Sequencing: Single Cell RNA-seq

Lieven Clement and Koen Van den Berge

Last edited on 21 November, 2023

Contents

1 Introduction

1.1 Evolution of gene expression measurements

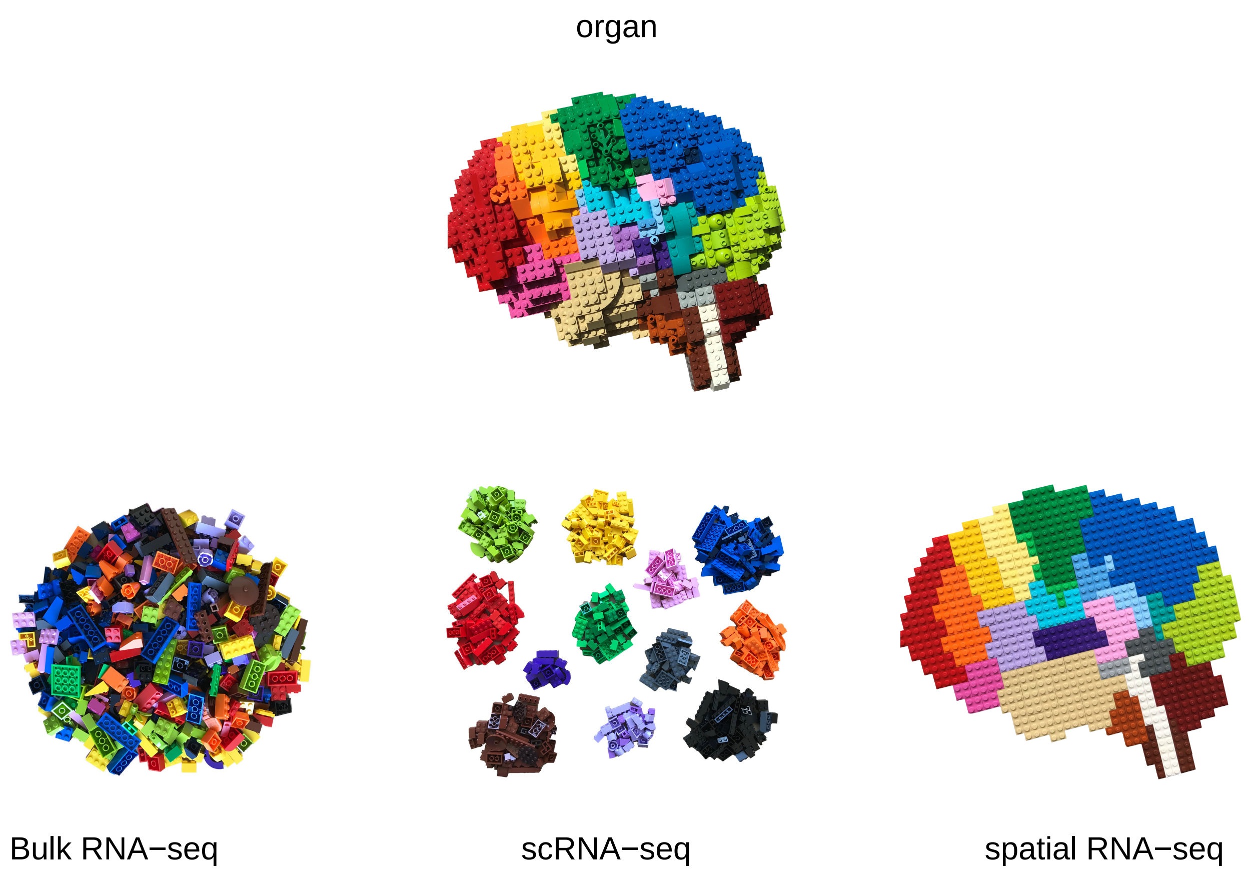


Figure 1: Image credit: @BoXia7

1.2 Exciting applications

1.3 Bulk vs single cell

1.3.1 Protocol

- Bulk RNA-seq datasets are typically characterized by a high sequencing depth, i.e., typically millions of reads are sequenced for each individual sample. This is possible since there are often only a limited number of samples being sequenced.
- This is different for single-cell RNA-sequencing (scRNA-seq), where many individual cells (thousands to millions) are sequenced. Due to this high cell number, only a limited number of sequencing reads are typically available for each individual cell.

This difference in wetlab protocol will have a significant impact on the statistical analysis of the data. The data analysis is affected by the data-generating protocol, and one must carefully take this into consideration.

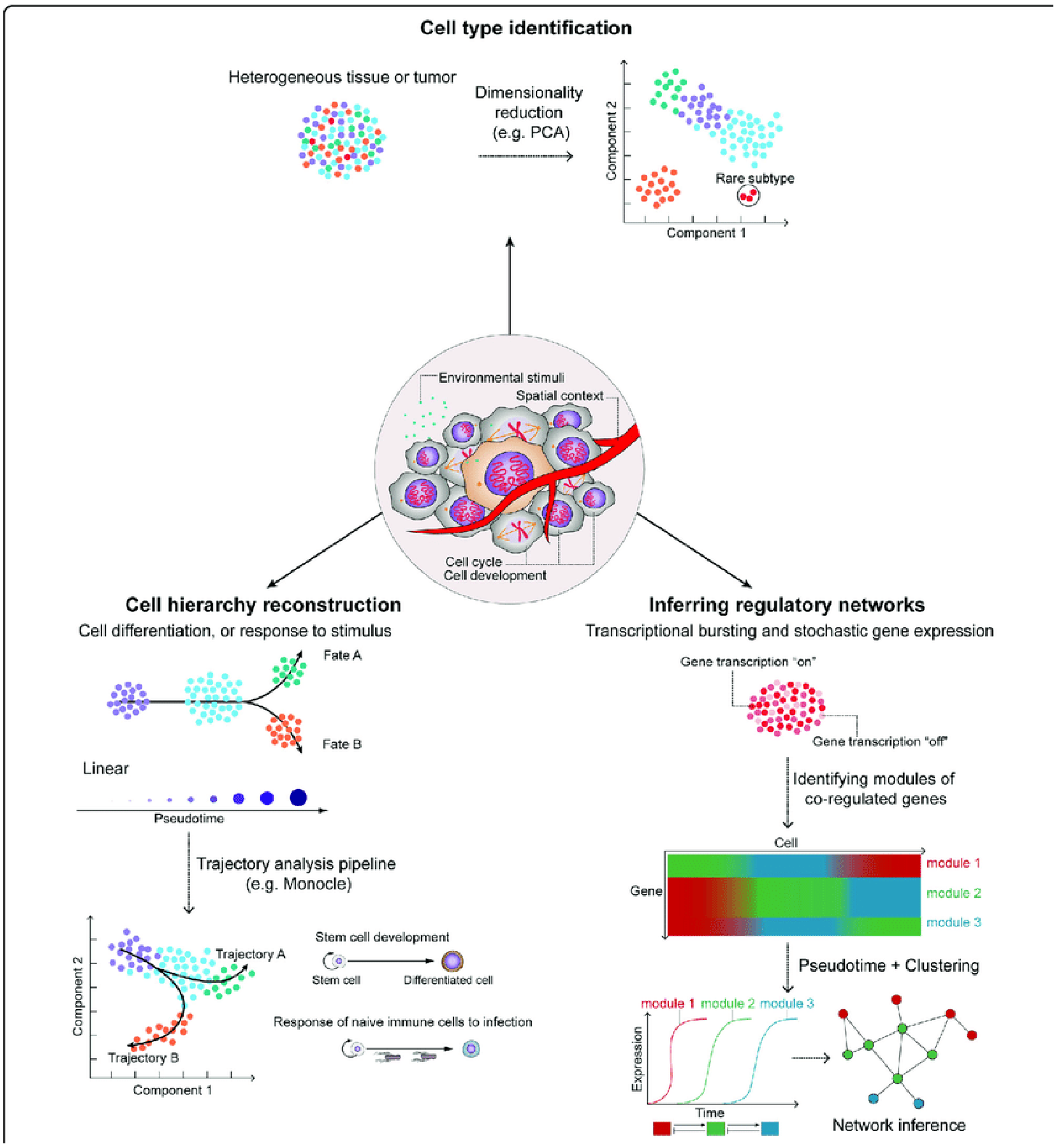


Figure 2: Hwang et al. 2018, doi: 10.1038/s12276-018-0071-8

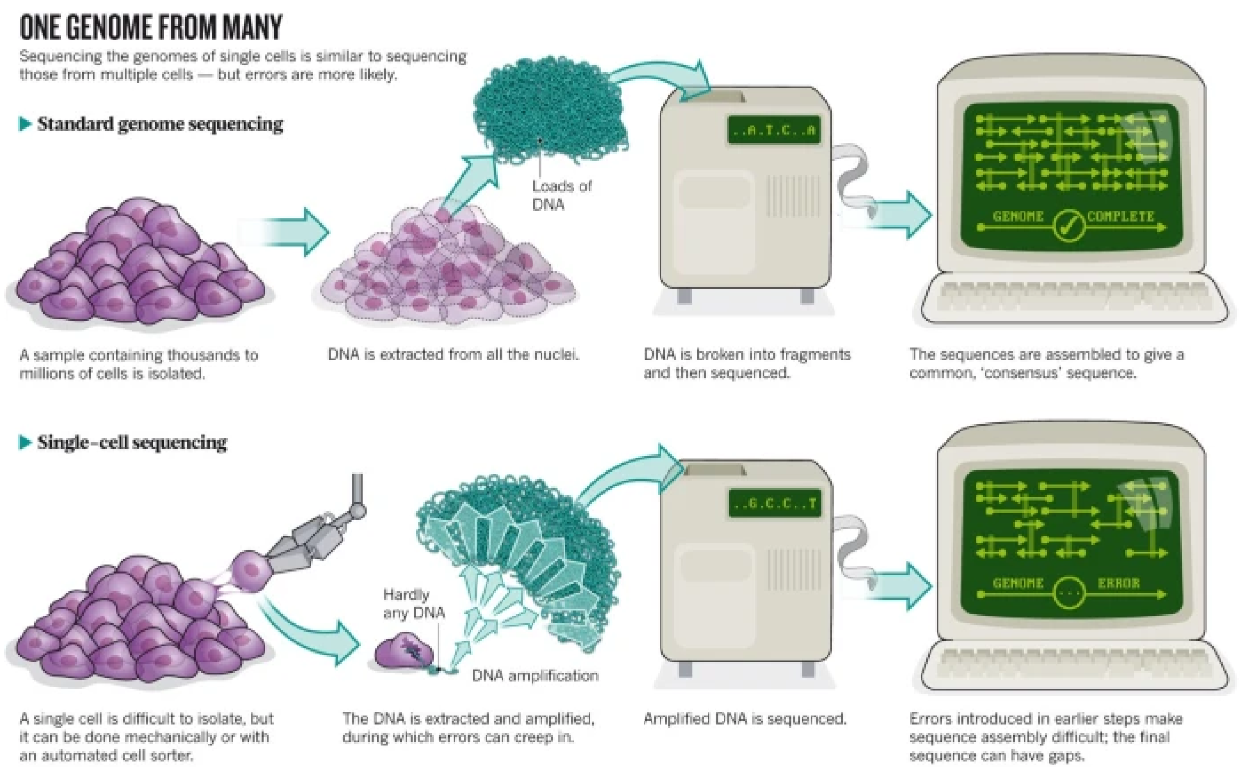

ONE GENOME FROM MANY
Sequencing the genomes of single cells is similar to sequencing those from multiple cells — but errors are more likely.
Standard genome sequencing
Loads of DNA
..A.T.C..A
GENOME COMPLETE
A sample containing thousands to millions of cells is isolated.
DNA is extracted from all the nuclei.
DNA is broken into fragments and then sequenced.
The sequences are assembled to give a common, 'consensus' sequence.
Single-cell sequencing
Hardly any DNA
DNA amplification
..G.C.C..T
GENOME ERROR
A single cell is difficult to isolate, but it can be done mechanically or with an automated cell sorter.
The DNA is extracted and amplified, during which errors can creep in.
Amplified DNA is sequenced.
Errors introduced in earlier steps make sequence assembly difficult; the final sequence can have gaps.

Figure 3: Owens, 2012, doi: 10.1038/491027a

1.3.2 Variability in data

1.4 scRNA-seq Protocols

- Full-length protocols (typically plate-based). Here, a count has a similar interpretation to bulk RNA-seq: the number of sequenced reads that mapped back to that genomic locus. The count is a proxy for the concentration of mRNA molecules in the cell.
- UMI-based protocols (typically droplet-based). Here, one count corresponds to one observed mRNA molecule present in the cell. Thanks to the use of UMI barcodes, PCR artifacts are reduced. However, no full length information on transcript because with fragmentation based protocols only one fragment would have the UMI!

Both protocols suffer from a low 'capture frequency', i.e., the number of molecules present in the cell that have actually been captured and may therefore be observed.

1.5 Multiplexing

Experimental Design!!!

```
eh <- ExperimentHub()
```

```
## snapshotDate(): 2022-04-26
```

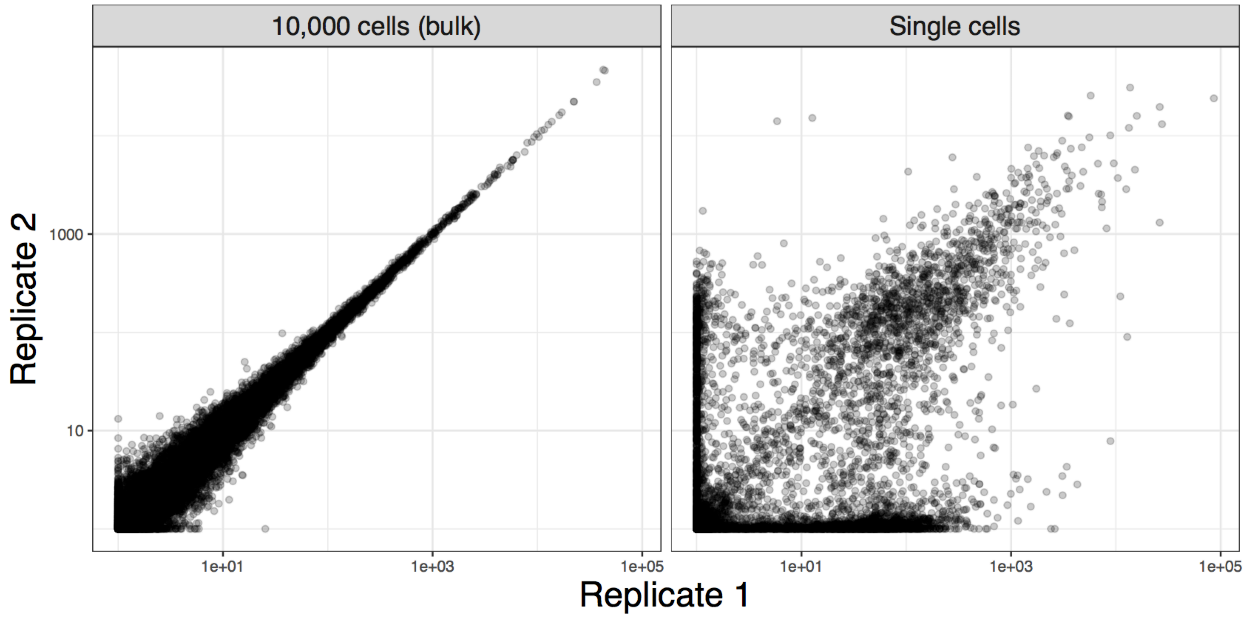


Figure 4: Image credit: Davide Risso

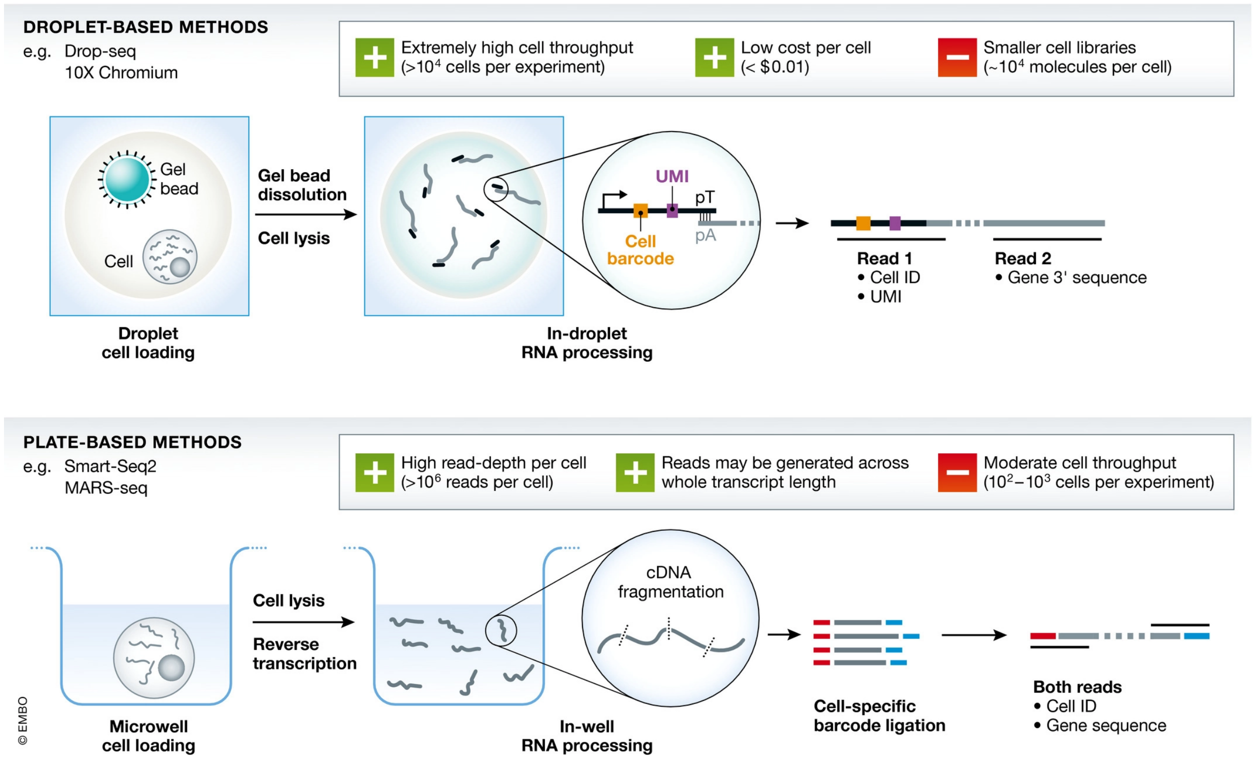

DROPLET-BASED METHODS
e.g. Drop-seq
10X Chromium
+ Extremely high cell throughput (>10^4 cells per experiment)
+ Low cost per cell (< $0.01)
− Smaller cell libraries (~10^4 molecules per cell)
Gel bead
Cell
Gel bead dissolution
Cell lysis
UMI
pT
pA
Cell barcode
Read 1
• Cell ID
• UMI
Read 2
• Gene 3' sequence
Droplet cell loading
In-droplet RNA processing
PLATE-BASED METHODS
e.g. Smart-Seq2
MARS-seq
+ High read-depth per cell (>10^6 reads per cell)
+ Reads may be generated across whole transcript length
− Moderate cell throughput (10^2–10^3 cells per experiment)
Cell lysis
Reverse transcription
cDNA fragmentation
Cell-specific barcode ligation
Both reads
• Cell ID
• Gene sequence
Microwell cell loading
In-well RNA processing
© EMBO

