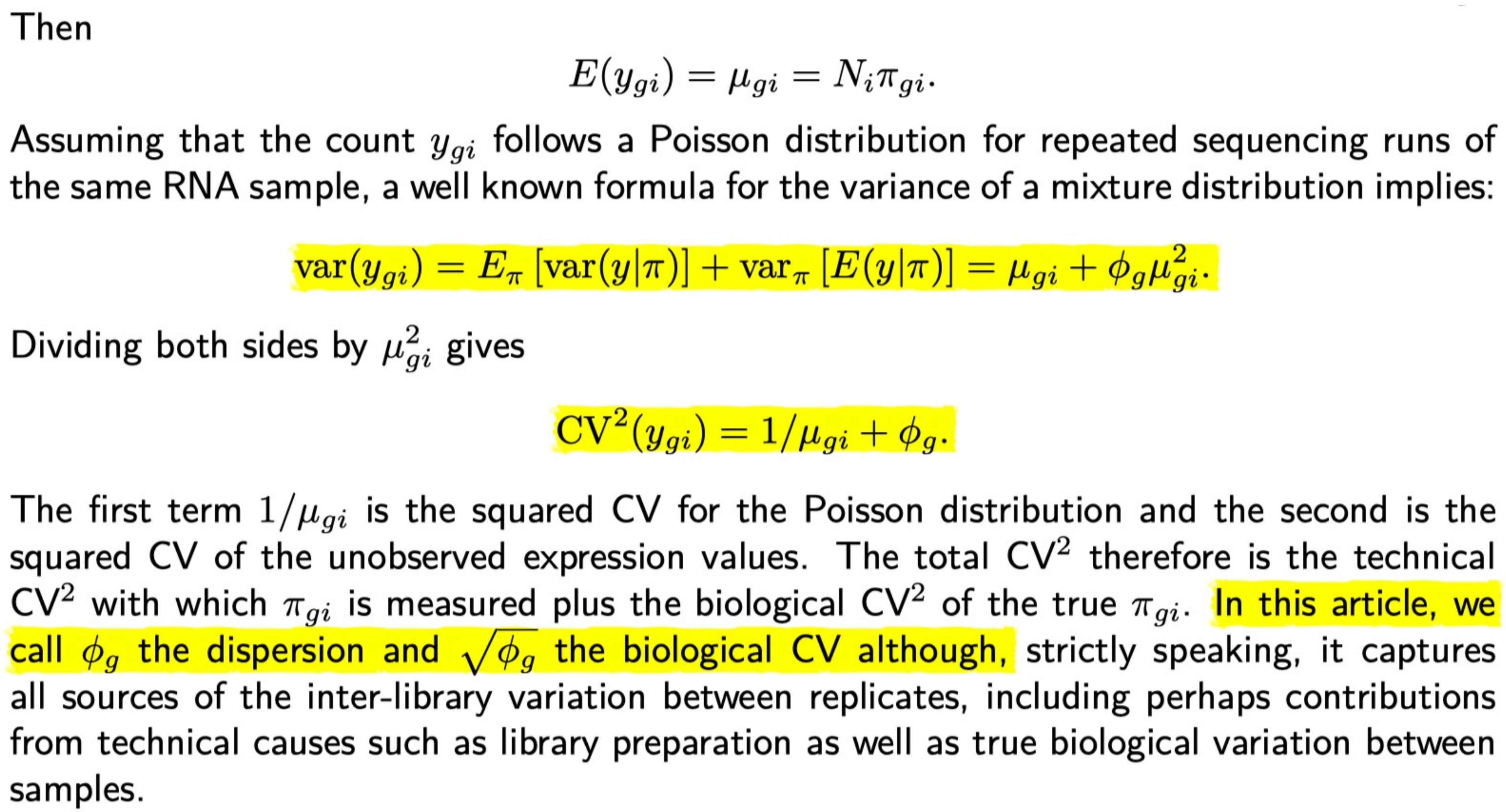
Negative Binomial Model

- Assume π_{gi} be the true fraction of RNAseq reads in sample i that can originate from gene g, sum to 1 across all genes per sample i.

- Let $\sqrt{\phi_g}$ denote the coefficient of variation (CV, standard deviation divided by mean) of π_{gi} among all samples.

- Let y_{gi} denote read count for sample i and gene g, sum to N_i which is the library size.

Negative binomial model



Then

$$E(y_{gi}) = \mu_{gi} = N_i \pi_{gi}.$$

Assuming that the count y_{gi} follows a Poisson distribution for repeated sequencing runs of the same RNA sample, a well known formula for the variance of a mixture distribution implies:

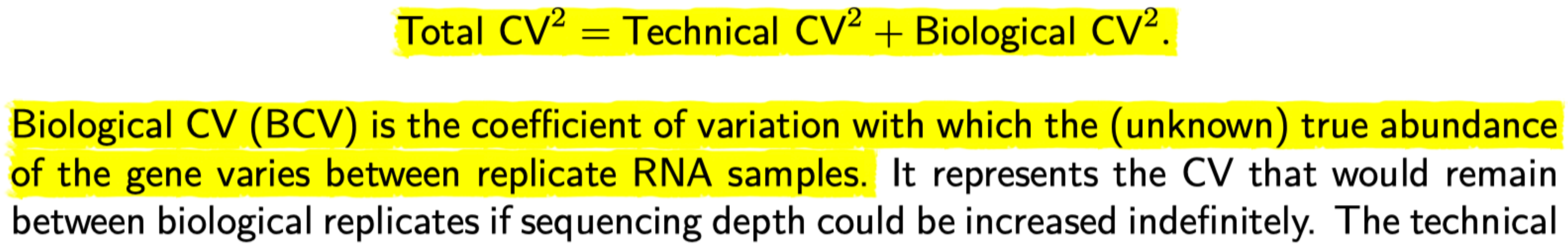
$$\text{var}(y_{gi}) = E_\pi\left[\text{var}(y|\pi)\right] + \text{var}_\pi\left[E(y|\pi)\right] = \mu_{gi} + \phi_g \mu_{gi}^2.$$

Dividing both sides by μ_{gi}^2 gives

$$\text{CV}^2(y_{gi}) = 1/\mu_{gi} + \phi_g.$$

The first term $1/\mu_{gi}$ is the squared CV for the Poisson distribution and the second is the squared CV of the unobserved expression values. The total CV^2 therefore is the technical CV^2 with which π_{gi} is measured plus the biological CV^2 of the true π_{gi}. In this article, we call ϕ_g the dispersion and $\sqrt{\phi_g}$ the biological CV although, strictly speaking, it captures all sources of the inter-library variation between replicates, including perhaps contributions from technical causes such as library preparation as well as true biological variation between samples.