Figure 5: Griffiths et al., 2018, doi.org/10.15252/msb.20178046

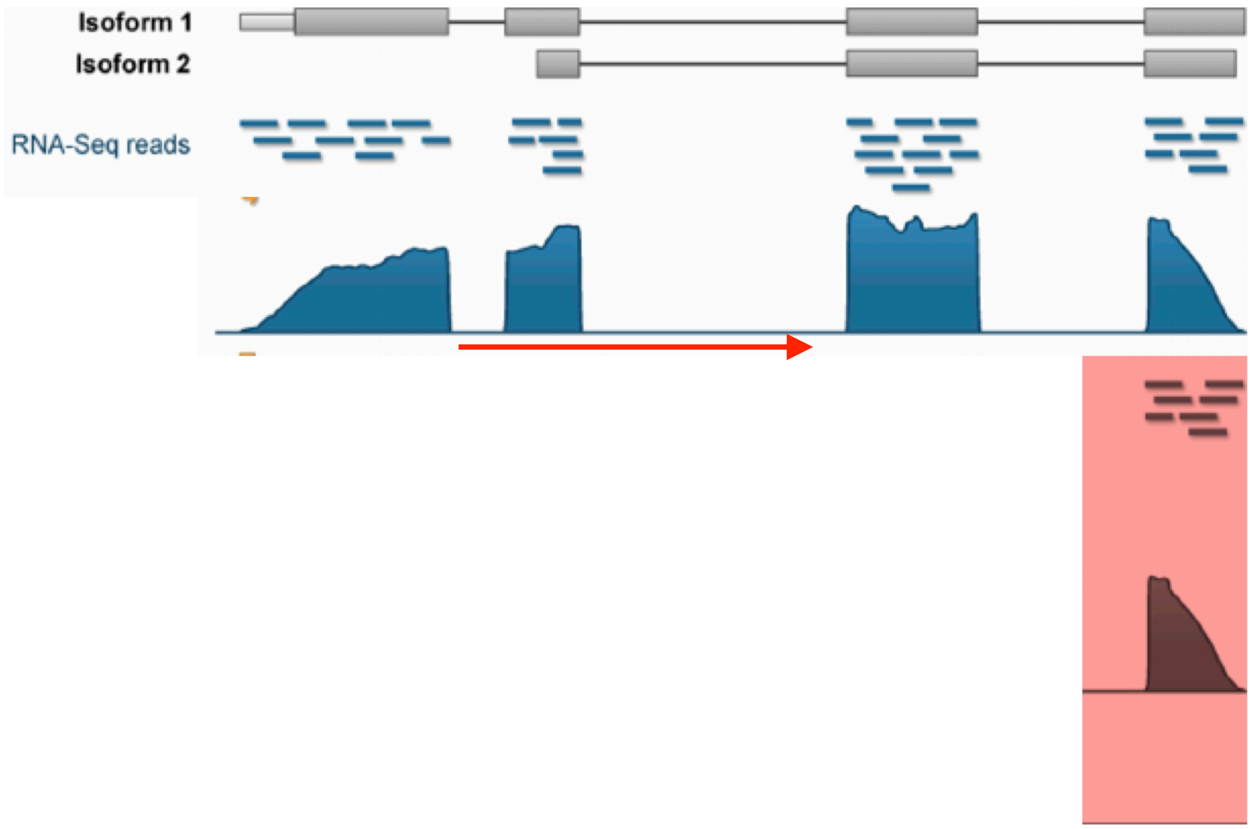


Figure 6: Image credit: Davide Risso

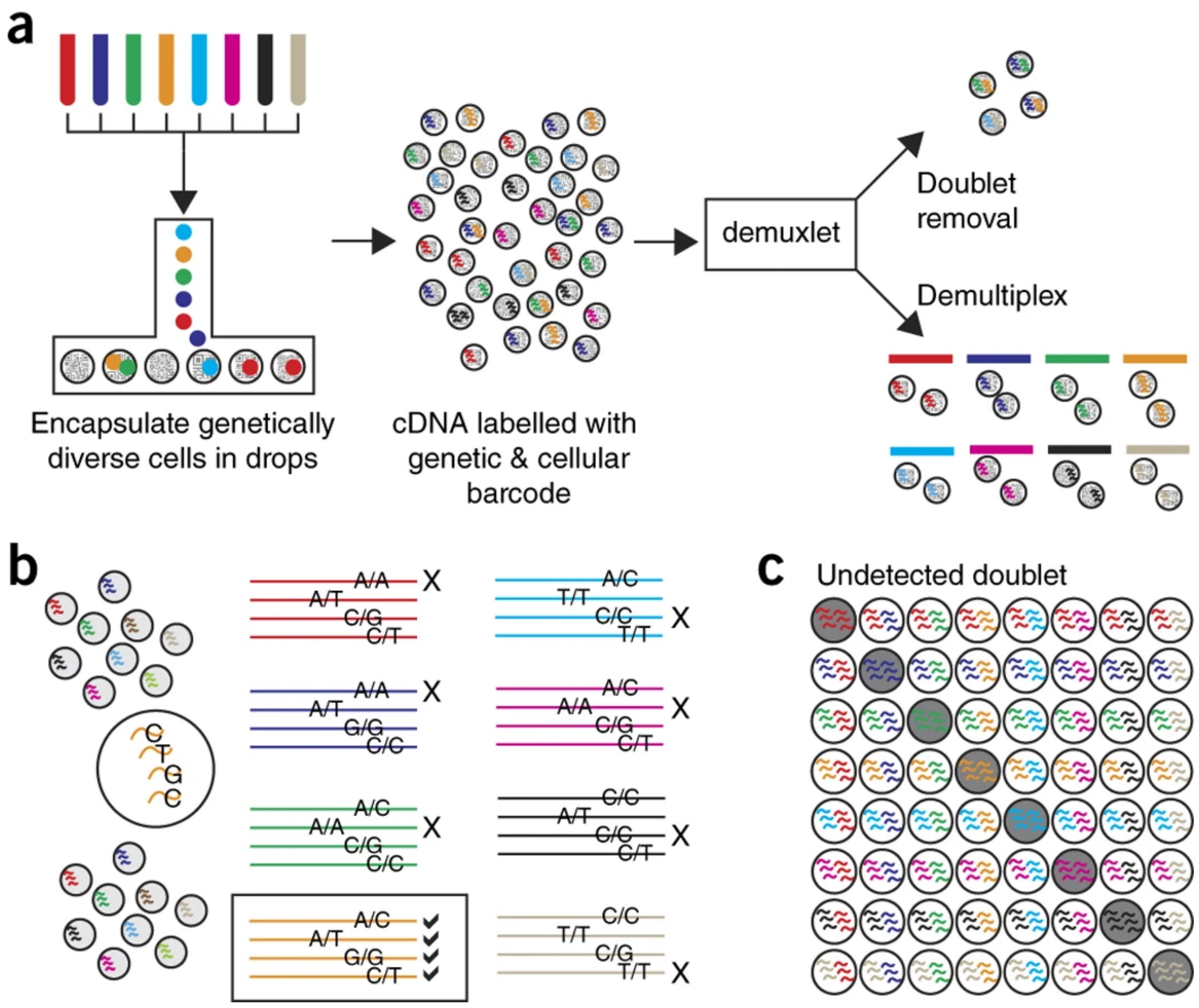

a
Encapsulate genetically diverse cells in drops
cDNA labelled with genetic & cellular barcode
demuxlet
Doublet removal
Demultiplex
b
A/A
A/T
C/G
C/T
G/G
C/C
A/C
T/T
C
T
G
C
c
Undetected doublet

Figure 7: Kang et al. 2018, doi: 10.1038/nbt.4042

```
sce <- eh[["EH2259"]]
```

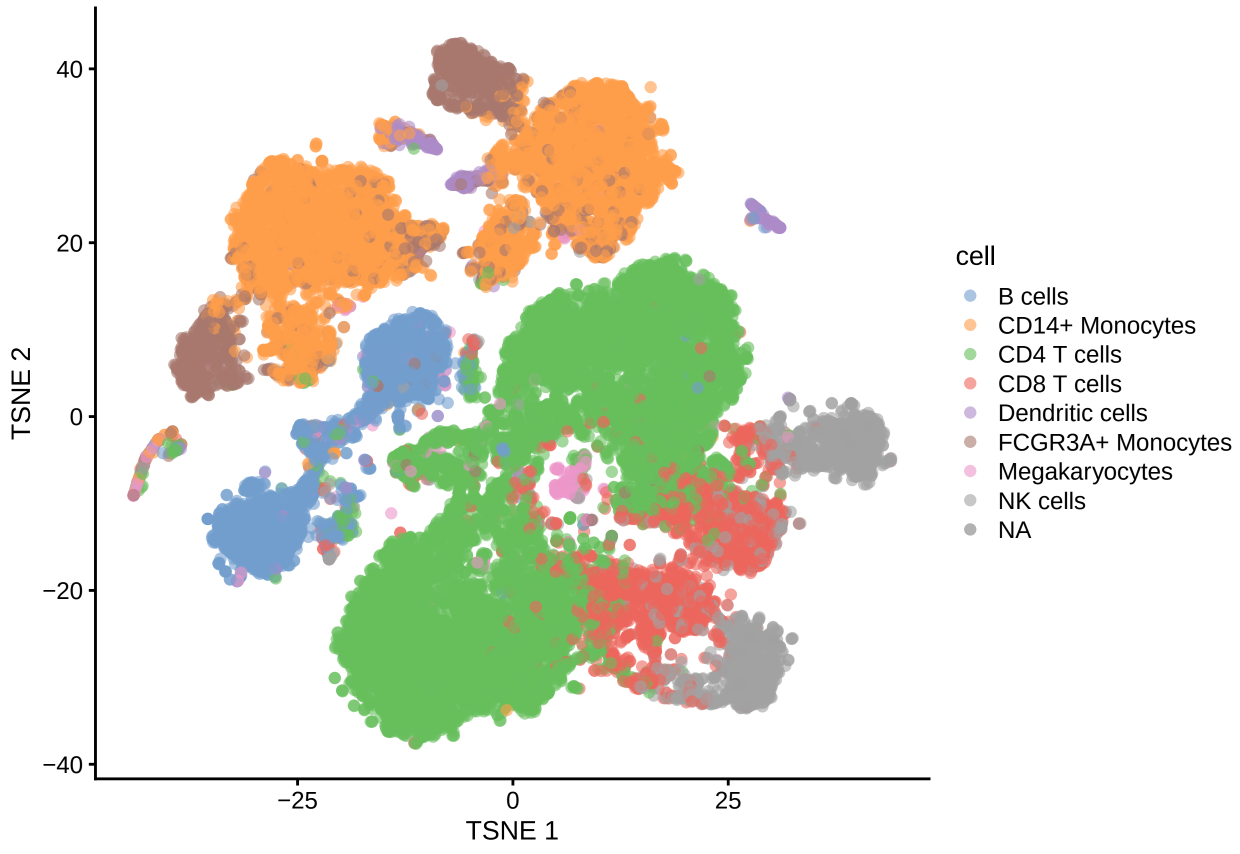
```
## see ?muscData and browseVignettes('muscData') for documentation
```

```
## loading from cache
```

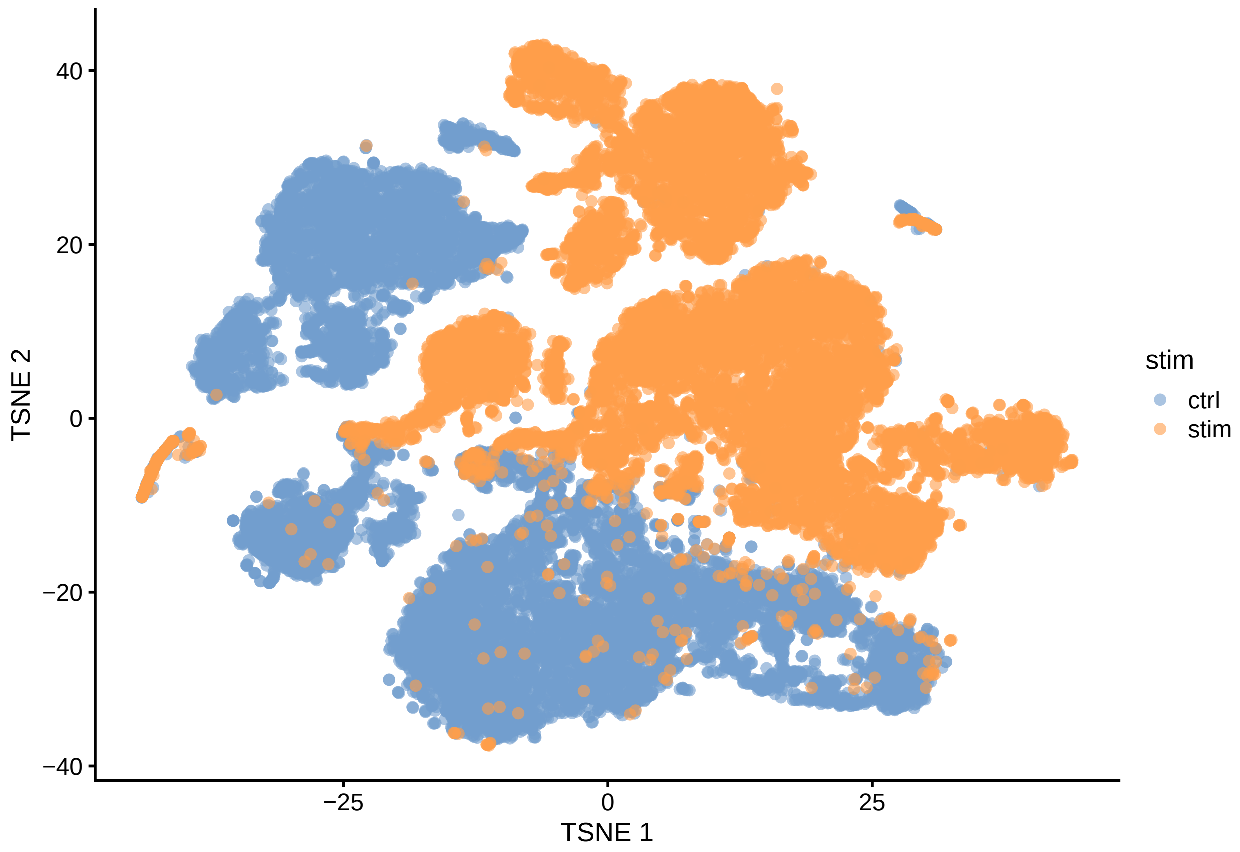
```
sce
```

```
## class: SingleCellExperiment
## dim: 35635 29065
## metadata(0):
## assays(1): counts
## rownames(35635): MIR1302-10 FAM138A ... MT-ND6 MT-CYB
## rowData names(2): ENSEMBL SYMBOL
## colnames(29065): AAACATACAATGCC-1 AAACATACATTTCC-1 ... TTTGCATGGTTTGG-1
##   TTTGCATGTCTTAC-1
## colData names(5): ind stim cluster cell multiplets
## reducedDimNames(1): TSNE
## mainExpName: NULL
## altExpNames(0):
```

```
plotReducedDim(sce, dimred="TSNE", colour_by="cell")
```



```r
plotReducedDim(sce, dimred="TSNE", colour_by="stim")
```



- Note, that we see huge effect of treatment. If we see such large effects we always have to be on our guard!
- Cells of 8 patients were not stimulated or stimulated.
- Cells of different patients could be hashed.
- So all control cells were sequenced in a first run and all stimulated cells were on a second sequencing run.
- So the large effect might be an effect of batch!

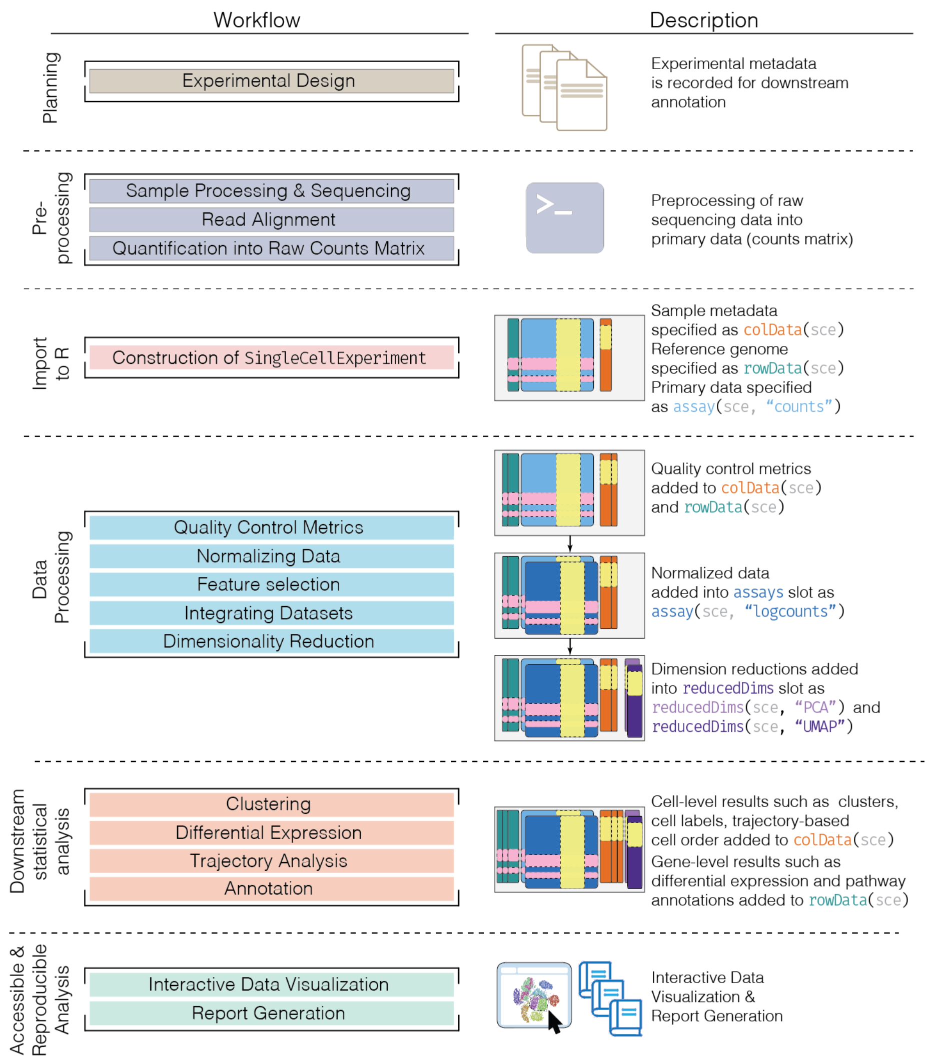
2 Typical Workflow

See OSCA book

2.1 Preprocessing

Additional challenges arise as compared to bulk RNA-seq.

- Cellular barcode identification: Errors occurring during PCR amplification and sequencing can corrupt the cellular barcode sequence.



Workflow
Description
Planning
Experimental Design
Experimental metadata is recorded for downstream annotation
Pre-processing
Sample Processing & Sequencing
Read Alignment
Quantification into Raw Counts Matrix
Preprocessing of raw sequencing data into primary data (counts matrix)
Import to R
Construction of SingleCellExperiment
Sample metadata specified as colData(sce)
Reference genome specified as rowData(sce)
Primary data specified as assay(sce, "counts")
Data Processing
Quality Control Metrics
Normalizing Data
Feature selection
Integrating Datasets
Dimensionality Reduction
Quality control metrics added to colData(sce) and rowData(sce)
Normalized data added into assays slot as assay(sce, "logcounts")
Dimension reductions added into reducedDims slot as reducedDims(sce, "PCA") and reducedDims(sce, "UMAP")
Downstream statistical analysis
Clustering
Differential Expression
Trajectory Analysis
Annotation
Cell-level results such as clusters, cell labels, trajectory-based cell order added to colData(sce)
Gene-level results such as differential expression and pathway annotations added to rowData(sce)
Accessible & Reproducible Analysis
Interactive Data Visualization
Report Generation
Interactive Data Visualization & Report Generation

Figure 8: Image credit: OSCA book

  - Alevin-fry only quantifies cells reliably assigned as non-empty droplets by looking for the "knee" inflection point in the CDF of the total number of UMIs for each droplet.
  - Once this list of barcodes is generated, reads with cellular barcodes not in this set are "corrected" against it by checking if they are within one edit of the set.

- Cell demultiplexing:

  - Cell Hashing: Cellular barcode identification from HTO counts.
  - Biological variation: Model-based clustering of cells to individuals.

- Processing thousands-millions of cells

- UMI identification and collapsing:

  - Determine the set of genes corresponding to all reads of a particular UMI.
  - Often, a UMI count is assigned to the gene with highest frequency amongst all reads having that UMI. Also here, UMI barcodes may become corrupted.

2.2 Single cell experiment class

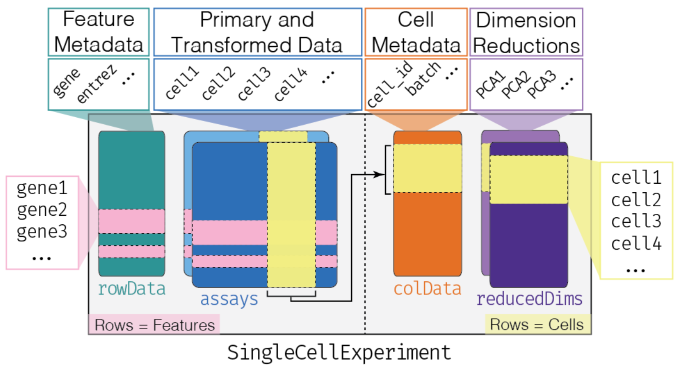


Figure 9: Image credit: OSCA book

Why use a SingleCellExperiment object?

- Convenient framework to store all essential (meta)data in one object.
- All different (meta)data structures are linked: avoids confusion on whether, e.g., the cluster labels and the count matrix are identically sorted.
- Filtering of genes or cells immediately also filters the corresponding metadata.
- Most Bioconductor software accepts a SingleCellExperiment object as input, and outputs a SingleCell-Experiment object $ rightarrow$ one common infrastructure across entire analysis workflow.
- Please see the OSCA chapter on SingleCellExperiment class for more information and how to use it (please try and get familiar!).

2.2.1 Example

We illustrate these steps using a dataset on Peripheral Blood Mononuclear Cell (PBMC) data from a healthy donor provided by 10X Genomics. We use the DropletTestFiles Bioconductor package to download the raw (i.e., unfiltered) count matrix that contains the UMI counts of all genes in all droplets.

```r
raw.path <- getTestFile("tenx-2.1.0-pbmc4k/1.0.0/raw.tar.gz")
```

```
## snapshotDate(): 2022-04-26
```

```
## see ?DropletTestFiles and browseVignettes('DropletTestFiles') for documentation
```

```
## loading from cache
```

```r
out.path <- file.path(tempdir(), "pbmc4k")
untar(raw.path, exdir=out.path)

fname <- file.path(out.path, "raw_gene_bc_matrices/GRCh38")
sce <- read10xCounts(fname, col.names=TRUE)
sce
```

```
## class: SingleCellExperiment
## dim: 33694 737280
## metadata(1): Samples
## assays(1): counts
## rownames(33694): ENSG00000243485 ENSG00000237613 ... ENSG00000277475
##   ENSG00000268674
## rowData names(2): ID Symbol
## colnames(737280): AAACCTGAGAAACCAT-1 AAACCTGAGAAACCGC-1 ...
##   TTTGTCATCTTTAGTC-1 TTTGTCATCTTTCCTC-1
## colData names(2): Sample Barcode
## reducedDimNames(0):
## mainExpName: NULL
## altExpNames(0):
```

The read10xCounts function starts from the output of the Cell Ranger software and imports the data into R as an object of class SingleCellExperiment.

We can notice that the dimension of the matrix is very big; in fact this matrix includes the UMI that have been detected in all the droplets that have been sequenced, including the empty droplets that may contain only ambient RNA.

This is a very sparse matrix, with a large fraction of zeros; read10xCounts is aware of this and stores the counts as a sparse matrix, which has a very small memory footprint.

2.2.2 Explore SingleCellExperiment object

```r
sce %>%
  counts %>%
  head(n=c(6,6))
```

```
## 6 x 6 sparse Matrix of class "dgCMatrix"
##                 AAACCTGAGAAACCAT-1 AAACCTGAGAAACCGC-1 AAACCTGAGAAACCTA-1
## ENSG00000243485                  .                  .                  .
## ENSG00000237613                  .                  .                  .
## ENSG00000186092                  .                  .                  .
## ENSG00000238009                  .                  .                  .
## ENSG00000239945                  .                  .                  .
## ENSG00000239906                  .                  .                  .
##                 AAACCTGAGAAACGAG-1 AAACCTGAGAAACGCC-1 AAACCTGAGAAAGTGG-1
## ENSG00000243485                  .                  .                  .
## ENSG00000237613                  .                  .                  .
## ENSG00000186092                  .                  .                  .
## ENSG00000238009                  .                  .                  .
## ENSG00000239945                  .                  .                  .
## ENSG00000239906                  .                  .                  .
```

```
sce %>%
  counts %>%
  class
```

```
## [1] "dgCMatrix"
## attr(,"package")
## [1] "Matrix"
```

```
sce %>%
  assays
```

```
## List of length 1
## names(1): counts
```

```
sce %>%
  rowData %>%
  head
```

```
## DataFrame with 6 rows and 2 columns
##                              ID        Symbol
##                     <character>   <character>
## ENSG00000243485 ENSG00000243485  RP11-34P13.3
## ENSG00000237613 ENSG00000237613       FAM138A
## ENSG00000186092 ENSG00000186092         OR4F5
## ENSG00000238009 ENSG00000238009  RP11-34P13.7
## ENSG00000239945 ENSG00000239945  RP11-34P13.8
## ENSG00000239906 ENSG00000239906 RP11-34P13.14
```

```
sce %>%
  colData %>%
  head
```

```
## DataFrame with 6 rows and 2 columns
##                                    Sample            Barcode
##                               <character>        <character>
## AAACCTGAGAAACCAT-1 /tmp/RtmpqbhPBi/pbmc.. AAACCTGAGAAACCAT-1
## AAACCTGAGAAACCGC-1 /tmp/RtmpqbhPBi/pbmc.. AAACCTGAGAAACCGC-1
## AAACCTGAGAAACCTA-1 /tmp/RtmpqbhPBi/pbmc.. AAACCTGAGAAACCTA-1
## AAACCTGAGAAACGAG-1 /tmp/RtmpqbhPBi/pbmc.. AAACCTGAGAAACGAG-1
## AAACCTGAGAAACGCC-1 /tmp/RtmpqbhPBi/pbmc.. AAACCTGAGAAACGCC-1
## AAACCTGAGAAAGTGG-1 /tmp/RtmpqbhPBi/pbmc.. AAACCTGAGAAAGTGG-1
```

2.2.3 Use sensible gene names

Before starting the analysis, it may be a good idea to store the names of the genes in a more human-friendly ID system. We can also include information on the chromosome location of the genes; this will be useful for e.g. identifying mitochondrial genes.

```
library(scuttle)
rownames(sce) <- uniquifyFeatureNames(
    rowData(sce)$ID, rowData(sce)$Symbol)

rowData(sce)$location <- mapIds(EnsDb.Hsapiens.v86,
                                     keys=rowData(sce)$ID,
                                     column="SEQNAME", keytype="GENEID")
```

```
## Warning: Unable to map 144 of 33694 requested IDs.
```

```
rowData(sce)$location[is.na(rowData(sce)$location)] <- "NA"
rowData(sce)
```

```
## DataFrame with 33694 rows and 3 columns
##                            ID       Symbol    location
##                   <character>  <character> <character>
## RP11-34P13.3 ENSG00000243485 RP11-34P13.3           1
## FAM138A      ENSG00000237613      FAM138A           1
## OR4F5        ENSG00000186092        OR4F5           1
## RP11-34P13.7 ENSG00000238009 RP11-34P13.7           1
## RP11-34P13.8 ENSG00000239945 RP11-34P13.8           1
## ...                      ...          ...         ...
## AC233755.2   ENSG00000277856   AC233755.2  KI270726.1
## AC233755.1   ENSG00000275063   AC233755.1  KI270726.1
## AC240274.1   ENSG00000271254   AC240274.1  KI270711.1
## AC213203.1   ENSG00000277475   AC213203.1  KI270713.1
## FAM231B      ENSG00000268674      FAM231B  KI270713.1
```

2.3 Filter non-informative genes

```
keep <- rowSums(assays(sce)$counts > 0) > 10
table(keep)
```

FALSE	TRUE
19706	13988

```
sce <- sce[keep,]
```

2.4 Detection and removal of empty droplets

The first step is the identification of droplets that do not contain any live cell.

The reason why these droplets contain some RNA is that there may be some ambient RNA due to some cell leaking or they may contain dead or dying cells.

2.4.1 Example

The barcodeRanks function can be used to rank the barcodes by number of UMIs and to estimate the knee and inflection point of the distribution.

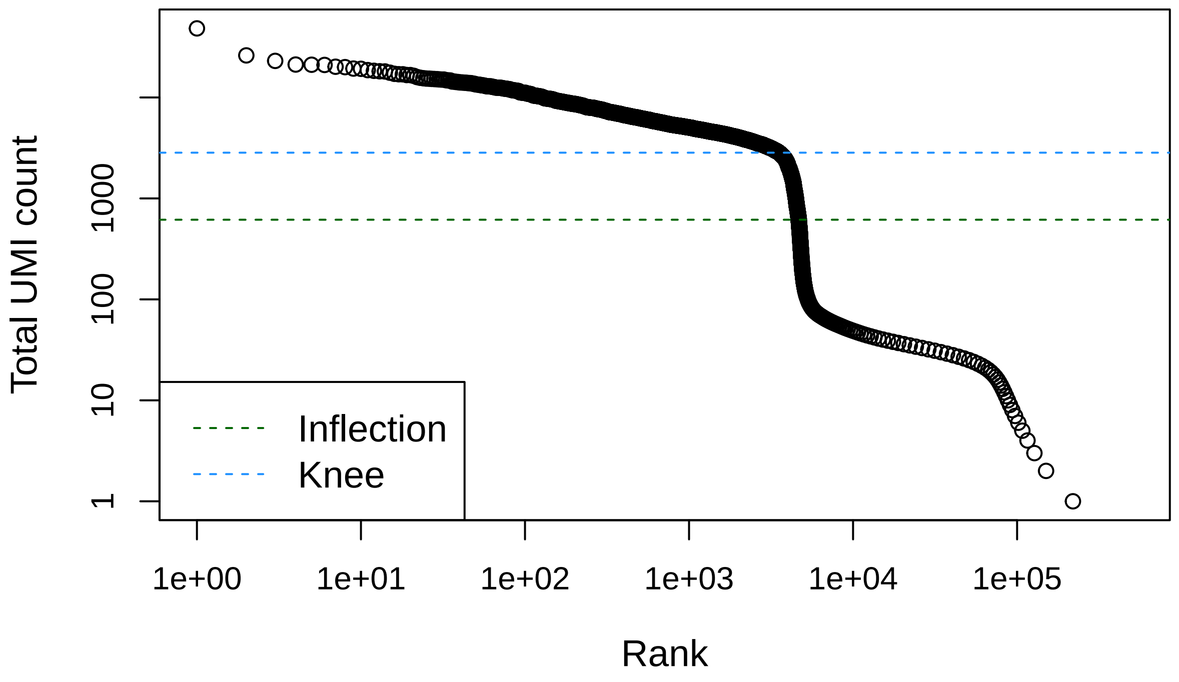
```
bcrank <- barcodeRanks(counts(sce))

# Only showing unique points for plotting speed.
uniq <- !duplicated(bcrank$rank)
plot(bcrank$rank[uniq], bcrank$total[uniq], log="xy",
    xlab="Rank", ylab="Total UMI count", cex.lab=1.2)
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 1 y value <= 0 omitted from
## logarithmic plot
```

```
abline(h=metadata(bcrank)$inflection, col="darkgreen", lty=2)
abline(h=metadata(bcrank)$knee, col="dodgerblue", lty=2)

legend("bottomleft", legend=c("Inflection", "Knee"),
        col=c("darkgreen", "dodgerblue"), lty=2, cex=1.2)
```



- There is a sharp distinction between droplets with very high counts, very likely to contain a live cell, and droplets with very low counts, very likely to be empty.
- However, it is not straightforward to classify the droplets in the middle of the distribution.

- We can apply a statistical test of hypothesis to decide, for each droplet, if its RNA profile is significantly different from the profile of ambient RNA, estimated from the very low counts (Aaron TL Lun et al. 2019).
- We use a very low threshold on the False Discovery Rate to have very few false positive cells.

```
set.seed(100)
e.out <- emptyDrops(counts(sce))
summary(e.out$FDR <= 0.001)
```

Mode	FALSE	TRUE	NA's
logical	988	4300	731992

The large majority of droplets are not tested, since by default all droplets with fewer than 100 UMIs are considered empty.

```
table(colSums(counts(sce))>100, e.out$FDR<=0.001, useNA = "ifany")
```

/	FALSE	TRUE	NA
FALSE	0	0	731992
TRUE	988	4300	0

We can now proceed by removing the empty droplets and keep only the ones identified to be cells.

```
sce <- sce[,which(e.out$FDR <= 0.001)]
sce
```

```
## class: SingleCellExperiment
## dim: 13988 4300
## metadata(1): Samples
## assays(1): counts
## rownames(13988): FO538757.2 AP006222.2 ... AC004556.1 AC240274.1
## rowData names(3): ID Symbol location
## colnames(4300): AAACCTGAGAAGGCCT-1 AAACCTGAGACAGACC-1 ...
##   TTTGTCAGTTAAGACA-1 TTTGTCATCCCAAGAT-1
## colData names(2): Sample Barcode
## reducedDimNames(0):
## mainExpName: NULL
## altExpNames(0):
```

2.5 Quality control

Some cells have poor quality (e.g., damaged or stressed cells).

- Expression of mitochondrial genes, total number of genes expressed, read distribution across genes and library size are good indicators for a cell's quality.

2.5.1 Concerns

One concern is that we would inadvertently remove, e.g., an entire cell type from the downstream analysis. QC variables may not be independent of biological state. There is a trade-off between only working with high-quality cells and risk of removing biological signal.

2.5.2 Criteria

- A high-quality cell should have
  - A high number of genes expressed
  - A fair library size
  - No major expression of mitochondrial genes, relative to other genes
- Based on this information, we could set a fixed threshold, e.g., "keep cells with 4000 genes expressed, a library size of at least 10000 and fewer than 5% of the total output coming from mitochondrial genes".
  - Requires substantial experience with the biological system and scRNA-seq data, e.g., tumors typically express a relatively higher amount of mitochondrial genes.
- A better approach is to use an adaptive threshold.
  - Again assumes most cells are high-quality cells $\rightarrow$ identification of outliers allows identification of low-quality cells.
  - For example, for removing low library size cells in the OSCA book they use

$$I(N_i < median(\mathbf{N}) - 3 \times MAD(\mathbf{N}))$$

with N_i the library size of cell i and $\mathbf{N} = [N_1, \dots, N_m]$ with m the number of cells.

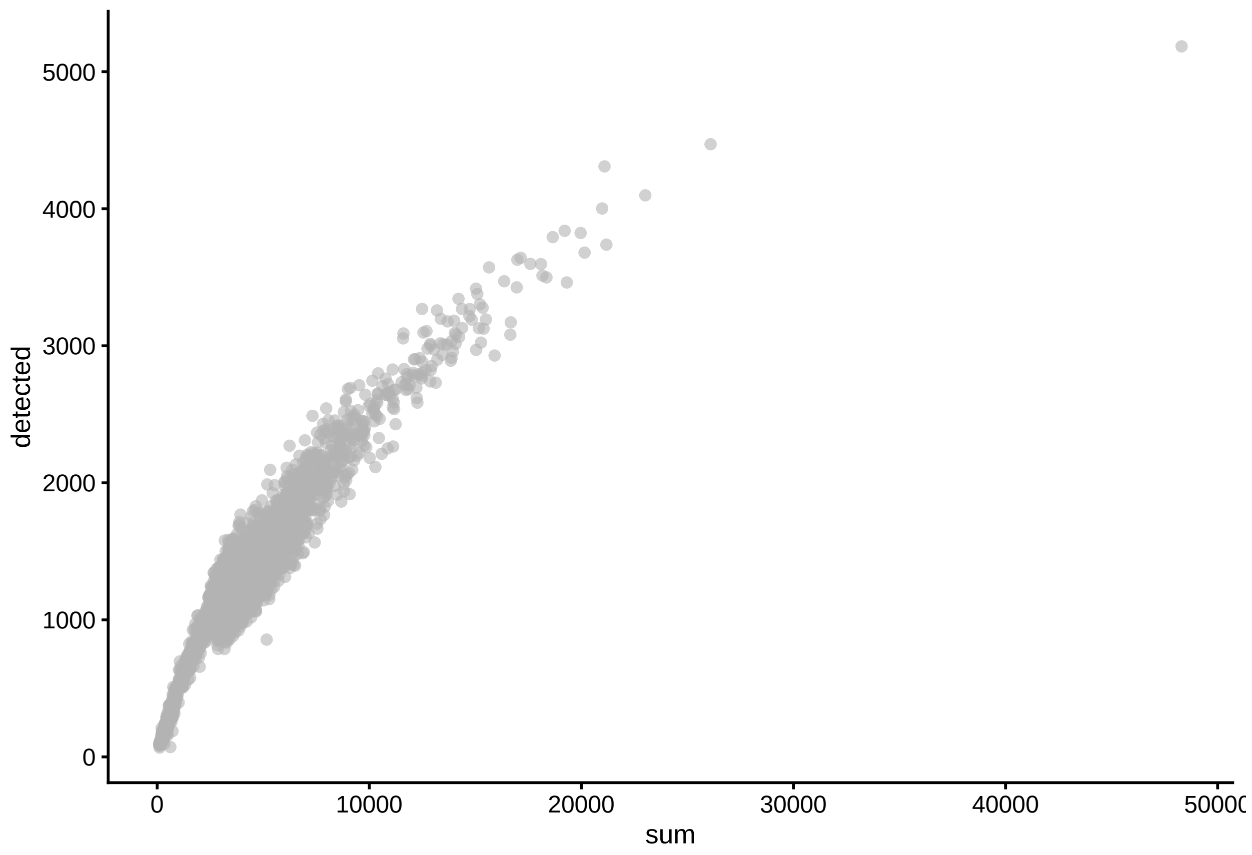
2.5.3 Example

The `perCellQCMetrics` function can be used to compute a set of metrics useful to evaluate the quality of the samples. The `isOutlier` function uses a data driven threshold to define cells of lower quality compared to the rest of the dataset.