Biological Coefficient of Variation (BCV)



$$\text{Total CV}^2 = \text{Technical CV}^2 + \text{Biological CV}^2.$$

Biological CV (BCV) is the coefficient of variation with which the (unknown) true abundance of the gene varies between replicate RNA samples. It represents the CV that would remain between biological replicates if sequencing depth could be increased indefinitely. The technical

Quasi negative binomial

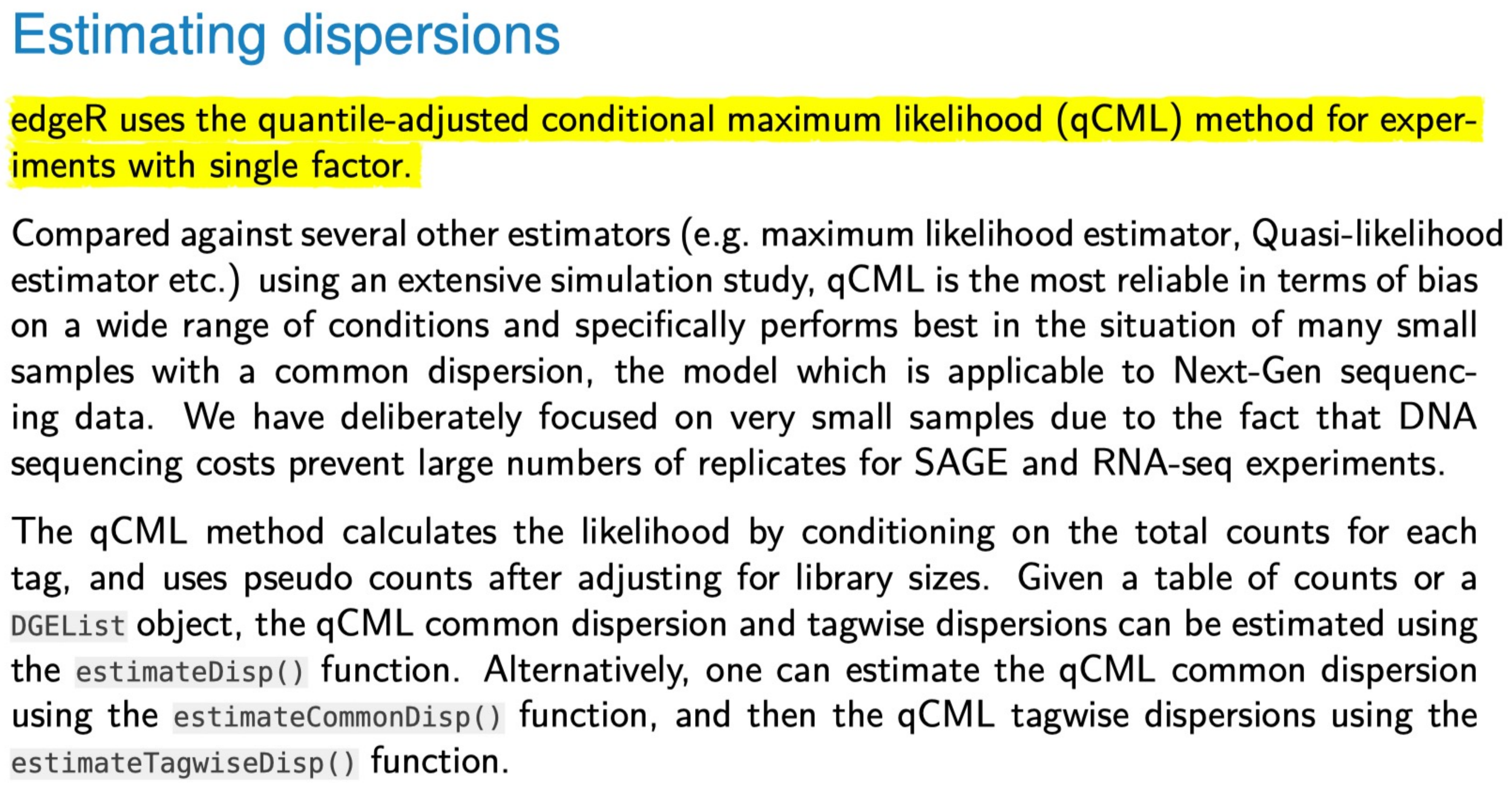
The NB model can be extended with quasi-likelihood (QL) methods to account for gene-specific variability from both biological and technical sources [23, 22]. Under the QL framework, the variance of the count y_{gi} is a quadratic function of the mean,

$$\text{var}(y_{gi}) = \sigma_g^2(\mu_{gi} + \phi\mu_{gi}^2),$$

where ϕ is the NB dispersion parameter and σ_g^2 is the QL dispersion parameter.

Any increase in the observed variance of y_{gi} will be modelled by an increase in the estimates for ϕ and/or σ_g^2. In this model, the NB dispersion ϕ is a global parameter whereas the QL is gene-specific, so the two dispersion parameters have different roles. The NB dispersion describes the overall biological variability across all genes. It represents the observed variation that is attributable to inherent variability in the biological system, in contrast to the Poisson variation from sequencing. The QL dispersion picks up any gene-specific variability above and below the overall level.