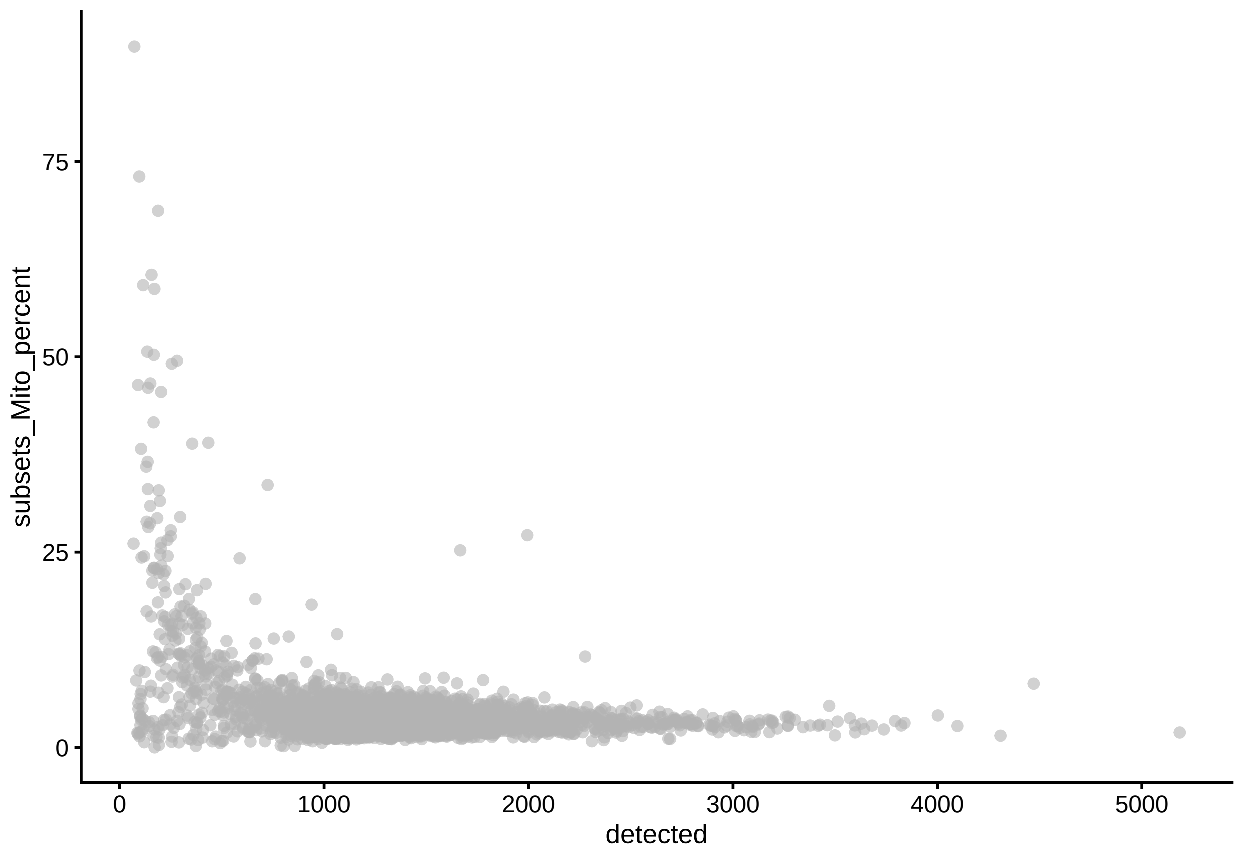
```r
stats <- perCellQCMetrics(sce,
            subsets=list(Mito=which(rowData(sce)$location=="MT")))
```

High-quality cells should have many features expressed, and a low contribution of mitochondrial genes. Here, we see that several cells have a very low number of expressed genes, and where most of the molecules are derived from mitochondrial genes. This indicates likely damaged cells, presumably because of loss of cytoplasmic RNA from perforated cells, so they have to be removed for the downstream analysis.

```r
colData(sce) <- cbind(colData(sce), stats)
plotColData(sce, x = "sum", y="detected")
```



```
plotColData(sce, x = "detected", y="subsets_Mito_percent")
```



We normally remove cells that are outlying with respect to

1. A low sequencing depth (number of UMIs);
2. A low number of genes detected;
3. A high percentage of reads from mitochondrial genes.

```r
lowFeatures <- isOutlier(stats$detected, type="lower", log=TRUE)
lowLib <- isOutlier(stats$sum, type="lower", log=TRUE)
highMito <- isOutlier(stats$subsets_Mito_percent, type="higher")
table(lowFeatures)
```

FALSE	TRUE
4006	294

```r
table(lowLib)
```

FALSE	TRUE
3973	327

```r
table(highMito)
```

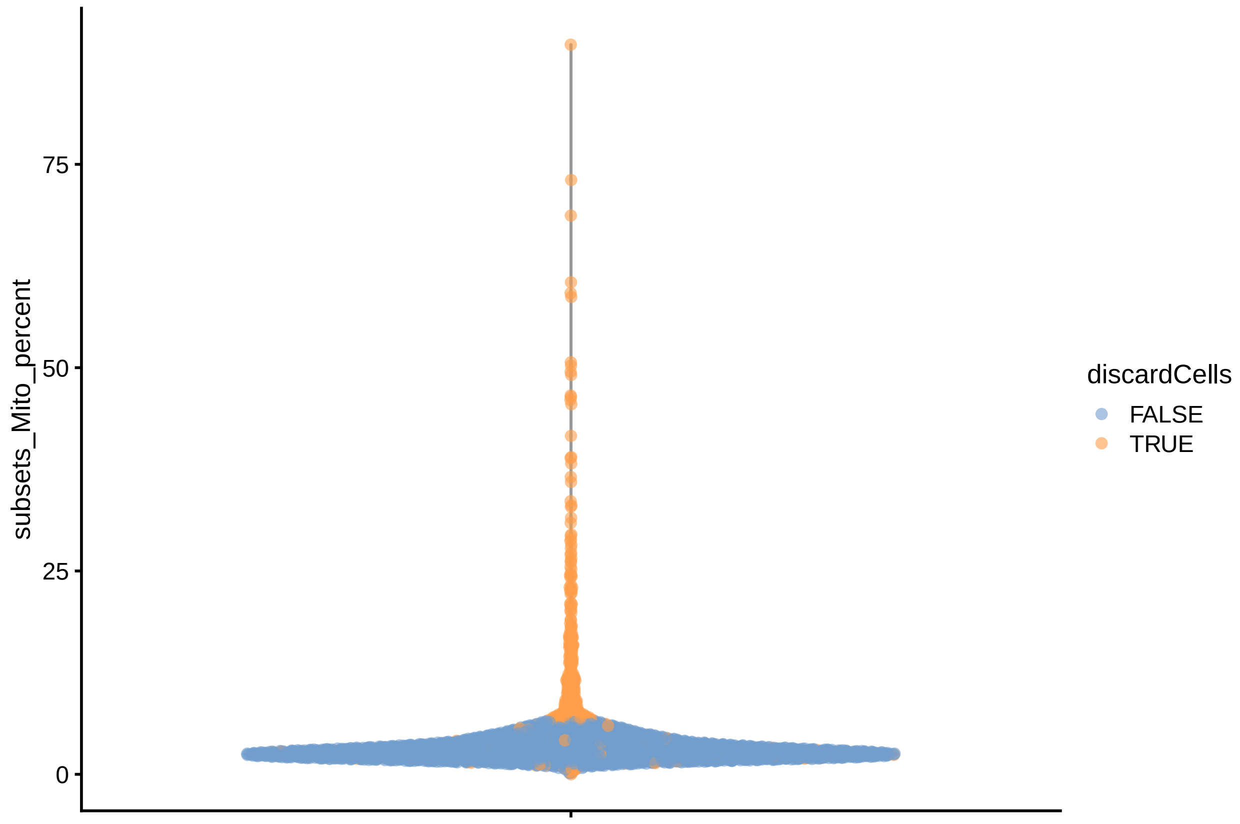
FALSE	TRUE
3990	310

```
discardCells <- (lowLib|lowFeatures|highMito)
table(discardCells)
```

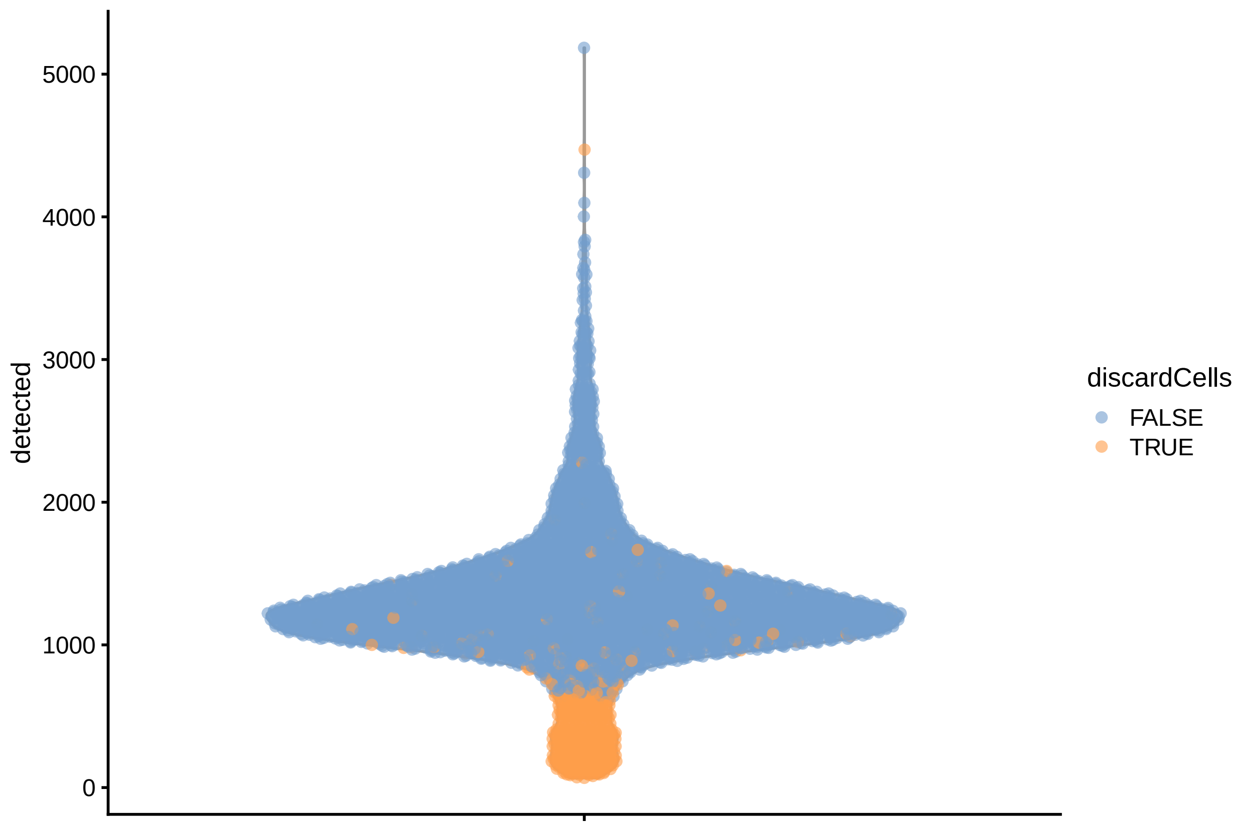
FALSE	TRUE
3864	436

```
colData(sce)$discardCells <- discardCells
```

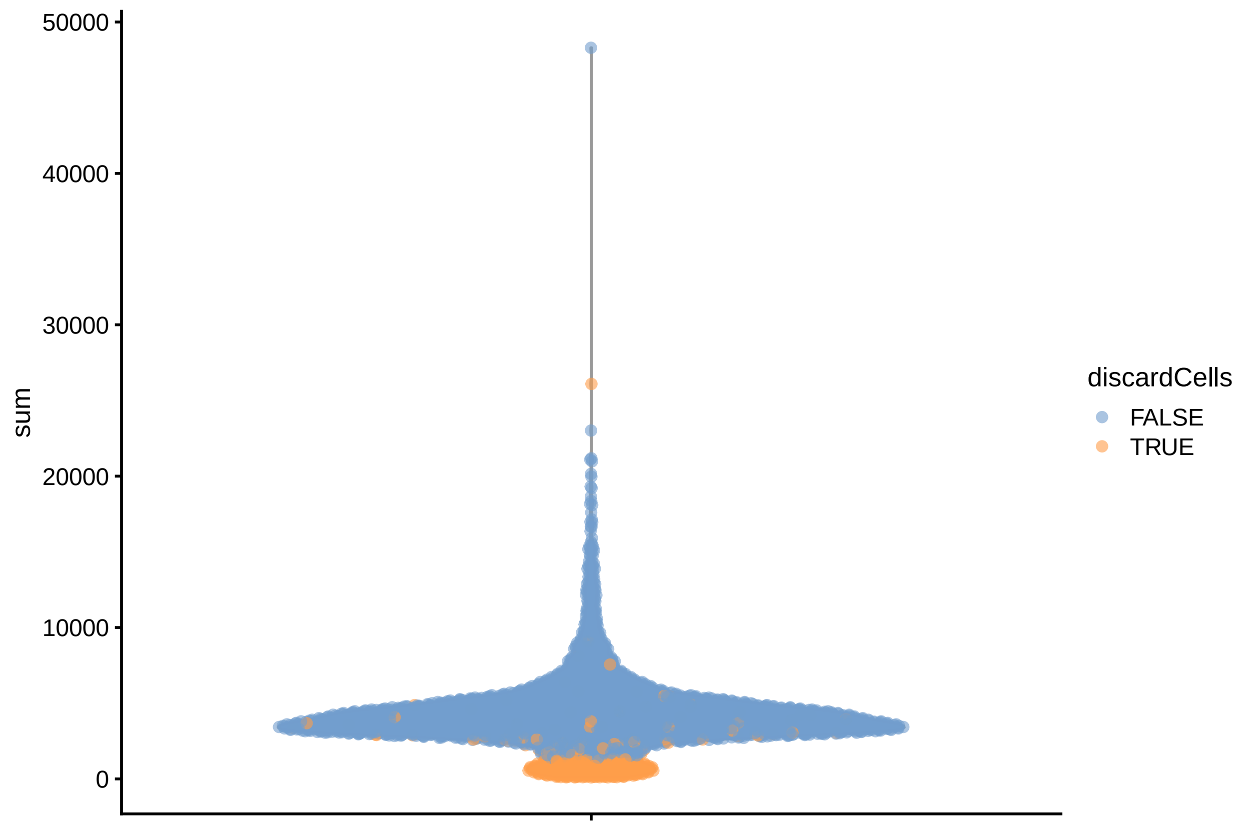
```
scater::plotColData(sce, y="subsets_Mito_percent",
        colour_by="discardCells")
```



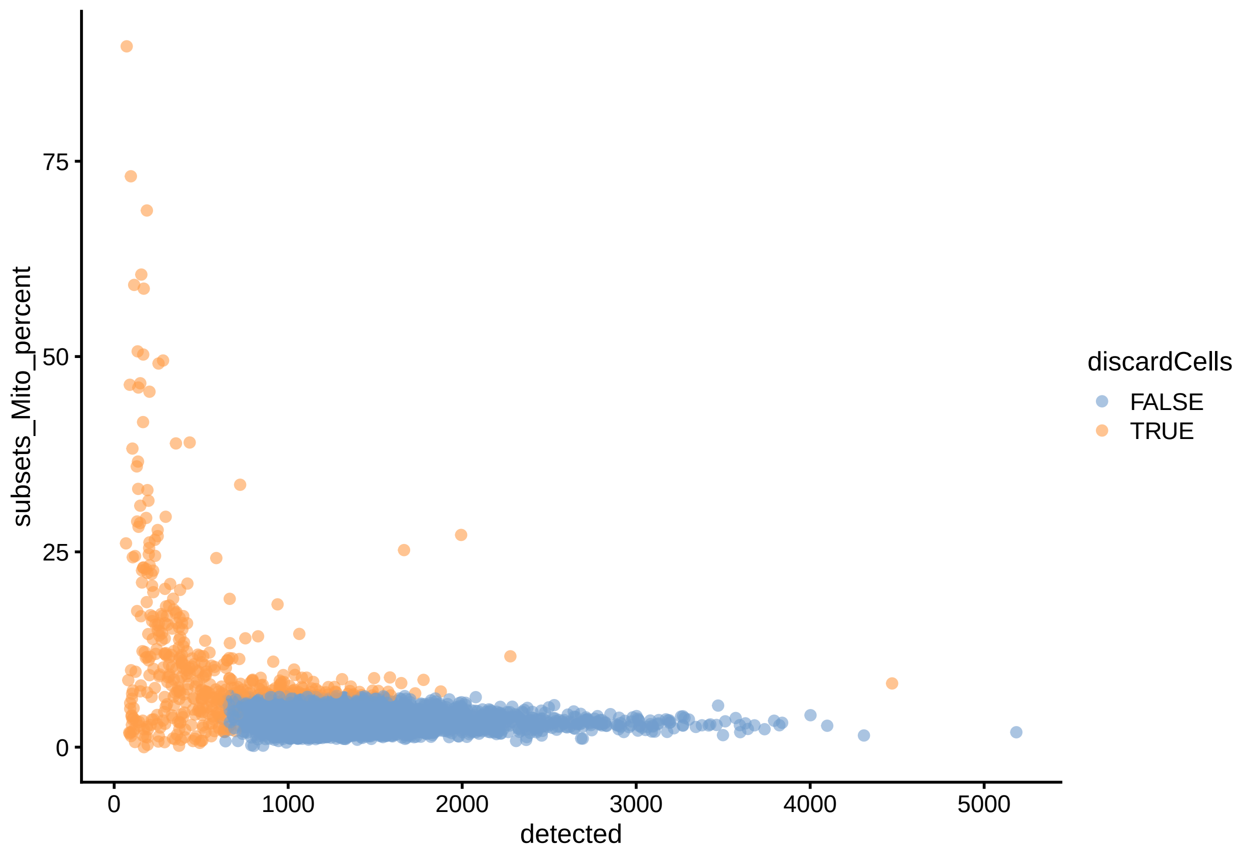
```
scater::plotColData(sce, y="detected",
        colour_by="discardCells")
```



```r
scater::plotColData(sce, y="sum",
              colour_by="discardCells")
```



```r
plotColData(sce, x = "detected", y="subsets_Mito_percent", colour_by = "discardCells")
```



2.6 Detection and removal of doublets

Not all droplets contain a single cell; some contain >1 cell.

Doublet cell clusters may be mistaken for intermediate/transitory states.

2.6.1 Methods

```
knitr::include_graphics("./images_sequencing/scDblFinder.png")
```

1. PCA on log-normalized expression counts.
2. Randomly select two cells, sum their counts and normalize, and project into PCA space.
3. Repeat step 2 many times.
4. For each cell, check its neighborhood to simulated doublets as compared to original cells.
5. Use this information, along with other predictors, to train a classifier for doublets.

2.6.2 Concerns

A potential concern is that we would inadvertently remove biological single cells, that may have e.g. a low transcriptome complexity. There is a trade-off between only working with high-quality cells and risk of removing biological signal.

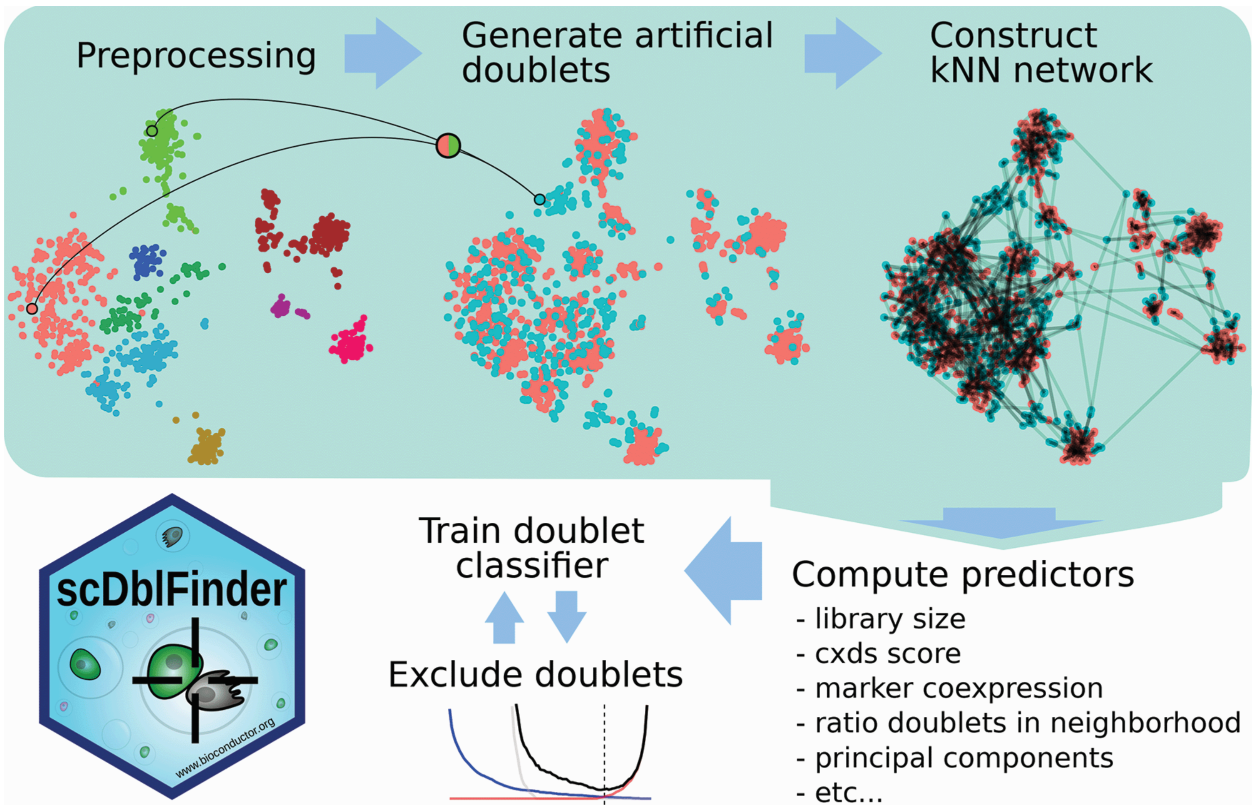


Figure 10: Germain et al. 2021, doi: 10.12688/f1000research.73600.2

2.6.3 Example

```
dbl.dens <- computeDoubletDensity(sce)
sce$DoubletScore <- dbl.dens
summary(dbl.dens)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0	0.1548	0.3182	0.692634	0.7052	20.124

```
dbl.calls <- doubletThresholding(data.frame(score=dbl.dens),
    method="griffiths", returnType="call")
summary(dbl.calls)
```

```
## singlet doublet
##    3651     649
```

```
sce <- sce[, dbl.calls == "singlet"]
sce
```

```
## class: SingleCellExperiment
## dim: 13988 3651
## metadata(1): Samples
## assays(1): counts
## rownames(13988): FO538757.2 AP006222.2 ... AC004556.1 AC240274.1
## rowData names(3): ID Symbol location
## colnames(3651): AAACCTGAGAAGGCCT-1 AAACCTGAGACAGACC-1 ...
##   TTTGTCAGTTAAGACA-1 TTTGTCATCCCAAGAT-1
## colData names(10): Sample Barcode ... discardCells DoubletScore
## reducedDimNames(0):
## mainExpName: NULL
## altExpNames(0):
```

Note that, similarly to what we have said for the removal of low-quality cells, one should be careful in removing putative doublets. Indeed, it may be difficult to distinguish between rare transient cell populations and doublet cells.

2.7 Normalisation

Normalization aims to remove technical effects such as sequencing depth so that comparisons between cells are not confounded by them.

As in bulk RNA-seq, the most commonly used methods are scaling normalizations, where a scaling factor (also called size factor, normalization factor) is estimated for each cell.

2.7.1 Concerns

- Some cell types may naturally have fewer mRNA molecules and therefore a lower library size. Removal of sequencing depth effects across cells therefore also risks removing some biological signal.
- The assumption of a cell-specific scaling factor is that any technical effect it supposedly accounts for, affects all genes equally through a scaling of the mean count. This is not necessarily realistic.
- Scaling the data and log-transforming the data is not a good option. Use offsets!

```
knitr::include_graphics("./images_sequencing/glmPCAFig2.png")
```

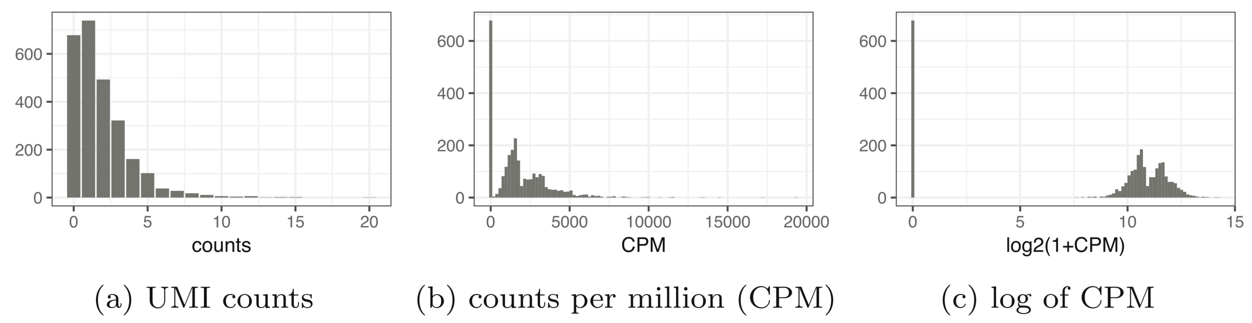


Fig. 2 Example of how current approaches to normalization and transformation artificially distort differences between zero and nonzero counts. **a** UMI count distribution for gene ENSG00000114391 in the monocytes biological replicates negative control dataset. **b** Counts per million (CPM) distribution for the exact same count data. **c** Distribution of $log_2(1 + CPM)$ values for the exact same count data

Figure 11: Townes et al. 2019, DOI: 10.1186/s13059-019-1861-6

2.7.2 Offsets

Let y_{gi} be the expression for gene g in cell i. Assume, as in bulk RNA-seq that

$E[y_{gi}] = \mu_{gi} = \pi_{gi} s_i N_i$

In the edgeR model, $s_i N_i$ is referred to as the 'effective library size', which is used as offset in the GLM, thereby effectively focussing the statistical inference on π_{gi}.

- TMM and DESeq2 normalisation are largely driven by a few genes without zero counts!

Popular, bespoke single-cell normalization methods attempt to alleviate these issues.

2.7.3 Example

Here, we use the scran method to normalize the data for differences in sequencing depth. This approach is based on the deconvolution of size factors estimated from pools of cells (Lun et all. 2016).

Since we have a heterogeneous cell population, we perform a quick clustering to make sure that we pool together cells that are not too different from each other.

```
library(scran)
cl <- quickCluster(sce)
table(cl)
```

1	2	3	4	5	6	7	8	9	10	11
204	203	364	262	547	999	190	233	399	124	126

```
sce <- computeSumFactors(sce, clusters = cl)
```

```
## Warning in (function (x, sizes, min.mean = NULL, positive = FALSE, scaling =
## NULL) : encountered non-positive size factor estimates
```

```
summary(sizeFactors(sce))
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0027868	0.7399026	0.9211244	1	1.148053	5.538119

We can check that the estimated library sizes are not too far from the library size factors, estimated from the total number of counts.

```
plot(librarySizeFactors(sce), sizeFactors(sce), xlab="Library size factor",
    ylab="Deconvolution size factor", log='xy', pch=16,
    col=as.integer(factor(cl)))
abline(a=0, b=1, col="red")
```

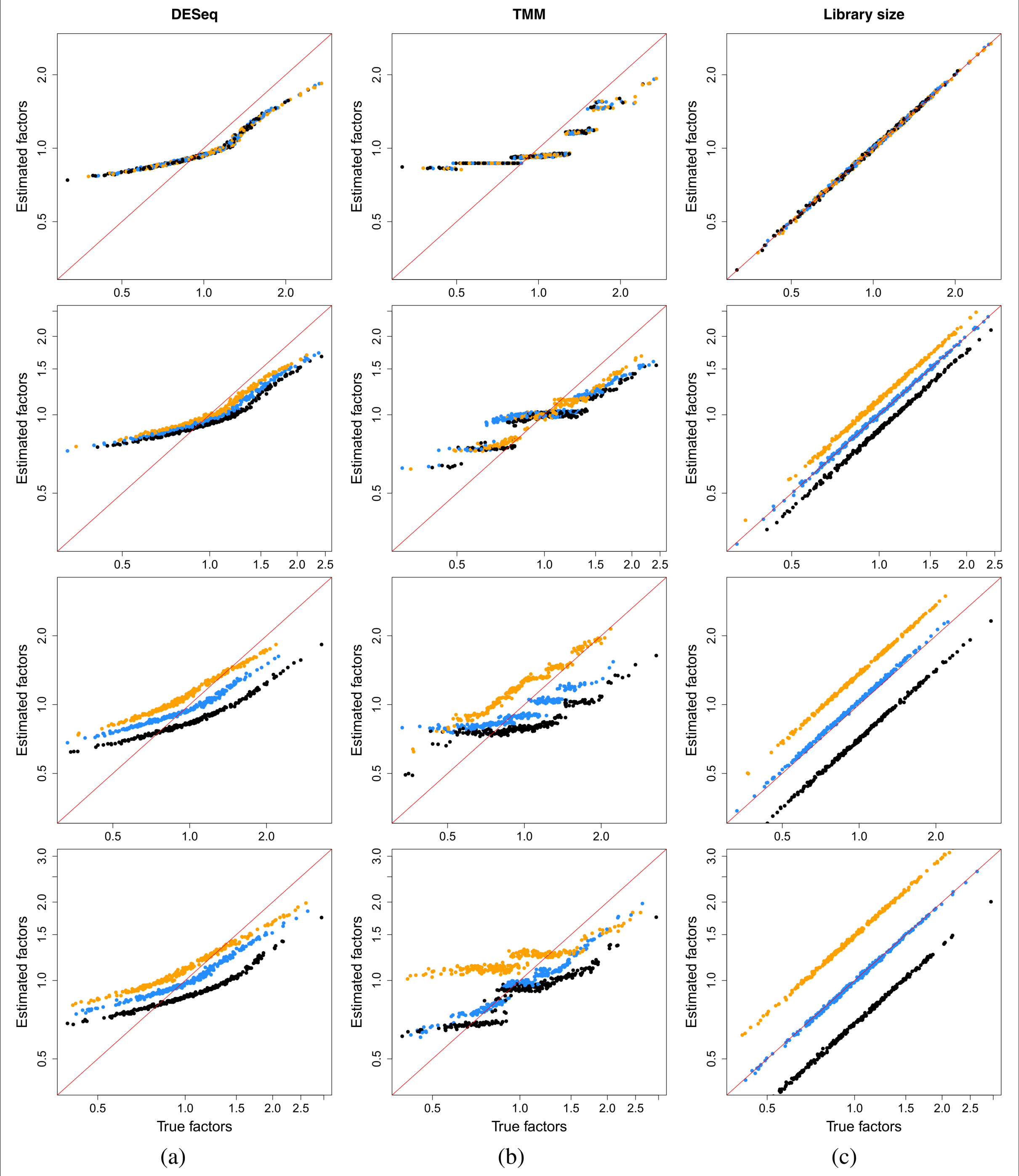
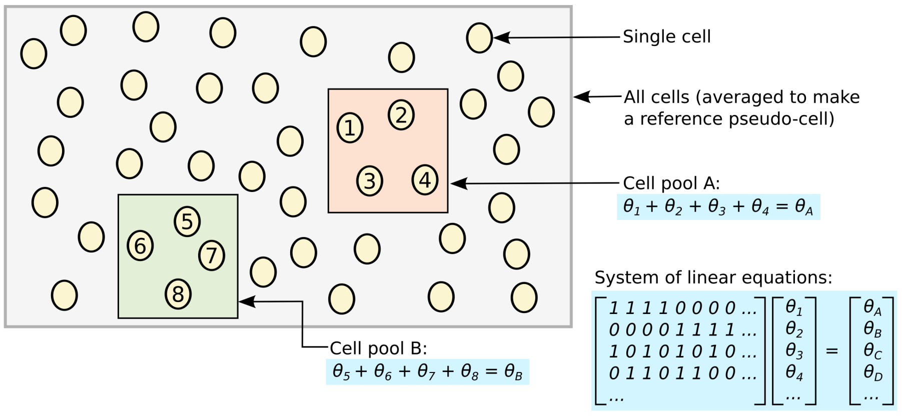


Fig. 1 Performance of existing normalization methods on the simulated data with DE genes and stochastic zeroes. The size factor estimates for all cells are plotted against the true values for **a** DESeq, **b** TMM, and **c** library size normalization. Simulations were performed with no DE (*first row*), moderate DE (*second row*), strong DE (*third row*), and varying magnitudes of DE (*fourth row*). Axes are shown on a log-scale. For comparison, each set of size factors was scaled such that the grand mean across cells was the same as that for the true values. The *red line* represents equality between the rescaled estimates and true factors. Cells in the first, second, and third subpopulations are shown in *black*, *blue*, and *orange*, respectively. *DE* differentially expressed, *TMM* trimmed mean of *M* values

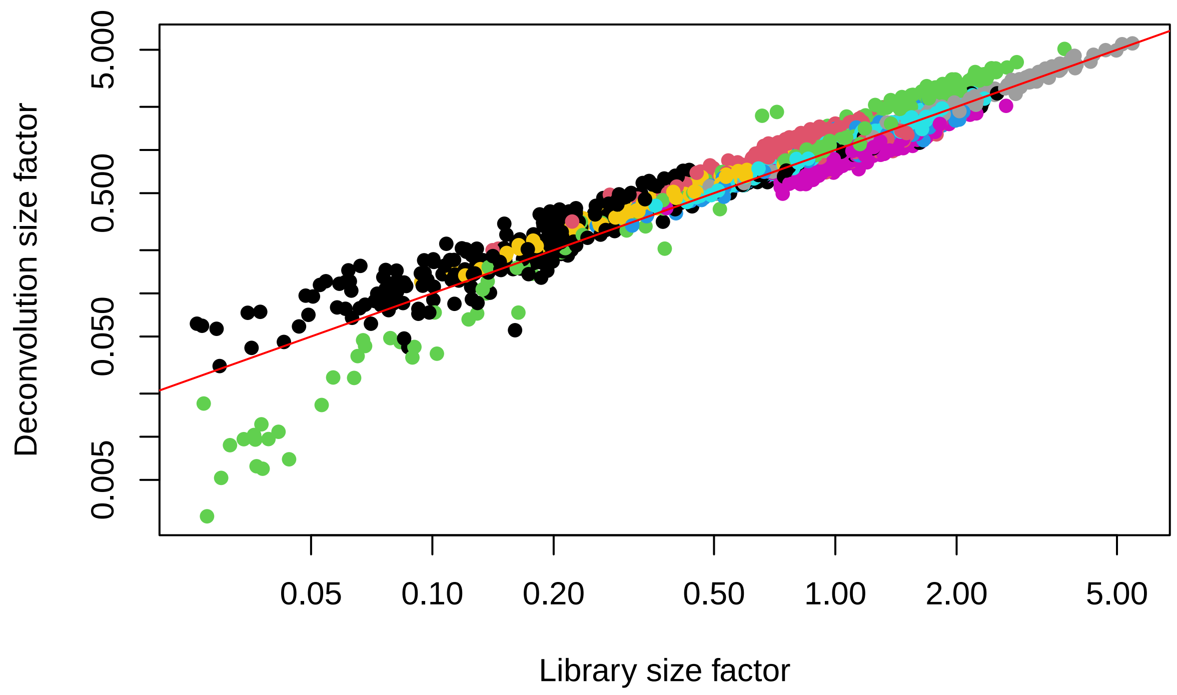
Figure 12: Lun et al. 2016, DOI: 10.1186/s13059-016-0947-7


Single cell
All cells (averaged to make a reference pseudo-cell)
Cell pool A:
$\theta_1 + \theta_2 + \theta_3 + \theta_4 = \theta_A$
System of linear equations:
Cell pool B:
$\theta_5 + \theta_6 + \theta_7 + \theta_8 = \theta_B$

$$\begin{bmatrix} 1\,1\,1\,1\,0\,0\,0\,0 \ldots \\ 0\,0\,0\,0\,1\,1\,1\,1 \ldots \\ 1\,0\,1\,0\,1\,0\,1\,0 \ldots \\ 0\,1\,1\,0\,1\,1\,0\,0 \ldots \\ \ldots \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \ldots \end{bmatrix} = \begin{bmatrix} \theta_A \\ \theta_B \\ \theta_C \\ \theta_D \\ \ldots \end{bmatrix}$$

Fig. 3 Schematic of the deconvolution method. All cells in the data set are averaged to make a reference pseudo-cell. Expression values for cells in pool A are summed together and normalized against the reference to yield a pool-based size factor θ_A. This is equal to the sum of the cell-based factors θ_j for cells $j = 1$–4 and can be used to formulate a linear equation. (For simplicity, the t_j term is assumed to be unity here.) Repeating this for multiple pools (e.g., pool B) leads to the construction of a linear system that can be solved to estimate θ_j for each cell j

Figure 13: Lun et al. 2016, DOI: 10.1186/s13059-016-0947-7



2.8 Feature selection

- The curse of dimension: In high-dimensional data, all data points appear to be dissimilar (i.e., they are far apart), as the data become sparse in the high-dimensional space as dimensions increase (e.g., Euclidean distance).

- What does this mean for scRNA-seq? Trying to recover the global structure of a dataset using all genes is not a good idea. Often, we perform a variable selection step, selecting genes which we believe to be highly informative for a dataset's structure. One may view this as a first simple step of dimension reduction.

We will thus select genes which we hope are informative for recovering the biological structure. Prior to visualisation.

2.8.1 Concerns

Feature selection is crucial for dimension reduction. However, it remains subjective and there are no clear guidelines to follow. The amount of features to select depends on the complexity of your dataset, and the informativeness of each individual feature.

Many issues remain.

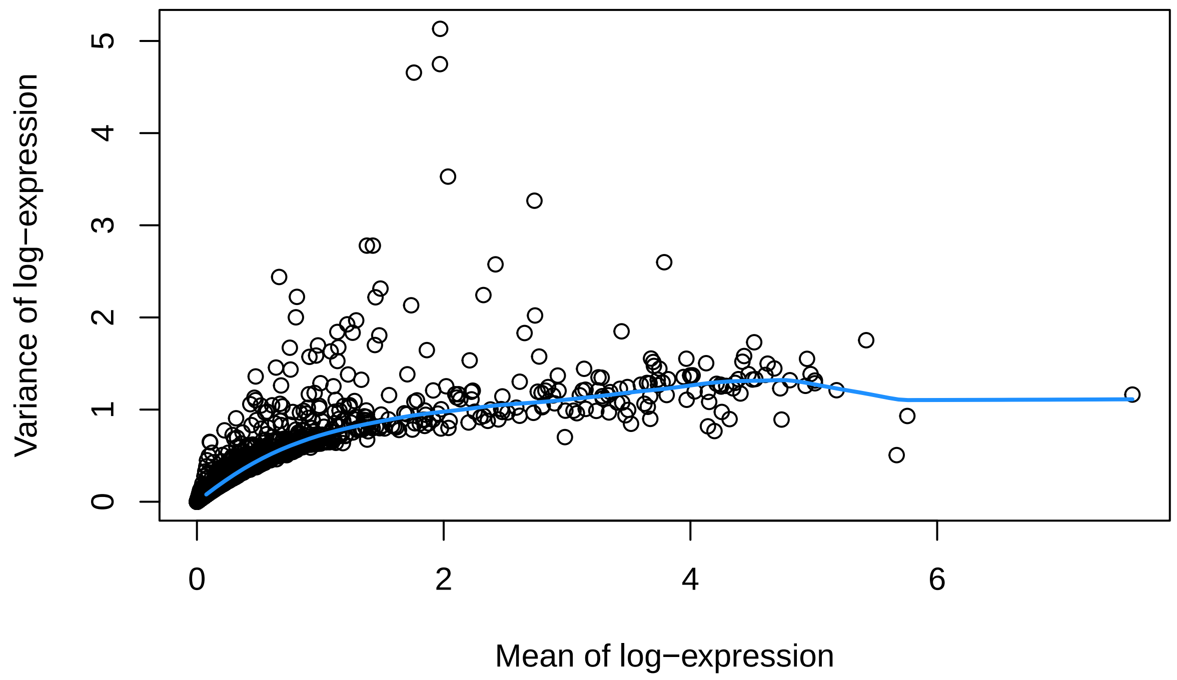
2.8.2 What defines an informative gene?

- Genes that are not expressed in any cell are obviously uninformative.
- Neither are genes that are highly expressed at a similar level in all cells. So what should we be looking for?
- Genes with high variance. But note our previous observations on the mean-variance relationship.
- Simply selecting features on high variance is common, but possibly not optimal.
- Genes with high variance relative to their mean. A (rightly so) popular approach is to select genes that have a high variance with respect to their mean. Often, first an empirical mean-variance trend is fitted, upon which genes with the highest positive residuals are selected. Being intuitive, reasonable and fairly straight-forward, this method is widely used.
- Genes with high deviance. See Townes et al. (2019) for feature selection based on Binomial deviance. Genes with a high deviance will most poorly fit a null model where the relative abundance is equal for all cells, which therefore are informative.

2.8.3 Example: feature selection based on high variance wrt mean

You need an assay log counts for this.

```r
sce <- logNormCounts(sce)
dec <- modelGeneVar(sce)
varfit <- metadata(dec)
plot(varfit$mean, varfit$var,
     xlab="Mean of log-expression",
    ylab="Variance of log-expression")
curve(varfit$trend(x), col="dodgerblue", add=TRUE, lwd=2)
```

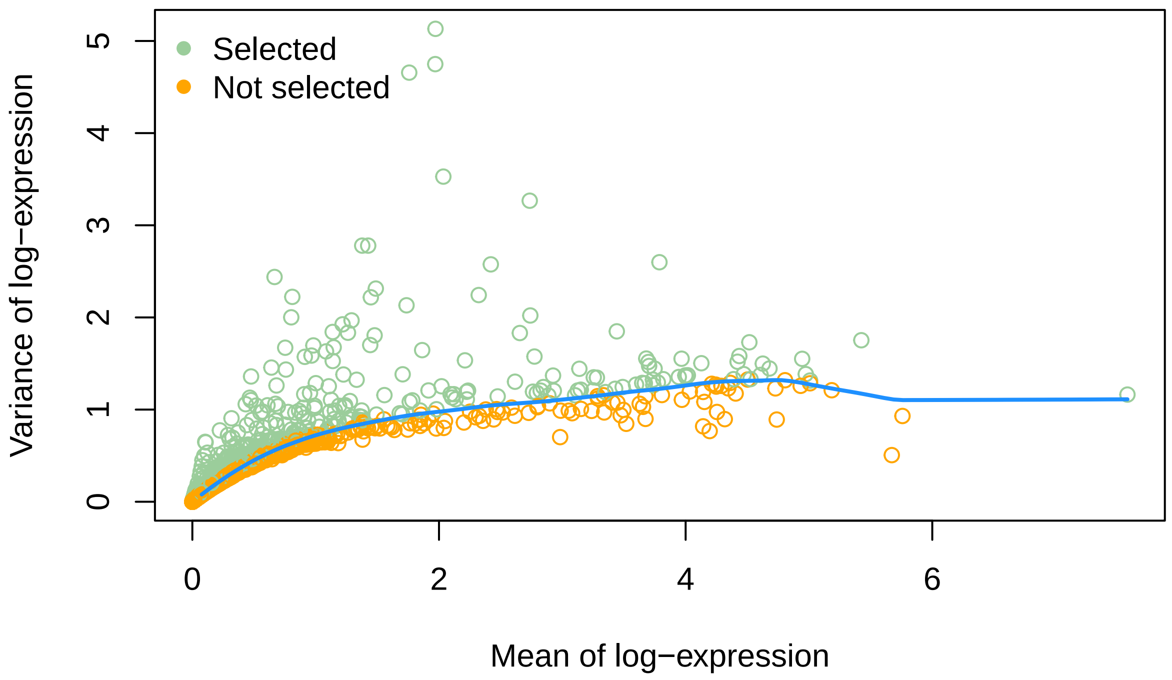


Select highly variable genes with variance larger than expected based on the mean variance trend. We then have to select the top number of most variable genes. We can for instance take the top 10% of the most variable genes

```r
hvg <- getTopHVGs(dec, prop=0.1)
head(hvg)
```

```
## [1] "LYZ"     "S100A9"  "S100A8"  "HLA-DRA" "CD74"    "TYROBP"
```

```r
plot(varfit$mean, varfit$var,
     col = c("orange", "darkseagreen3")[(names(varfit$mean) %in% hvg)+1],
     xlab="Mean of log-expression",
    ylab="Variance of log-expression")
curve(varfit$trend(x), col="dodgerblue", add=TRUE, lwd=2)
legend("topleft",
       legend = c("Selected", "Not selected"),
       col = c("darkseagreen3", "orange"),
       pch = 16,
       bty='n')
```



2.8.4 Genes with high deviance

See Townes et al. (2019) for feature selection based on Binomial deviance. Genes with a high deviance will most poorly fit a null model where the relative abundance is equal for all cells, which therefore are informative.