Quantile-adjusted Conditional Maximum Likelihood



Estimating dispersions

edgeR uses the quantile-adjusted conditional maximum likelihood (qCML) method for experiments with single factor.

Compared against several other estimators (e.g. maximum likelihood estimator, Quasi-likelihood estimator etc.) using an extensive simulation study, qCML is the most reliable in terms of bias on a wide range of conditions and specifically performs best in the situation of many small samples with a common dispersion, the model which is applicable to Next-Gen sequencing data. We have deliberately focused on very small samples due to the fact that DNA sequencing costs prevent large numbers of replicates for SAGE and RNA-seq experiments.

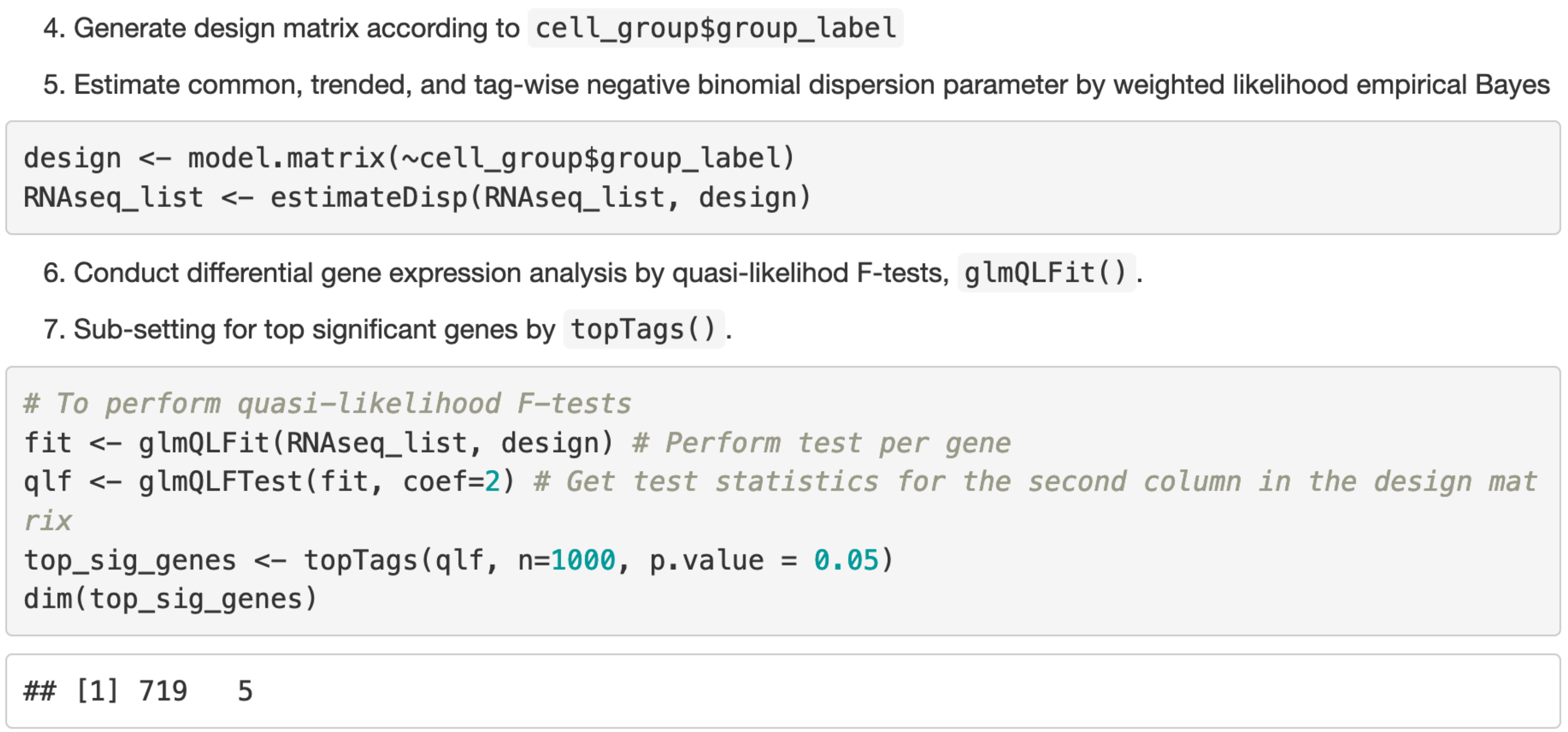
The qCML method calculates the likelihood by conditioning on the total counts for each tag, and uses pseudo counts after adjusting for library sizes. Given a table of counts or a `DGEList` object, the qCML common dispersion and tagwise dispersions can be estimated using the `estimateDisp()` function. Alternatively, one can estimate the qCML common dispersion using the `estimateCommonDisp()` function, and then the qCML tagwise dispersions using the `estimateTagwiseDisp()` function.

Testing for DE genes

For all the Next-Gen squencing data analyses we consider here, people are most interested in finding differentially expressed genes/tags between two (or more) groups. Once negative binomial models are fitted and dispersion estimates are obtained, we can proceed with testing procedures for determining differential expression using the exact test.

The exact test is based on the qCML methods. Knowing the conditional distribution for the sum of counts in a group, we can compute exact p-values by summing over all sums of counts that have a probability less than the probability under the null hypothesis of the observed sum of counts. The exact test for the negative binomial distribution has strong parallels with Fisher's exact test.