Note that the deviance for a GLM is defined as:

$$D = 2[l(\mathbf{y}, \mathbf{y}) - l(\mathbf{y}, \hat{})]$$

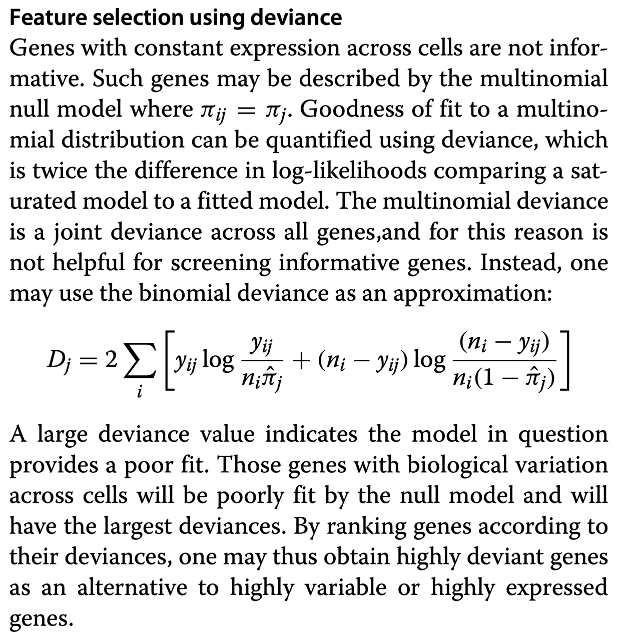
which is follows a χ^2 distribution with n degrees of freedom. Indeed, it is a LR test between the best model with a perfect model fit and the current model.

If the difference is significant, there is lack of fit.

2.8.5 Variance-stabilizing transformation (VST) using Pearson residuals

Lause et al. 2021 introduced the use of Pearson residuals to stabilize the variance (doi: 10.1186/s13059-021-02451-7).

- When a model is fitted to each gene we can construct pearson residuals that correct for the mean variance relation.
- Here we model $y_{ig} \sim NB(\mu_{ig} = \pi_g s_i N_i, \phi_g)$
- Pearson residuals are then defined as



Feature selection using deviance

Genes with constant expression across cells are not informative. Such genes may be described by the multinomial null model where $\pi_{ij} = \pi_j$. Goodness of fit to a multinomial distribution can be quantified using deviance, which is twice the difference in log-likelihoods comparing a saturated model to a fitted model. The multinomial deviance is a joint deviance across all genes,and for this reason is not helpful for screening informative genes. Instead, one may use the binomial deviance as an approximation:

$$D_j = 2 \sum_i \left[y_{ij} \log \frac{y_{ij}}{n_i \hat{\pi}_j} + (n_i - y_{ij}) \log \frac{(n_i - y_{ij})}{n_i(1 - \hat{\pi}_j)} \right]$$

A large deviance value indicates the model in question provides a poor fit. Those genes with biological variation across cells will be poorly fit by the null model and will have the largest deviances. By ranking genes according to their deviances, one may thus obtain highly deviant genes as an alternative to highly variable or highly expressed genes.

Figure 14: Townes et al. 2019, DOI: 10.1186/s13059-019-1861-6

$$e_i = \frac{y_{ig} - \mu_g}{\sqrt{\mu_g + \phi \mu_g^2}}$$

Indeed, the residuals are corrected for the mean variance relationship.

We can then assess if the residuals exhibit over excess variance with respect to the trend. If this is the case, the gene is informative. There is more variance than we expect under the intercept model.

$$\hat{\sigma}^2 = \sum_{i=1}^n \frac{e_i^2}{n-p}$$

Note, that this one of the variance estimator of the quasi-negative binomial model (See Lund et al., 2012, DOI 10.1515/1544-6115.1826).

Genes with high variance are informative.

Note, that there is also a close connection to the method of Townes. Another type of residuals for GLMs are deviance residuals, i.e.

$$e_i^d = \sqrt{2[l(y_{ig}, y_{ig}) - l(y_{ig}, \hat{\mu}_{ig})]}$$

Another variance estimator is

$$\hat{\sigma}^2 = \sum_{i=1}^n \frac{(e_i^d)^2}{n-p} = \frac{D}{n-p}$$

Note, that this is the other excess variance estimator for the quasi-negative binomial model. See Lund et al., 2012 (DOI 10.1515/1544-6115.1826).

2.9 Dimension reduction

We would like to be able to project the data to a low-dimensional space, while retaining as much information as possible. This will be useful downstream for, e.g., visualization, identification of batch effects, clustering, trajectory inference.

Many dimension reduction methods exist and this is a very active area of research in scRNA-seq. We will discuss the most common methods such as PCA, GLM-PCA, t-SNE and UMAP.

2.9.1 Concerns

Like feature selection, this remains a subjective step in the analysis. Several choices have to be made: input data, e.g., log-transformed data or not?; number of components to retain; parameter tuning; etc..

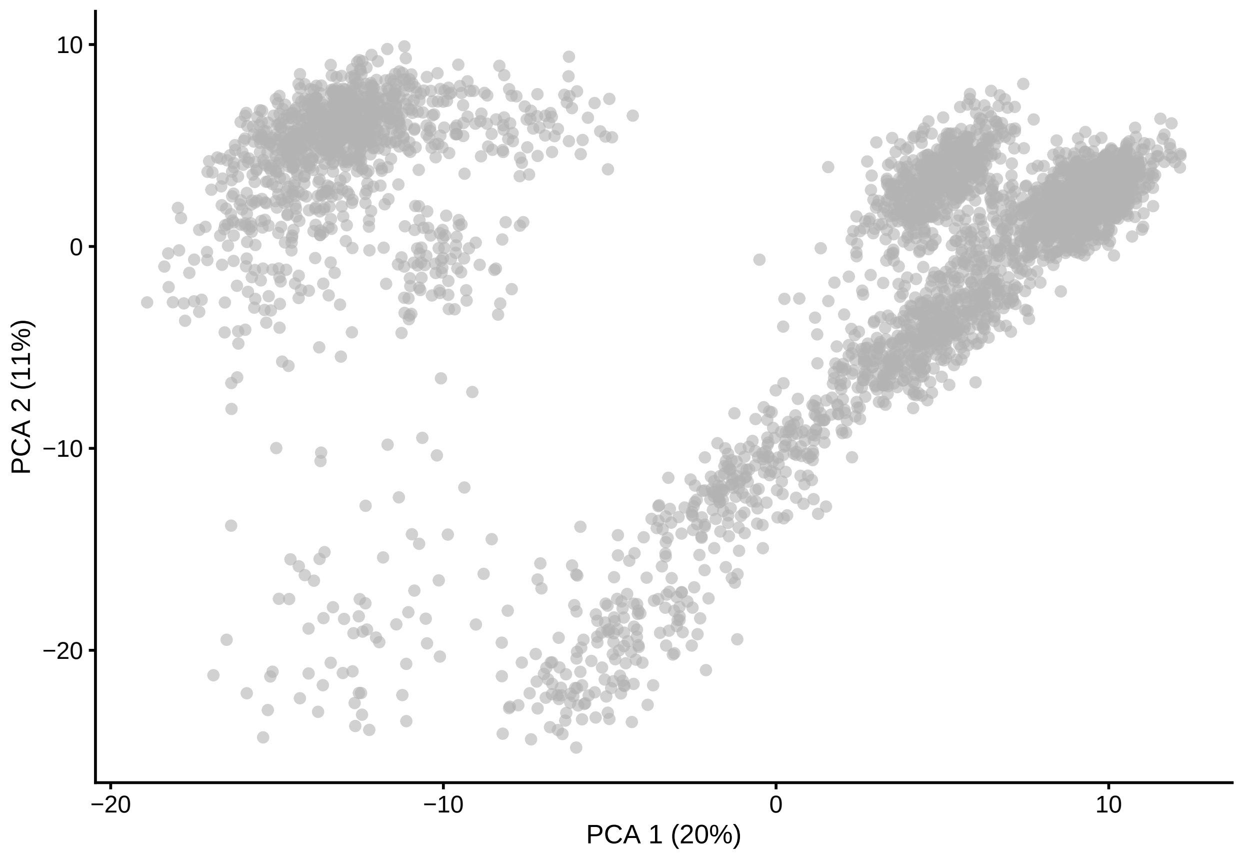
2.9.2 Principal Component Analysis

- A DR method is linear when the reduced dimensions are a linear function of the original variables.
- In PCA, each principal component is a linear combination of genes. See here for a geometric interpretation of PCA and for a PCA details
- Typically, PCA is performed on log-transformed normalized counts. The log-transformation helps somewhat, but not completely, to account for the mean-variance relationship.
- PCA works well for bulk RNA-seq data. However, the structure of scRNA-seq data is often too complex to be visualized by a small number of PCs.

2.9.2.1 Example

We do PCA on the highly variable genes.

```r
sce <- runPCA(sce, subset_row = hvg)
plotPCA(sce)
```



Note that the subset_row argument ensures that we use the previously selected HVGs to compute the PCs. Also note that by default runPCA will store the PCs in the PCA slot, hence overwriting the previous PCs. In the tutorials we will see that we already use PCA during the QC for visualisation purposes. These results are based on a quick normalization and prior to removing noisy genes and cells. Hence, it is ok to overwrite them. In other cases, it may be reasonable to keep more than one set of PCs: this can be achieved by specifying a different slot name via the name argument.

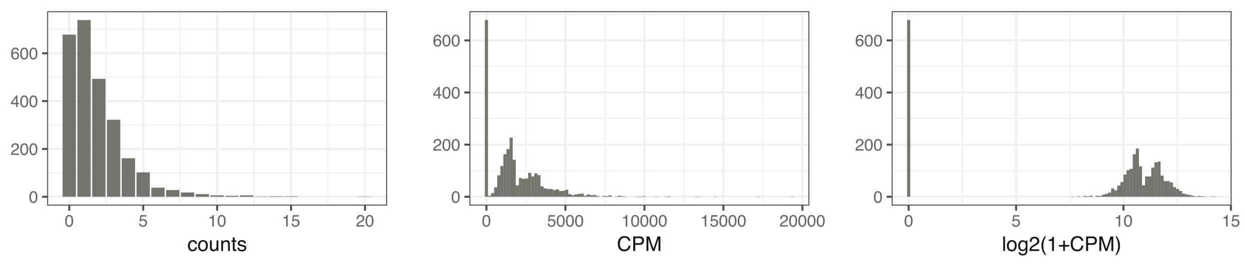
Finally, note that by default scater will compute the top 50 PCs. This is a reasonable choice, but it may be a good idea to explore the variance explained by each component to decide the number of components to retain. An alternative is to use the denoisePCA function from scran that aims at selecting the number of PCs that explain biological variability.

2.9.3 GLM-PCA

A generalization of PCA to exponential family distributions is provide by Townes et al. 2019. PCA is based on squared/Euclidean distances (e.g.Townes et al. 2019), which has a close link to Gaussian likelihood, which is inappropriate for count data due to the mean-variance relationship. Therefore, PCA results risks to be dominated by highly expressed genes, which have large variance.

They argue that UMI data are multinomially distributed, which can be approximated by a Poisson distribution when conditioning on the library size and propose a generalisation of PCA.

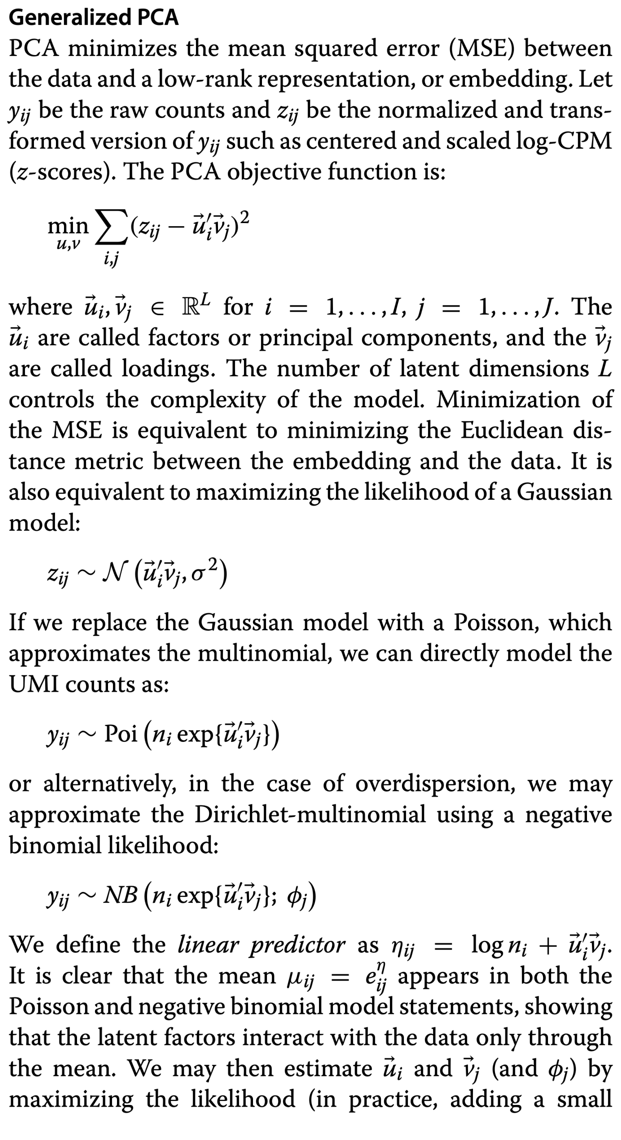
- Note, that full length plate based data scRNA-seq data are over disperse with respect to the Poisson distribution and therefore NB-PCA would be better.



(a) UMI counts (b) counts per million (CPM) (c) log of CPM

Fig. 2 Example of how current approaches to normalization and transformation artificially distort differences between zero and nonzero counts. **a** UMI count distribution for gene ENSG00000114391 in the monocytes biological replicates negative control dataset. **b** Counts per million (CPM) distribution for the exact same count data. **c** Distribution of $log_2(1 + CPM)$ values for the exact same count data

Figure 15: Townes et al. 2019, DOI: 10.1186/s13059-019-1861-6



Generalized PCA

PCA minimizes the mean squared error (MSE) between the data and a low-rank representation, or embedding. Let y_{ij} be the raw counts and z_{ij} be the normalized and transformed version of y_{ij} such as centered and scaled log-CPM (z-scores). The PCA objective function is:

$$\min_{u,v} \sum_{i,j} (z_{ij} - \vec{u}_i'\vec{v}_j)^2$$

where $\vec{u}_i, \vec{v}_j \in \mathbb{R}^L$ for $i = 1, \ldots, I$, $j = 1, \ldots, J$. The $\vec{u}_i$ are called factors or principal components, and the $\vec{v}_j$ are called loadings. The number of latent dimensions L controls the complexity of the model. Minimization of the MSE is equivalent to minimizing the Euclidean distance metric between the embedding and the data. It is also equivalent to maximizing the likelihood of a Gaussian model:

$$z_{ij} \sim \mathcal{N}\left(\vec{u}_i'\vec{v}_j, \sigma^2\right)$$

If we replace the Gaussian model with a Poisson, which approximates the multinomial, we can directly model the UMI counts as:

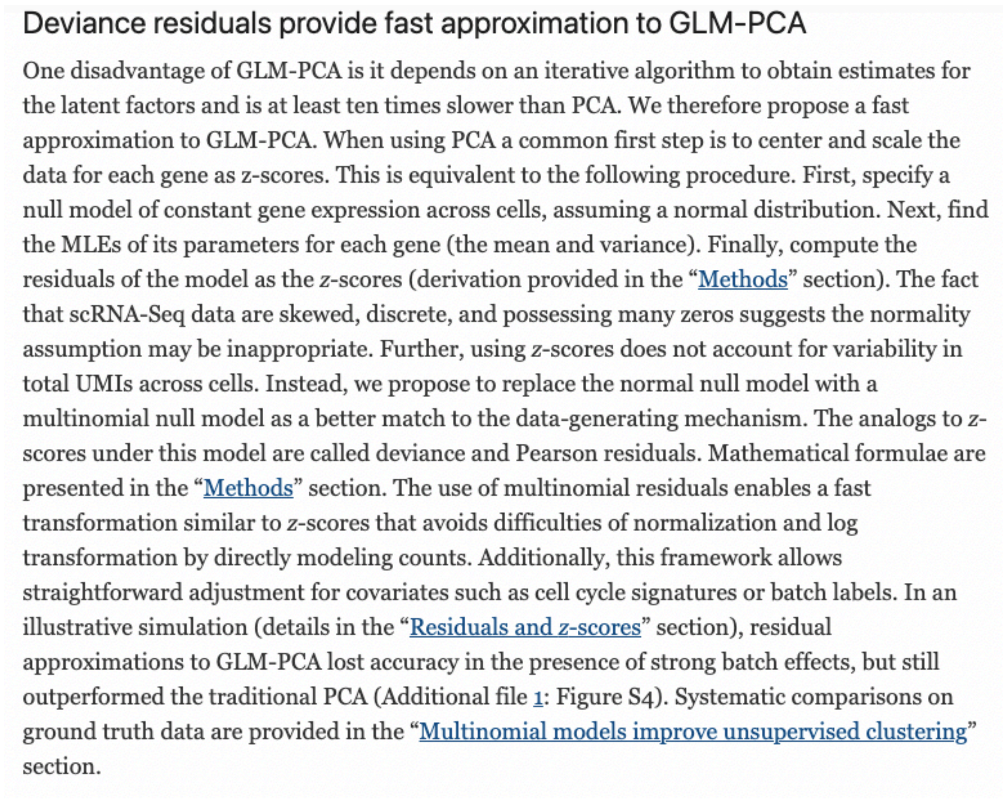
$$y_{ij} \sim \text{Poi}\left(n_i \exp\{\vec{u}_i'\vec{v}_j\}\right)$$

or alternatively, in the case of overdispersion, we may approximate the Dirichlet-multinomial using a negative binomial likelihood:

$$y_{ij} \sim NB\left(n_i \exp\{\vec{u}_i'\vec{v}_j\};\ \phi_j\right)$$

We define the *linear predictor* as $\eta_{ij} = \log n_i + \vec{u}_i'\vec{v}_j$. It is clear that the mean $\mu_{ij} = e^{\eta}_{ij}$ appears in both the Poisson and negative binomial model statements, showing that the latent factors interact with the data only through the mean. We may then estimate $\vec{u}_i$ and $\vec{v}_j$ (and ϕ_j) by maximizing the likelihood (in practice, adding a small

Figure 16: Townes et al. 2019, DOI: 10.1186/s13059-019-1861-6


Deviance residuals provide fast approximation to GLM-PCA

One disadvantage of GLM-PCA is it depends on an iterative algorithm to obtain estimates for the latent factors and is at least ten times slower than PCA. We therefore propose a fast approximation to GLM-PCA. When using PCA a common first step is to center and scale the data for each gene as z-scores. This is equivalent to the following procedure. First, specify a null model of constant gene expression across cells, assuming a normal distribution. Next, find the MLEs of its parameters for each gene (the mean and variance). Finally, compute the residuals of the model as the *z*-scores (derivation provided in the "Methods" section). The fact that scRNA-Seq data are skewed, discrete, and possessing many zeros suggests the normality assumption may be inappropriate. Further, using *z*-scores does not account for variability in total UMIs across cells. Instead, we propose to replace the normal null model with a multinomial null model as a better match to the data-generating mechanism. The analogs to *z*-scores under this model are called deviance and Pearson residuals. Mathematical formulae are presented in the "Methods" section. The use of multinomial residuals enables a fast transformation similar to *z*-scores that avoids difficulties of normalization and log transformation by directly modeling counts. Additionally, this framework allows straightforward adjustment for covariates such as cell cycle signatures or batch labels. In an illustrative simulation (details in the "Residuals and *z*-scores" section), residual approximations to GLM-PCA lost accuracy in the presence of strong batch effects, but still outperformed the traditional PCA (Additional file 1: Figure S4). Systematic comparisons on ground truth data are provided in the "Multinomial models improve unsupervised clustering" section.

Figure 17: Townes et al. 2019, DOI: 10.1186/s13059-019-1861-6

2.9.4 GLM-PCA Example

GLM-PCA is implemented in the `scry` Bioconductor package; it uses a Poisson model by default, but this can be changed to use negative binomial, multinomial or binomial.

Unlike runPCA, the GLM-PCA function does not allow for a selection of genes. Hence, we will first create a filtered SingleCellExperiment object that contains only the HVGs.

Similarly to PCA, GLM-PCA needs the number of components (factors) that need to be calculated. Here, we compute 10 latent factors.

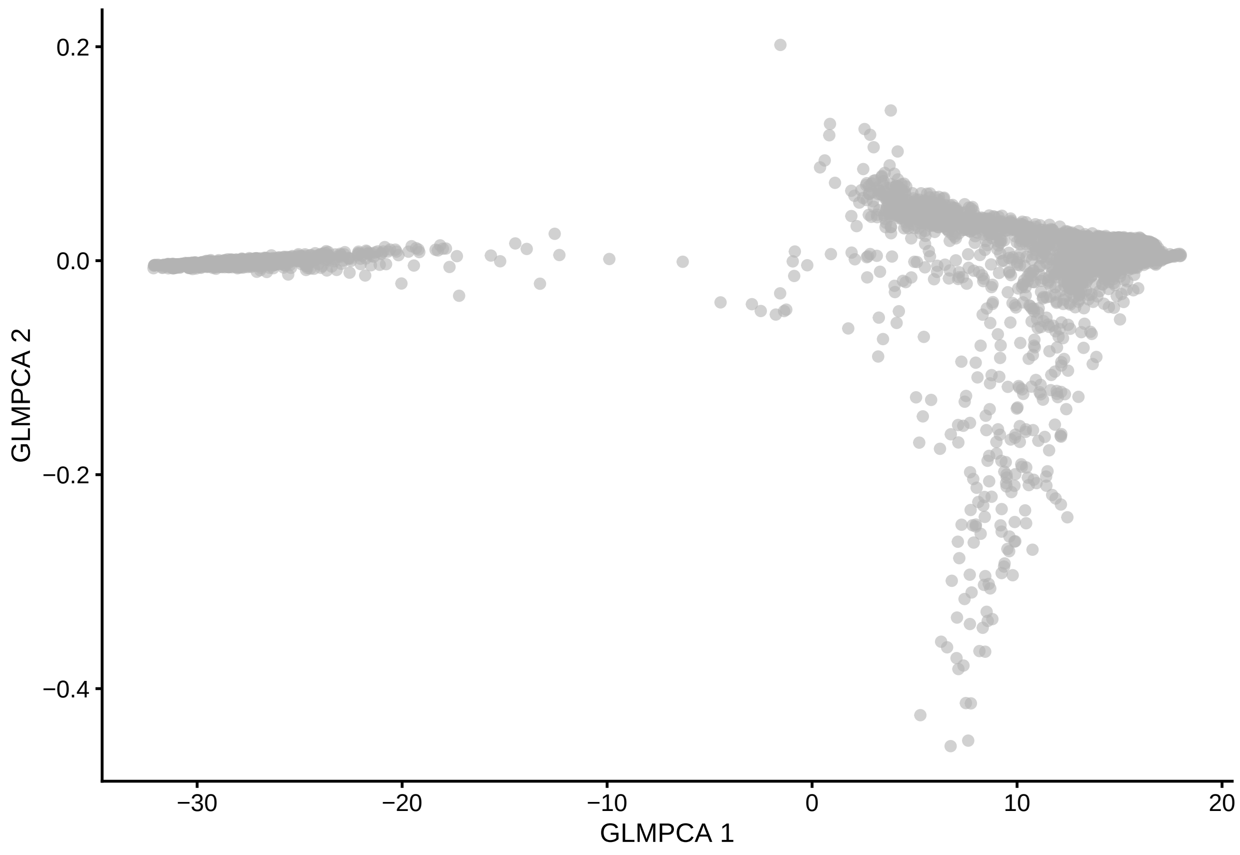
We use the minibatch argument to use only a random subset of observations to compute the gradient in the optimization algorithm. This speeds up computations and avoids memory problems in big datasets.

```r
suppressPackageStartupMessages({
  library(scry)
})
set.seed(100000)
filtered <- sce[hvg,]
filtered <- GLMPCA(filtered, L=10, minibatch = "stochastic")
filtered
```

```
## class: SingleCellExperiment
## dim: 874 3651
## metadata(2): Samples glmpca
## assays(2): counts logcounts
## rownames(874): LYZ S100A9 ... RBAK-RBAKDN WRB
## rowData names(3): ID Symbol location
## colnames(3651): AAACCTGAGAAGGCCT-1 AAACCTGAGACAGACC-1 ...
##   TTTGTCAGTTAAGACA-1 TTTGTCATCCCAAGAT-1
## colData names(11): Sample Barcode ... DoubletScore sizeFactor
## reducedDimNames(2): PCA GLMPCA
## mainExpName: NULL
## altExpNames(0):
```

To visualize the first two components of GLM-PCA, we can use the plotReducedDim function from `scater`.

```
plotReducedDim(filtered, "GLMPCA")
```



2.9.5 NewWave

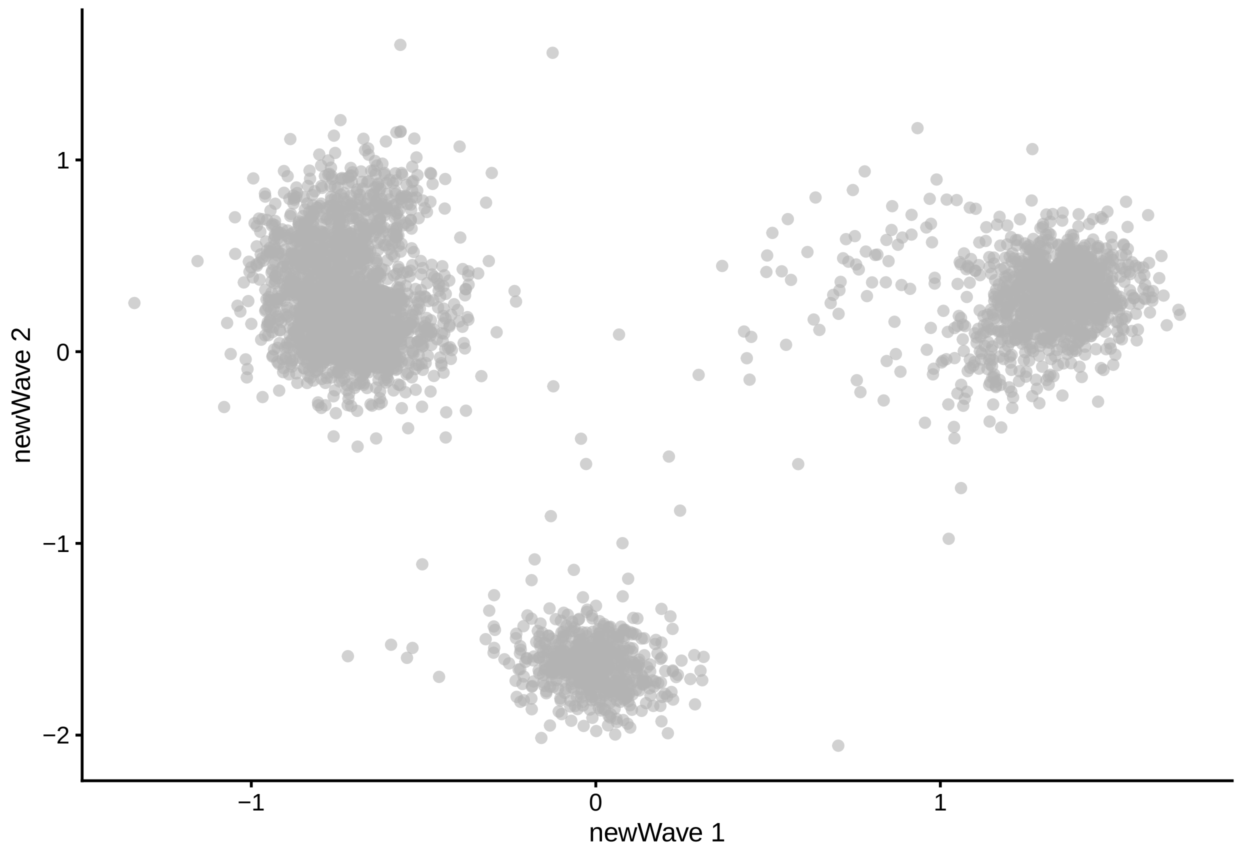
An alternative method is implemented in the NewWave package, which implements a negative binomial factor analysis model and uses a penalized likelihood to estimate the latent factors (Agostinis et al. 2022 DOI: 10.1093/bioinformatics/btac149).

2.9.5.1 NewWave Example NewWave can also use minibatches, and if multiple CPUs are available parallel computing, to speed up computations. The n_gene_par, n_cell_par, and n_gene_disp arguments allow the user to choose how many observations to use to estimate the gene- and cell-specific parameters. The children argument allows one to use multiple cores; here we use four, but you should change it depending on how many cores your computer has (see parallel::detectCores()).

```
suppressPackageStartupMessages({
  library(NewWave)
})
set.seed(100000)
filtered <- newWave(filtered, K=10, children=4,
                    n_gene_disp = 100, n_gene_par = 100,
                    n_cell_par = 100)
filtered
```

```
## class: SingleCellExperiment
## dim: 874 3651
## metadata(2): Samples glmpca
## assays(2): counts logcounts
## rownames(874): LYZ S100A9 ... RBAK-RBAKDN WRB
## rowData names(3): ID Symbol location
## colnames(3651): AAACCTGAGAAGGCCT-1 AAACCTGAGACAGACC-1 ...
##   TTTGTCAGTTAAGACA-1 TTTGTCATCCCAAGAT-1
## colData names(11): Sample Barcode ... DoubletScore sizeFactor
## reducedDimNames(3): PCA GLMPCA newWave
## mainExpName: NULL
## altExpNames(0):
```

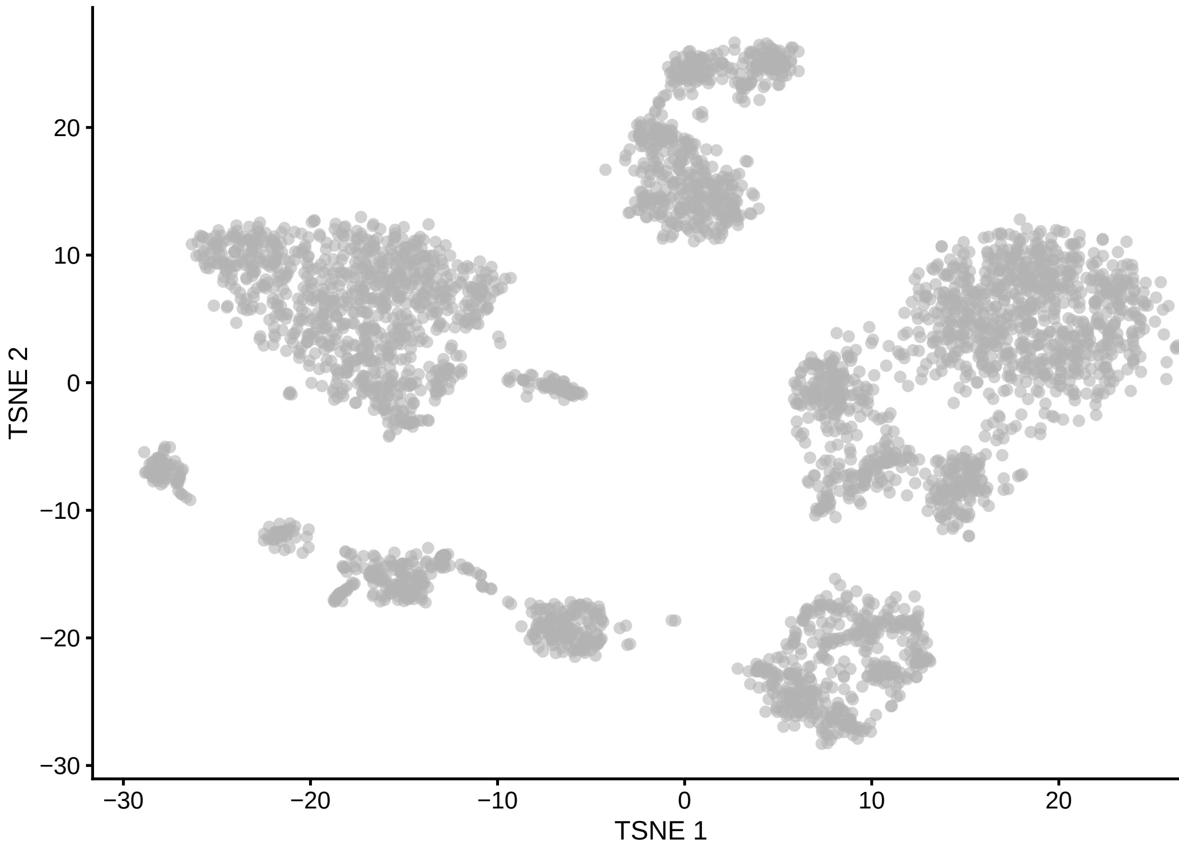
```
plotReducedDim(filtered, "newWave")
```



2.9.6 TSNE and UMAP

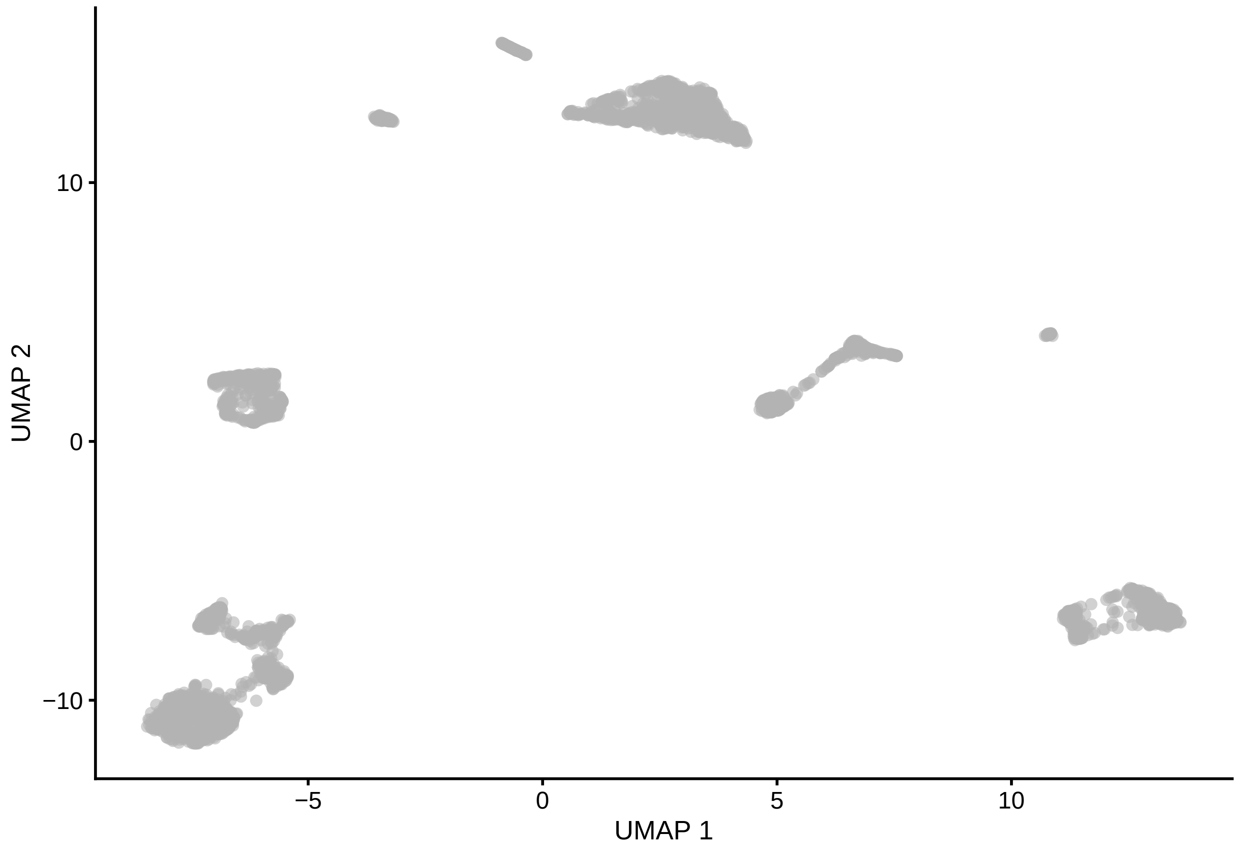
- Non-linear dimension reduction methods
- Visualizations of reduced dimensions from linear dimension reduction methods are often 'overcrowded', and it is hard to see structure.
- Non-linear dimension reduction methods can overcome this problem. As the name suggests, the reduced dimensions are a non-linear function of the observed data. We will not go into detail as to how these work under the hood, but provide a few guidelines for the most popular methods. Often, the top (10-50) PCs are provided as input.
- t-SNE: Preserving local rather than global distances. Therefore, distances on a t-SNE reduced dimension plot can only be interpreted locally, i.e., cells that are close together in reduced dimension will have a similar transcriptome, but cells that are far away may not necessarily have a very distinct transcriptome.
- UMAP: Claimed to be better than t-SNE on preserving global differences. Therefore, UMAP is also often used in analyses such as trajectory inference, where this is important.

```r
sce <- runTSNE(sce, dimred="PCA")
plotTSNE(sce)
```



2.9.6.1 Example

```r
  if(!"uwot" %in% installed.packages()[,1]) BiocManager::install("uwot")
sce <- runUMAP(sce, dimred = 'PCA', external_neighbors=TRUE)
plotUMAP(sce)
```



2.10 Clustering: unsupervised cell type identification

- The identification of homogeneous, biologically relevant entities that define a common cell identity.
- Traditional clustering methods do not perform great, often graph-based clustering or ensemble methods are used.

2.10.1 Concerns

- There are many resolutions at which this can be done, and there is no such thing as an optimal cell type resolution; cells can be described by a hierarchy of different cell types and states. More clusters may result in higher resolution cell types, but can hamper replicability of the results across datasets.

2.11 (Semi-)supervised cell type identification

- We can use prior knowledge to reliably identify cell clusters and annotate them.
- We can annotate a new scRNA-seq dataset by training a classifier that predicts cell type label based on existing data.

1. Gather reference data that are relevant for your current dataset you would like to annotate.
2. Train a classifier (lasso/ridge, SVM, neural net, kNN, …) to classify cells into cell types, given gene expression data.
3. Use pre-trained classifier to estimate, for each cell, P(cell i is cell type Z | gene expression of cell i).

2.11.1 Concerns

The reference data used for training the classifier must be representative for the dataset we would like to annotate. Problems may occur if e.g., different technologies are used, or particular (rare) cell types are not present in the reference data.