EdgeR



4. Generate design matrix according to `cell_group$group_label`
5. Estimate common, trended, and tag-wise negative binomial dispersion parameter by weighted likelihood empirical Bayes

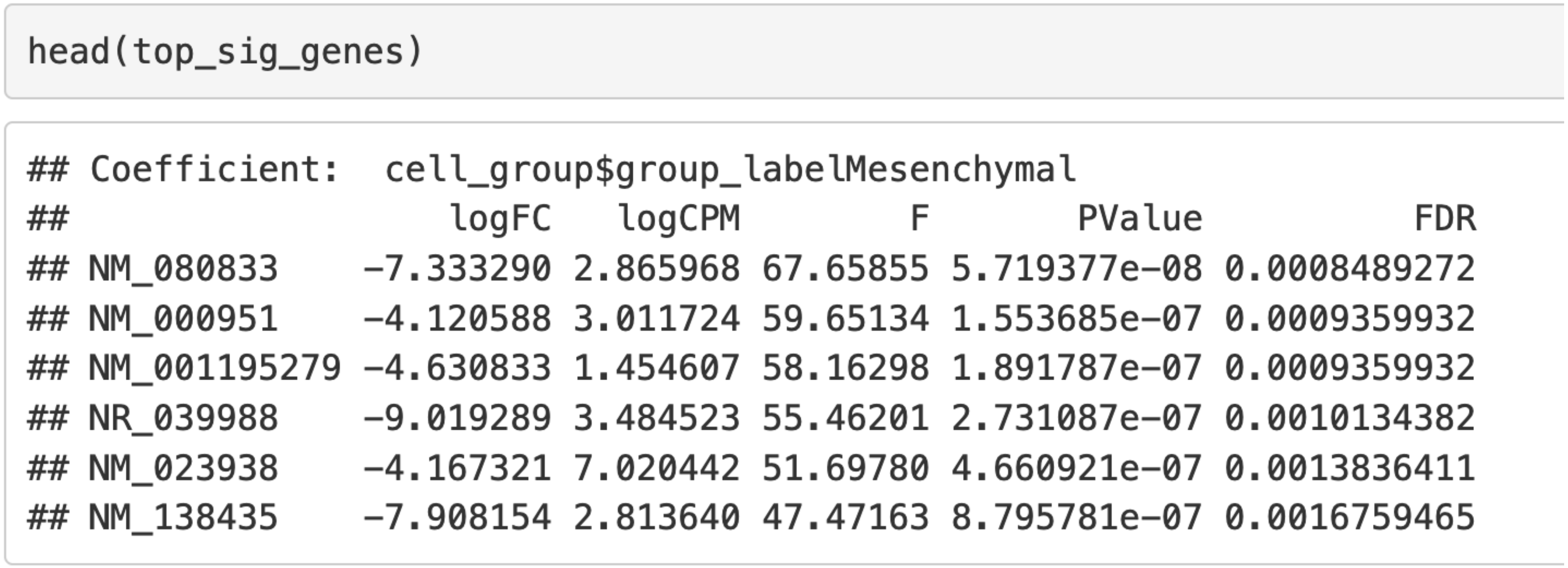
```
design <- model.matrix(~cell_group$group_label)
RNAseq_list <- estimateDisp(RNAseq_list, design)
```

6. Conduct differential gene expression analysis by quasi-likelihod F-tests, `glmQLFit()`.
7. Sub-setting for top significant genes by `topTags()`.

```
# To perform quasi-likelihood F-tests
fit <- glmQLFit(RNAseq_list, design) # Perform test per gene
qlf <- glmQLFTest(fit, coef=2) # Get test statistics for the second column in the design mat
rix
top_sig_genes <- topTags(qlf, n=1000, p.value = 0.05)
dim(top_sig_genes)
```

```
## [1] 719   5
```

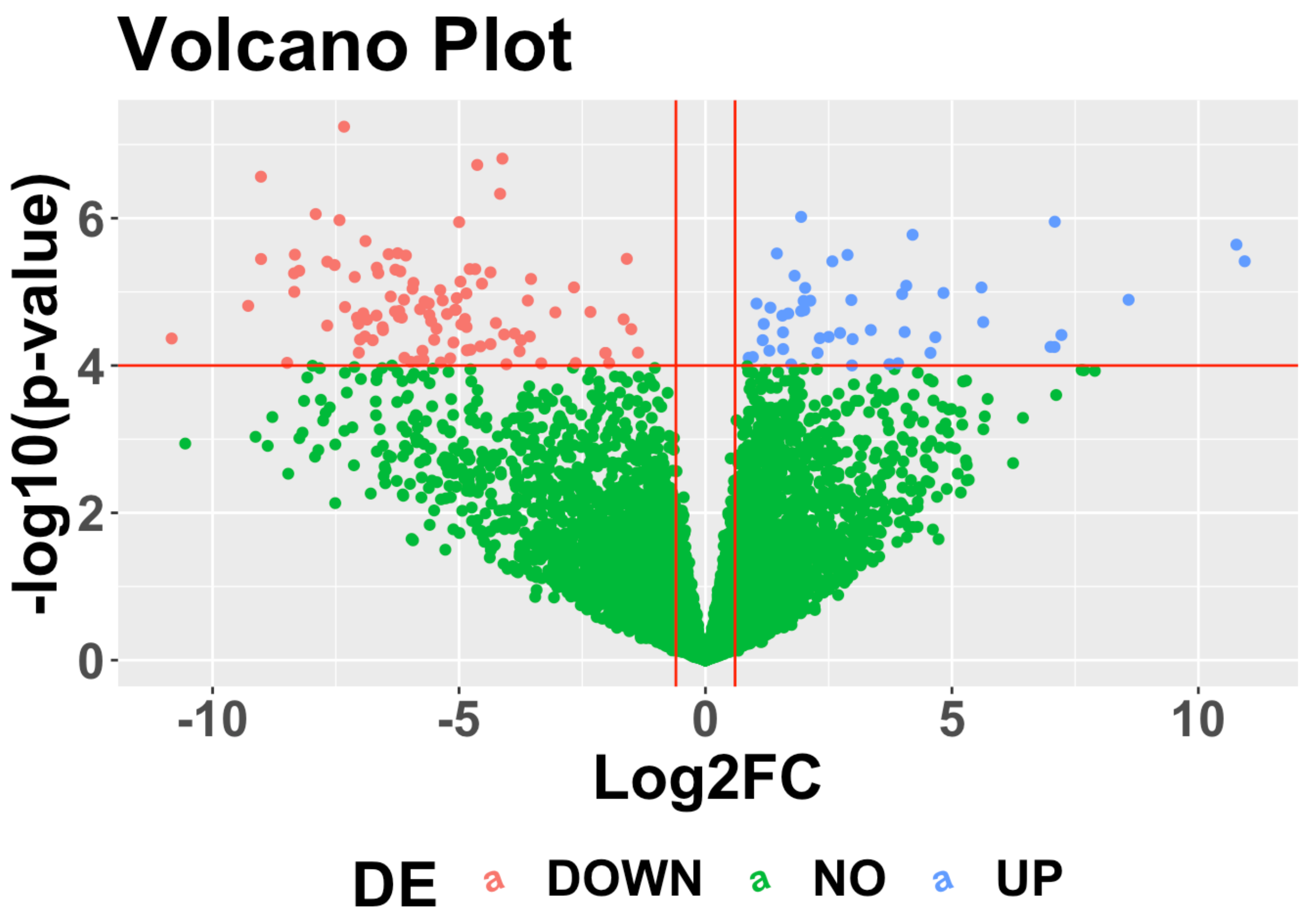
FDR (BH) Method is Used by EdgeR



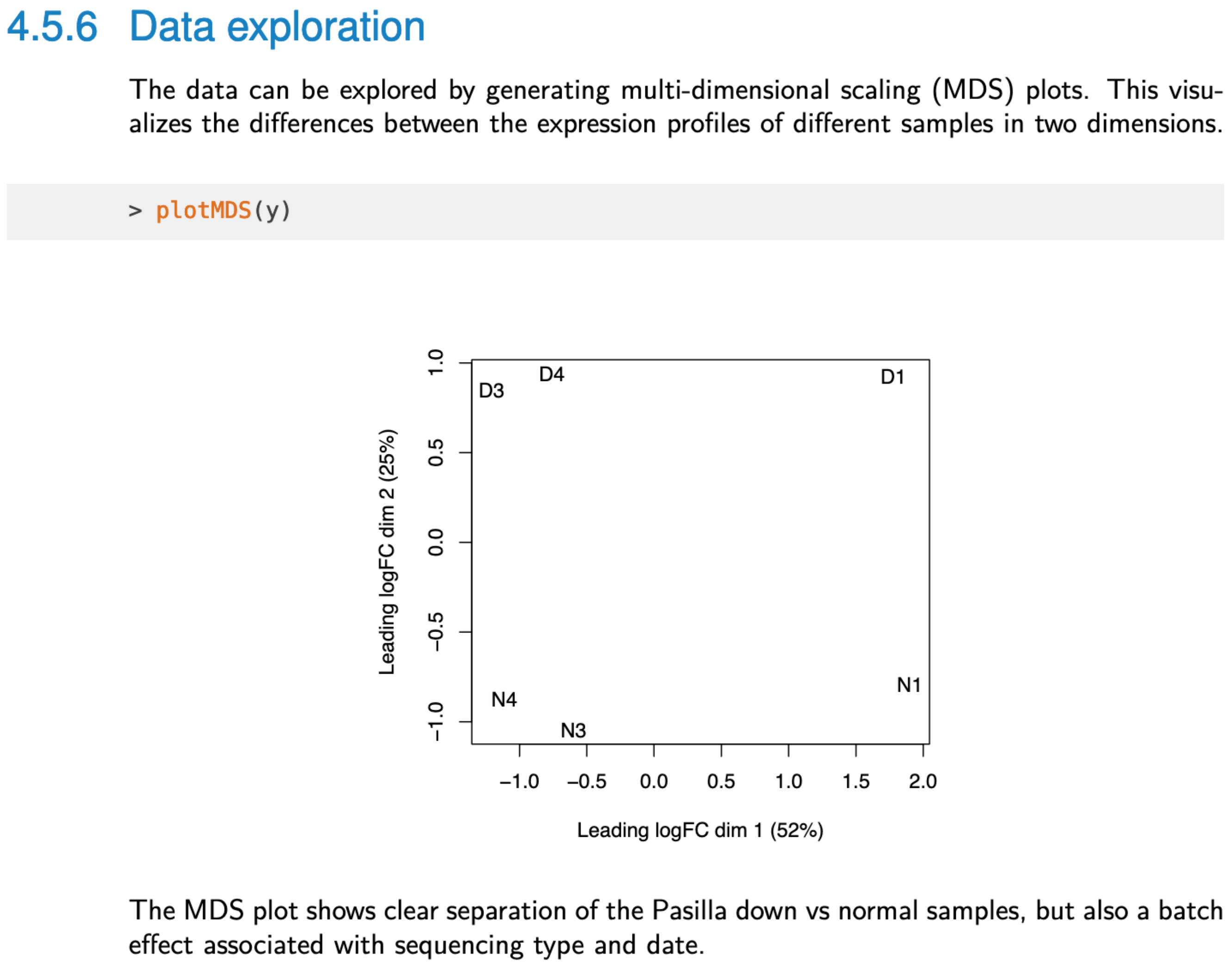
```r
head(top_sig_genes)
```

```
## Coefficient:  cell_group$group_labelMesenchymal
##                   logFC   logCPM        F       PValue          FDR
## NM_080833    -7.333290 2.865968 67.65855 5.719377e-08 0.0008489272
## NM_000951    -4.120588 3.011724 59.65134 1.553685e-07 0.0009359932
## NM_001195279 -4.630833 1.454607 58.16298 1.891787e-07 0.0009359932
## NR_039988    -9.019289 3.484523 55.46201 2.731087e-07 0.0010134382
## NM_023938    -4.167321 7.020442 51.69780 4.660921e-07 0.0013836411
## NM_138435    -7.908154 2.813640 47.47163 8.795781e-07 0.0016759465
```