3 Differential expression (DE), discovery of marker genes

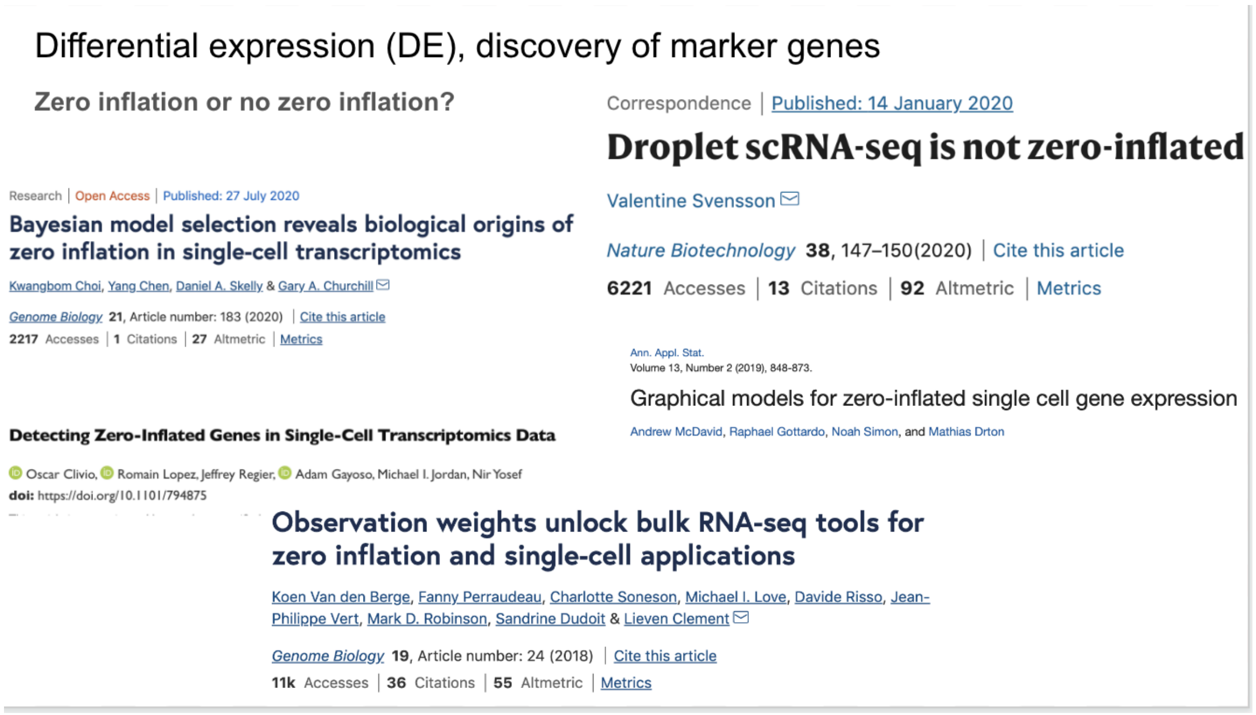
- DE is often a first step towards interpreting differences between cell groups, and provides a basis for further biological validation.
- DE analysis in scRNA-seq is typically similar to bulk RNA-seq, e.g. edgeR analysis. In fact, due to the high number of cells, often simpler methods may be used, e.g. Wilcoxon tests.

3.1 Concerns

- There is an ongoing debate in the community about the appropriate noise model for scRNA-seq data. For example, are the data zero inflated?
- Post-selection inference problem, the data are used twice
  1. Cluster
  2. Find biomarkers between clusters
- Pseudo-replication in multi-subject-multi-cell experiments!

3.2 Zero-inflation

```r
knitr::include_graphics("./images_sequencing/zeroInflation.png")
```



Differential expression (DE), discovery of marker genes

Zero inflation or no zero inflation?

Research | Open Access | Published: 27 July 2020

Bayesian model selection reveals biological origins of zero inflation in single-cell transcriptomics

Kwangbom Choi, Yang Chen, Daniel A. Skelly & Gary A. Churchill

Genome Biology **21**, Article number: 183 (2020) | Cite this article

2217 Accesses | **1** Citations | **27** Altmetric | Metrics

Detecting Zero-Inflated Genes in Single-Cell Transcriptomics Data

Oscar Clivio, Romain Lopez, Jeffrey Regier, Adam Gayoso, Michael I. Jordan, Nir Yosef

doi: https://doi.org/10.1101/794875

Correspondence | Published: 14 January 2020

Droplet scRNA-seq is not zero-inflated

Valentine Svensson

Nature Biotechnology **38**, 147–150(2020) | Cite this article

6221 Accesses | **13** Citations | **92** Altmetric | Metrics

Ann. Appl. Stat.
Volume 13, Number 2 (2019), 848-873.

Graphical models for zero-inflated single cell gene expression

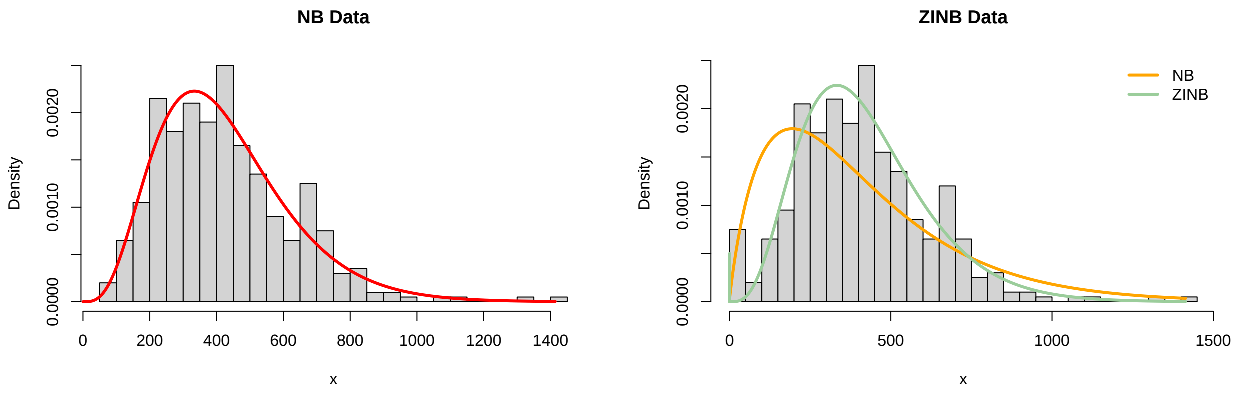
Andrew McDavid, Raphael Gottardo, Noah Simon, and Mathias Drton

Observation weights unlock bulk RNA-seq tools for zero inflation and single-cell applications

Koen Van den Berge, Fanny Perraudeau, Charlotte Soneson, Michael I. Love, Davide Risso, Jean-Philippe Vert, Mark D. Robinson, Sandrine Dudoit & Lieven Clement

Genome Biology **19**, Article number: 24 (2018) | Cite this article

11k Accesses | **36** Citations | **55** Altmetric | Metrics



- Motivation for zero-inflation is that we may consider two types of zeros in scRNA-seq data:
  - Biological zeros: We observe a zero count because the gene is not expressed in that cell.
  - Technical zeros: Even though the gene was expressed in that cell, we do not observe any molecules due to technical reasons (e.g., low capture efficiency / sequencing depth).
- A zero-inflated count distribution is a mixture distribution that consists of a point mass at zero and a count component. For example, a zero-inflated negative binomial (ZINB) distribution.

$$\begin{aligned} Y_{ig} &\sim ZINB(\mu_{ig}, \phi_g, \pi_{ig}) \\ f(Y_{ig}) &= \pi_{ig}\delta_0 + (1-\pi_{ig}) f_{NB}(\mu_{ig}, \pi_{ig}) \end{aligned}$$

Our take:

- Data from full-length protocols consist of high counts and many zeros, and are more likely to be zero inflated.

- Instead, data from UMI droplet-based protocols consist of low counts with manyzeros. This seems to reasonably fit under a negative binomial distribution, and zero inflation may not be necessary.

3.3 Pseudo-replication in multi-cell-multi-patient scRNA-seq data

- In multi-patient datasets, we have a hierarchical correlation structure.
- Cells are nested within patient → Expect correlation between cells from the same patient.

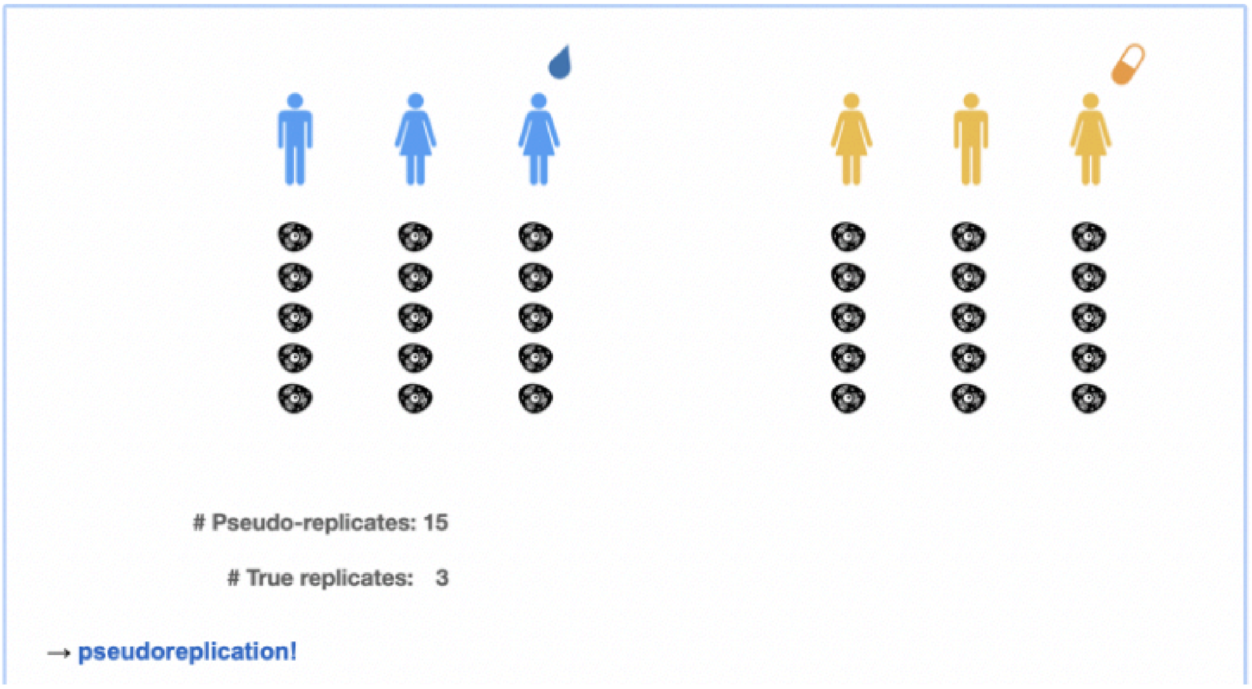


Figure 18: Image credit Milan Malfait

- When assessing differences in means within cell types between different treatments, we can bulk the cells of the same cell types to perform pseudobulks per subject.
- These pseudobulk samples can than be assessed using conventional bulk RNA-seq tools.

- Implementation see OSCA Book

3.4 Differential Distributions

- A few methods exist that compare distributions between treatments.
- But, they do not allow to interpret the difference in the distributions.

Acknowledgements

Thanks to Davide Risso for sharing his materials from the CSAMA 2022 workshop, which I used as the starting point for the R code in this document.

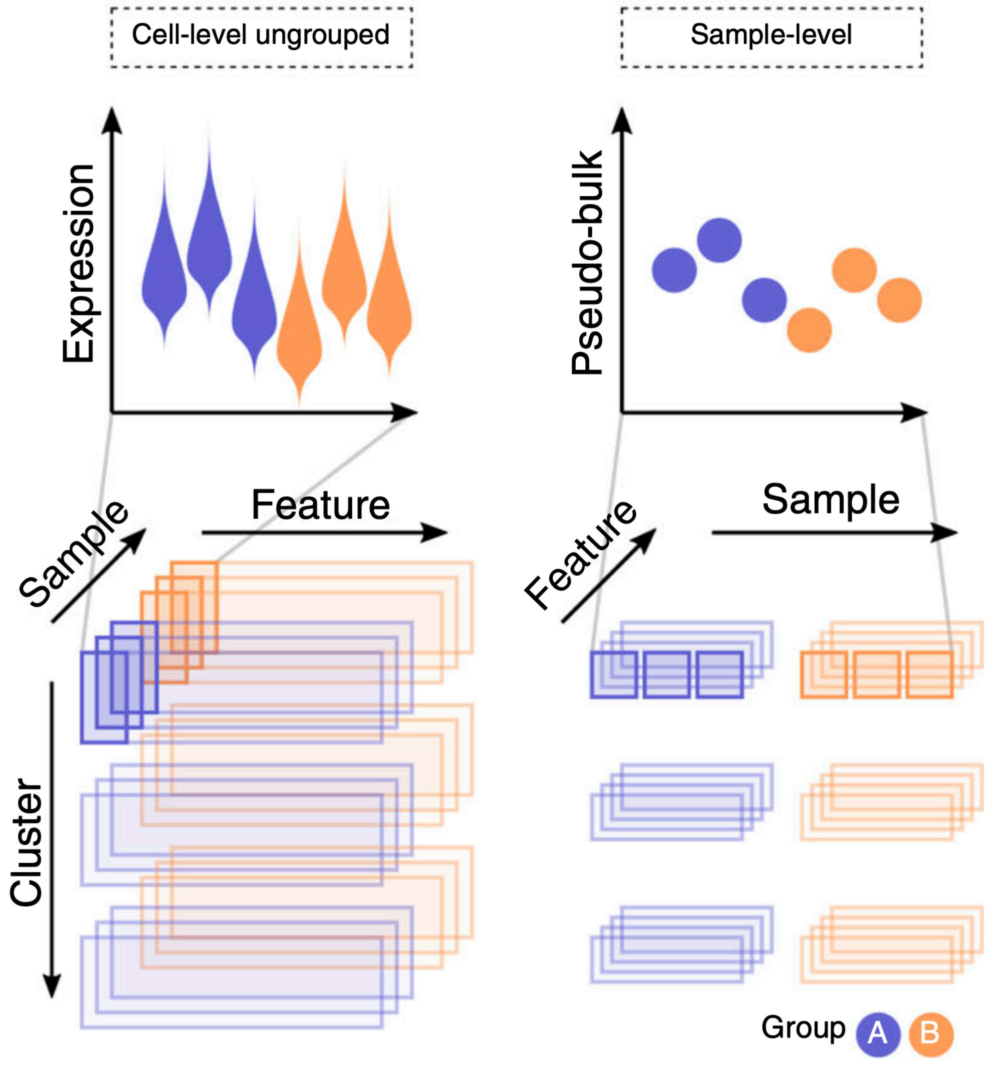
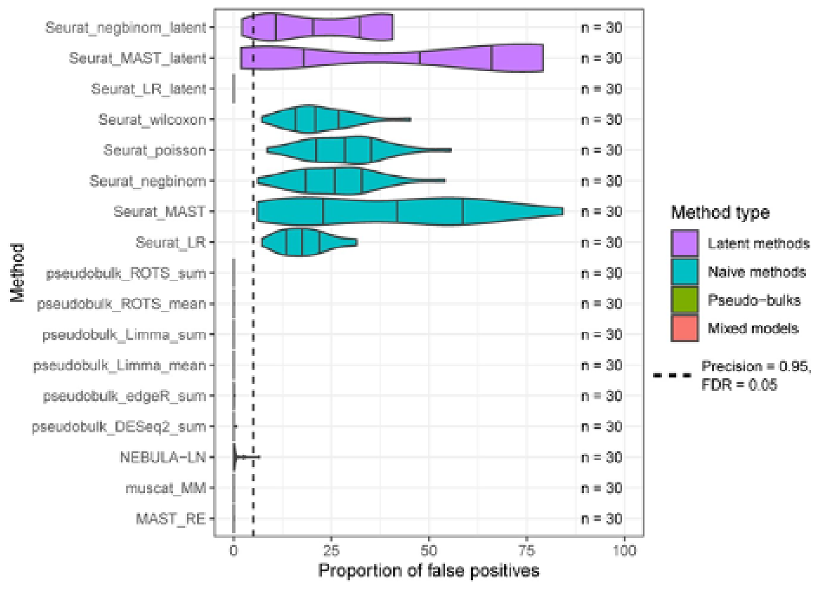


Figure 19: Crowell et al. 2020 (DOI: 10.1038/s41467-020-19894-4)

Figure 4 Mock analysis using real data to estimate the proportion of false positives. The mock analysis was performed ...



Brief Bioinform, Volume 23, Issue 5, September 2022, bbac286, https://doi.org/10.1093/bib/bbac286

The content of this slide may be subject to copyright: please see the slide notes for details.

Figure 20: Juntilla et al. 2022 (DOI 10.1093/bib/bbac286)

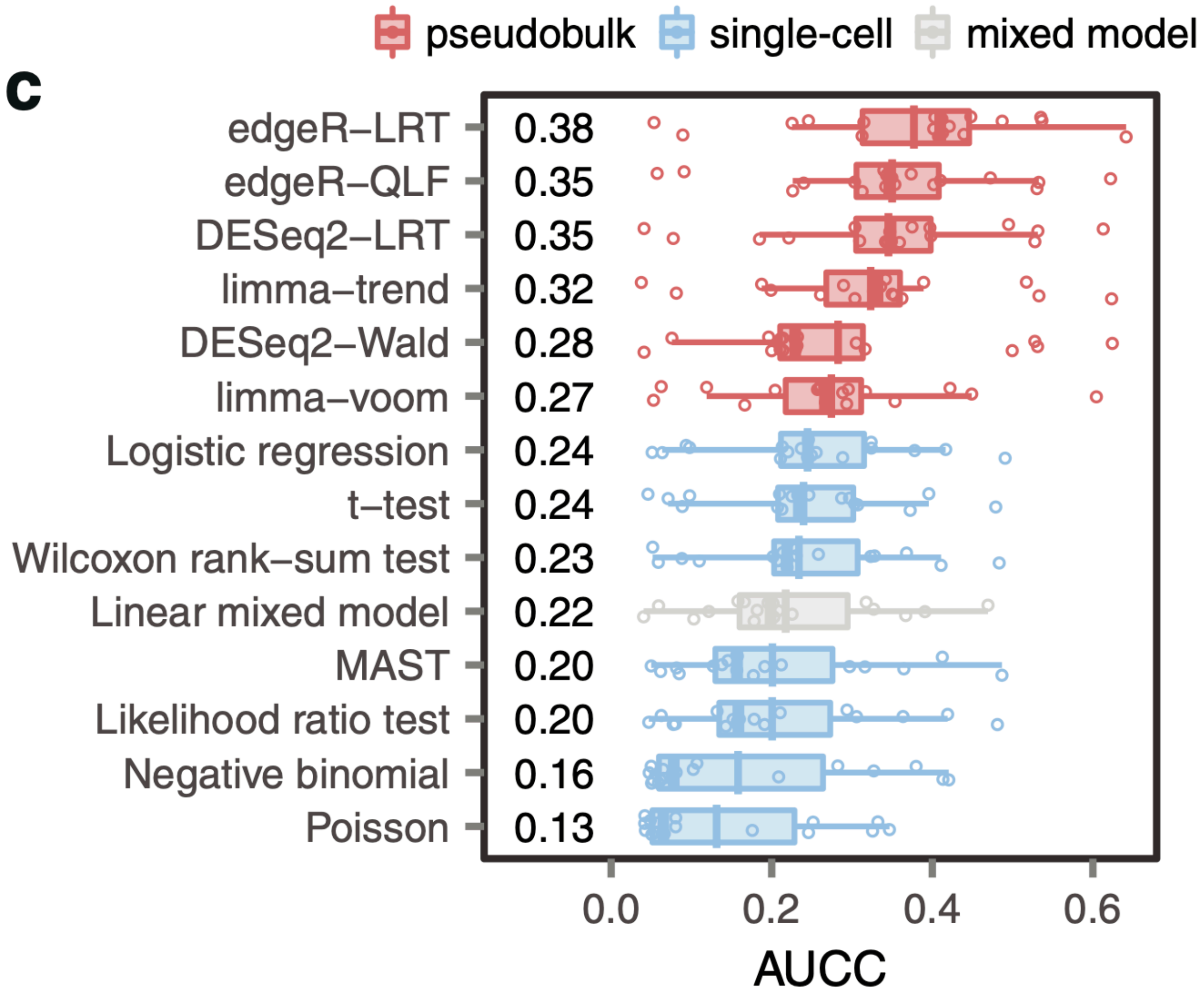


Figure 21: Squair et al. 2021 (DOI 10.1038/s41467-021-25960-2)