Visualize Differential Gene Expression Analysis Results : Volcano Plot



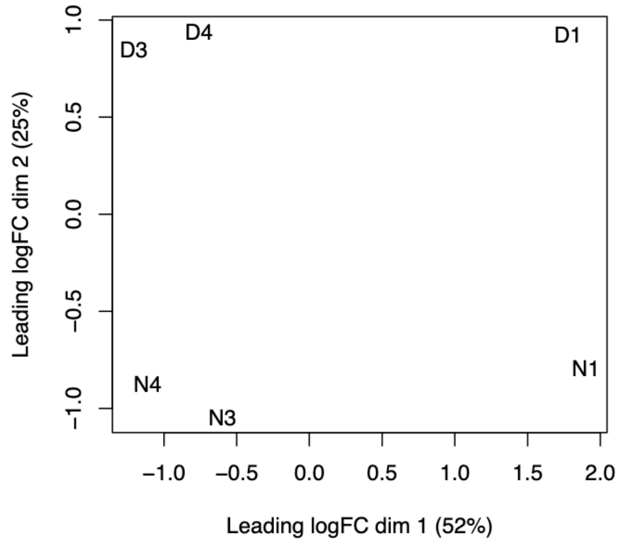
Example 2: Account for Batch Effects by EdgeR



4.5.6 Data exploration

The data can be explored by generating multi-dimensional scaling (MDS) plots. This visualizes the differences between the expression profiles of different samples in two dimensions.

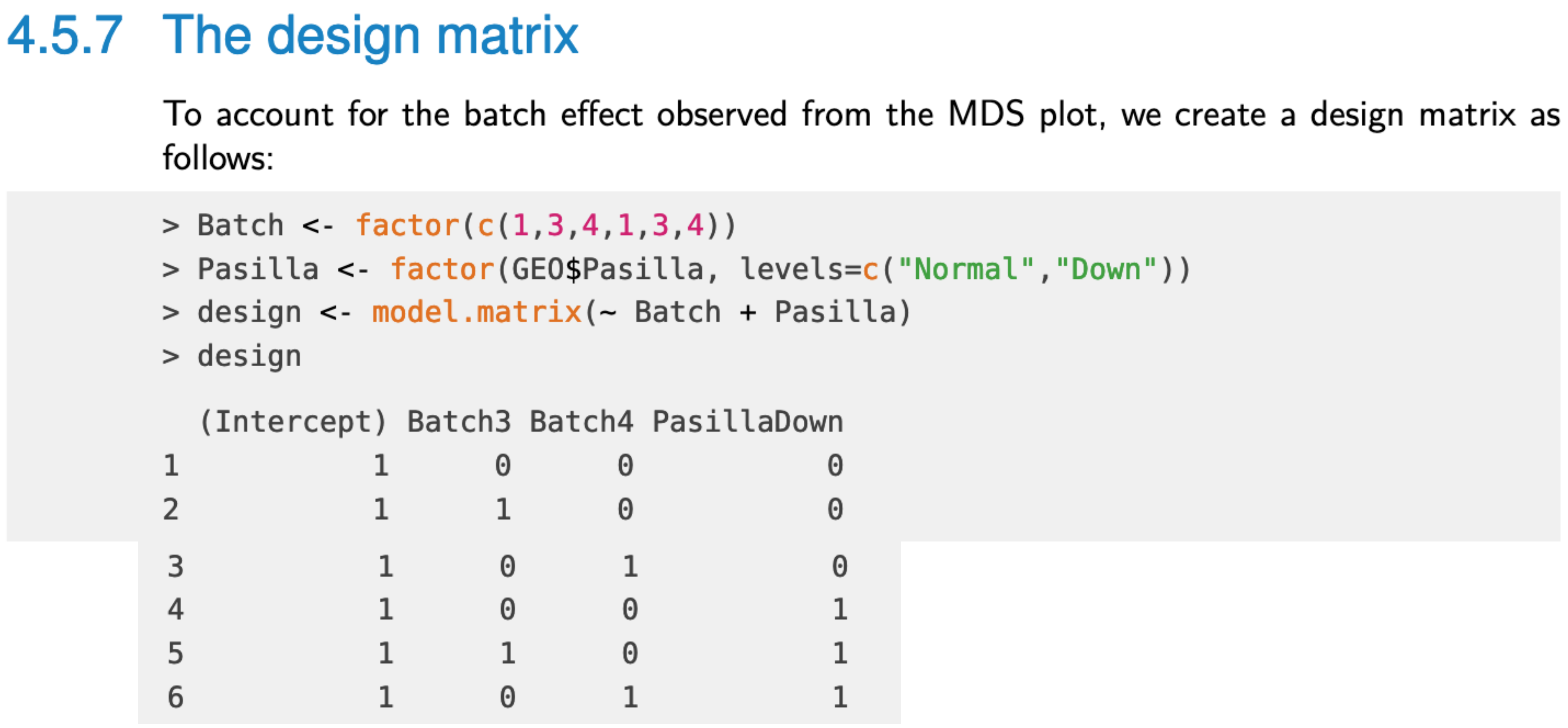
```
> plotMDS(y)
```



The MDS plot shows clear separation of the Pasilla down vs normal samples, but also a batch effect associated with sequencing type and date.

Note: Filtering and normalization has been applied to the DGEList object.

Multiple factors and covariates can be accounted through the design matrix in EdgeR



4.5.7 The design matrix

To account for the batch effect observed from the MDS plot, we create a design matrix as follows:

```
> Batch <- factor(c(1,3,4,1,3,4))
> Pasilla <- factor(GEO$Pasilla, levels=c("Normal","Down"))
> design <- model.matrix(~ Batch + Pasilla)
> design

  (Intercept) Batch3 Batch4 PasillaDown
1           1      0      0           0
2           1      1      0           0
3           1      0      1           0
4           1      0      0           1
5           1      1      0           1
6           1      0      1           1
```

Generalized linear models

Generalized linear models (GLMs) are an extension of classical linear models to nonnormally distributed response data [9]. GLMs specify probability distributions according to their mean-variance relationship, for example the quadratic mean-variance relationship specified above for read counts. Assuming that an estimate is available for ϕ_g, so the variance can be evaluated for any value of μ_{gi}, GLM theory can be used to fit a log-linear model

$$\log \mu_{gi} = \mathbf{x}_i^T \boldsymbol{\beta}_g + \log N_i$$

Estimating dispersions

For general experiments (with multiple factors), edgeR uses the Cox-Reid profile-adjusted likelihood (CR) method in estimating dispersions [25]. The CR method is derived to overcome the limitations of the qCML method as mentioned above. It takes care of multiple factors by fitting generalized linear models (GLM) with a design matrix.

Testing for DE genes

For general experiments, once dispersion estimates are obtained and negative binomial generalized linear models are fitted, we can proceed with testing procedures for determining differential expression using either quasi-likelihood (QL) F-test or likelihood ratio test.

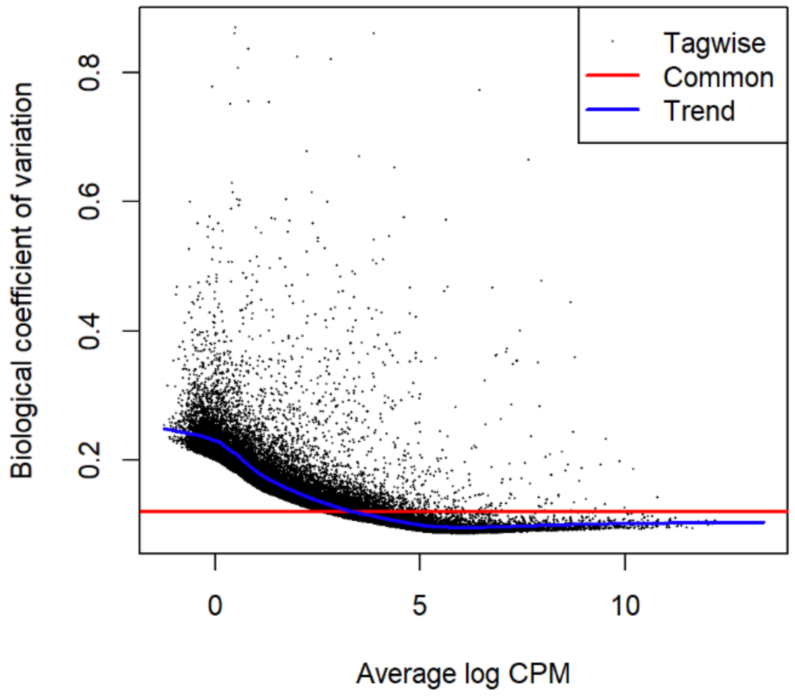
While the likelihood ratio test is a more obvious choice for inferences with GLMs, the QL F-test is preferred as it reflects the uncertainty in estimating the dispersion for each gene. It provides more robust and reliable error rate control when the number of replicates is small. The QL dispersion estimation and hypothesis testing can be done by using the functions `glmQLFit()` and `glmQLFTest()`.

Given raw counts, NB dispersion(s) and a design matrix, `glmQLFit()` fits the negative binomial GLM for each tag and produces an object of class `DGEGLM` with some new components. This `DGEGLM` object can then be passed to `glmQLFTest()` to carry out the QL F-test. User can select one or more coefficients to drop from the full design matrix. This gives the null model against which the full model is compared. Tags can then be ranked in order of evidence for differential expression, based on the p-value computed for each tag.

Estimating the dispersion

We estimate NB dispersions using the `estimateDisp` function. The estimated dispersions can be visualized with `plotBCV`.

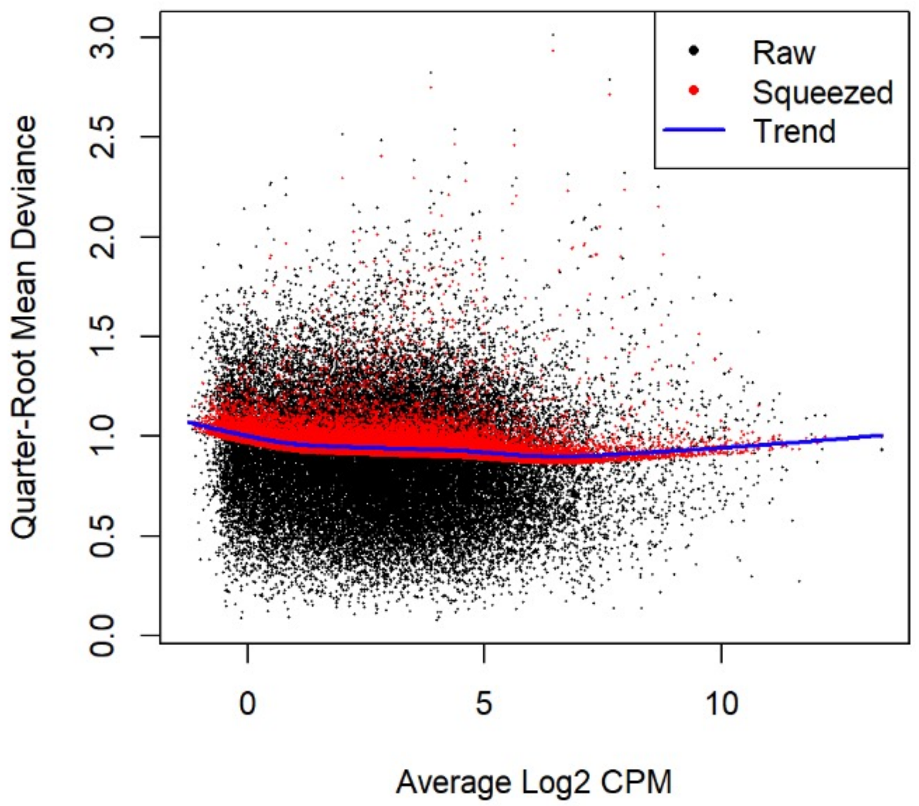
```
> y <- estimateDisp(y, design, robust=TRUE)
> y$common.dispersion

[1] 0.0145

> plotBCV(y)
```



Note that only the trended dispersion is used under the quasi-likelihood (QL) pipeline. The tagwise and common estimates are shown here but will not be used further.

For the QL dispersions, estimation can be performed using the `glmQLFit` function. The results can be visualized with the `plotQLDisp` function.

```
> fit <- glmQLFit(y, design, robust=TRUE)
> plotQLDisp(fit)
```



4.5.9 Differential expression

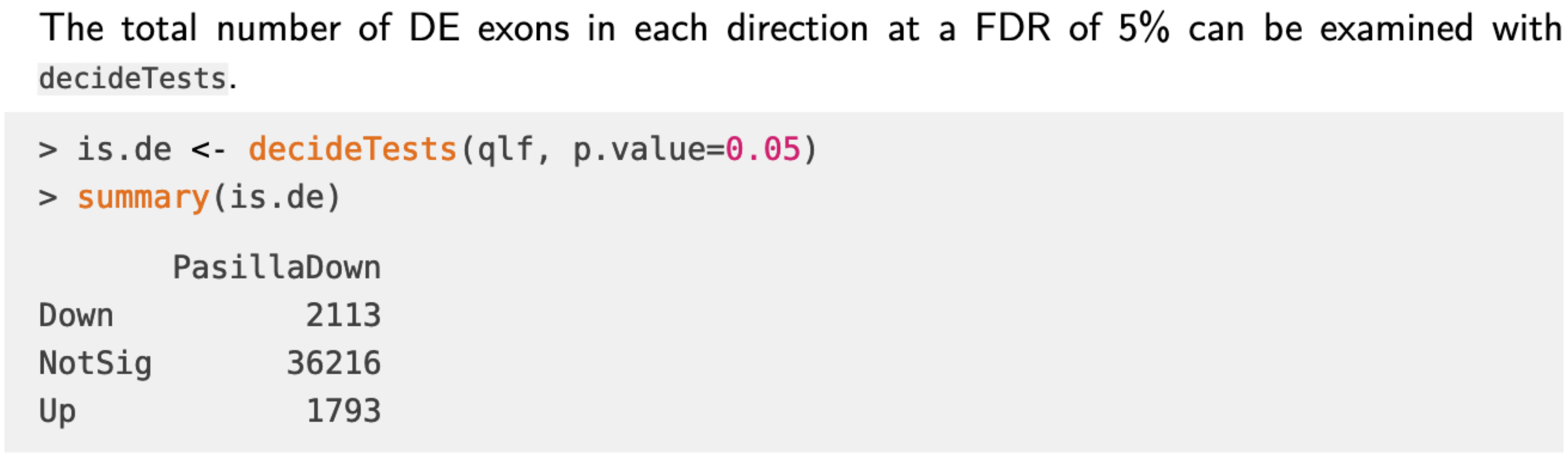
We test for differentially expressed exons between Pasilla knockdown and normal using the QL F-test.

```
> qlf <- glmQLFTest(fit, coef=4)
```

The top set of most significant exons can be examined with `topTags`. Here, a positive log-fold change represents exons that are up in Pasilla knockdown over normal. Multiplicity correction is performed by applying the Benjamini-Hochberg method on the p-values, to control the false discovery rate (FDR).

```
> topTags(qlf)

Coefficient:  PasillaDown 
       GeneID Chr    Start      End Length      Symbol logFC logCPM   F
150709  32007   X 10674926 10676128   1203        sesB -3.26   7.21 944
150713  32007   X 10675026 10676128   1103        sesB -3.26   7.21 943
150697  32008   X 10672987 10673728    742        Ant2  2.85   6.14 851
91614   42865  3R 19970915 19971592    678        Kal1 -4.43   3.81 754
107856  44030  3L  2561932  2562843    912         msn -2.46   5.59 601
150702  32008   X 10674230 10674694    465        Ant2  2.96   4.55 570
150695  32008   X 10674230 10674559    330        Ant2  2.95   4.54 569
70750   44258  3R  5271691  5272628    938          ps -2.28   5.95 567
11333   44548  2R  6407125  6408782   1658        lola  2.25   6.14 558
96434   43230  3R 22695915 22696094    180 BM-40-SPARC -2.28   8.54 536
         PValue      FDR
150709 4.58e-15 9.27e-11
150713 4.62e-15 9.27e-11
150697 9.96e-15 1.33e-10
91614  2.45e-14 2.46e-10
107856 1.32e-13 1.01e-09
150702 1.95e-13 1.01e-09
```

Summarize Differential Gene Expression Analysis Results



The total number of DE exons in each direction at a FDR of 5% can be examined with `decideTests`.

```
> is.de <- decideTests(qlf, p.value=0.05)
> summary(is.de)

       PasillaDown
Down          2113
NotSig       36216
Up            1793
```

Help links

- R function: prcomp() ; https://www.rdocumentation.org/packages/stats/versions/3.5.1/topics/prcomp

- UMAP: https://cran.r-project.org/web/packages/umap/vignettes/umap.html

- EdgeR: https://www.bioconductor.org/packages/devel/bioc/vignettes/edgeR/inst/doc/edgeRUsersGuide.pdf