



# scRNAseq normalization and gene set selection

Åsa Björklund  
asa.bjorklund@scilifelab.se

# Outline

- Introduction
- Normalization / Transformations
- Removal of confounders
- Gene set selection

Why do we need to normalize  
scRNAseq data?

# Biological and technical variation

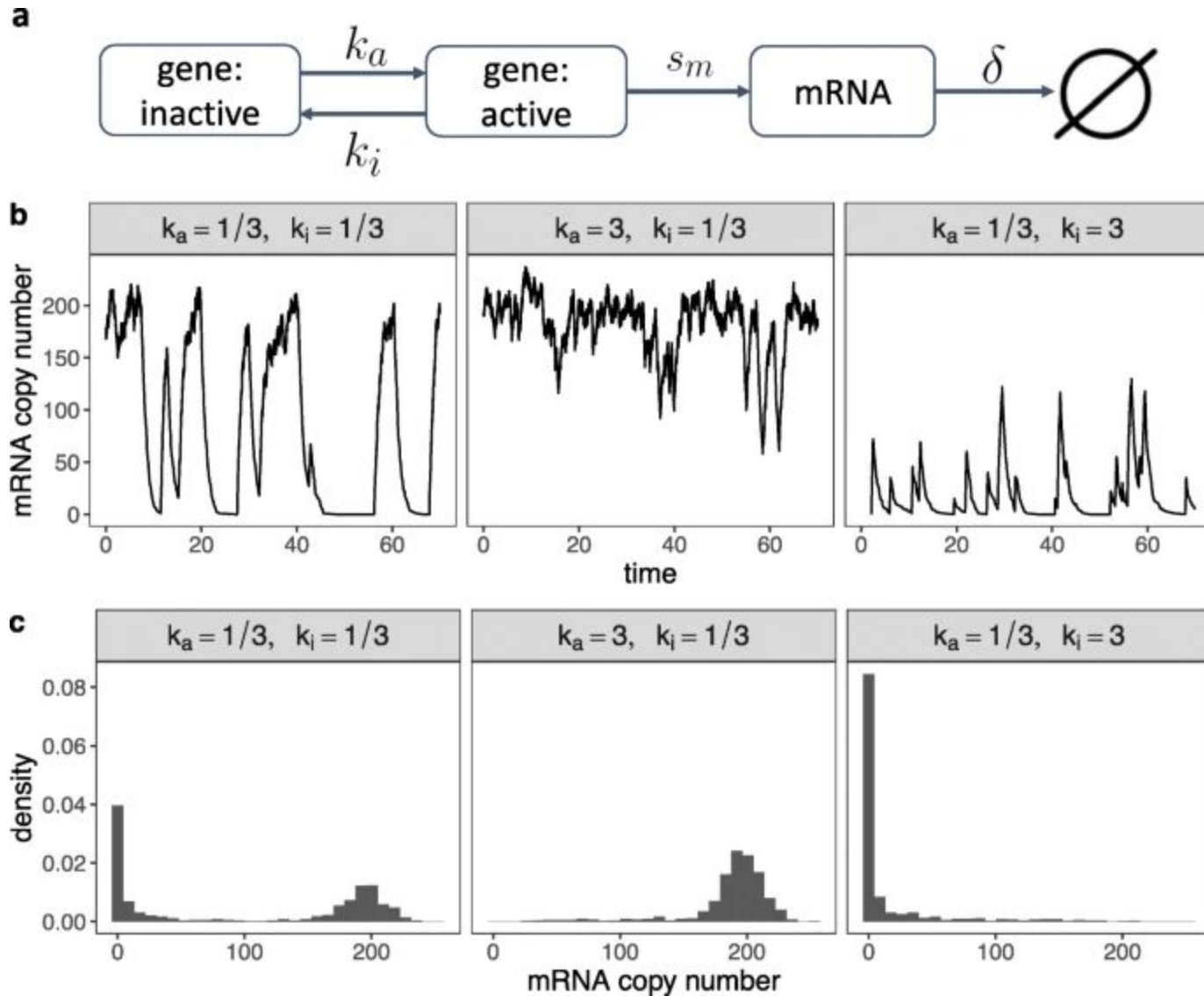
- Biological variation:
  - Cell type/state
  - Cell cycle
  - Cell size
  - Sex, Age, ...
  - Etc..
- Technical variation
  - Cell quality
  - Library prep efficiency
  - Batch effects
  - Etc...

# Biological and technical variation

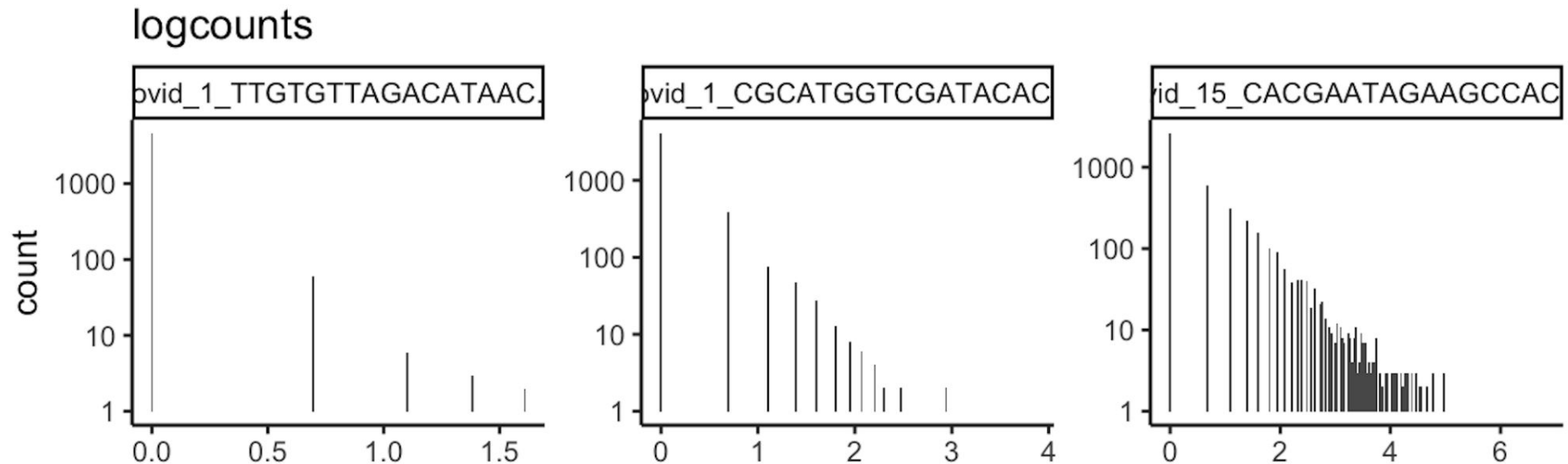
- Biological variation:
  - Cell type/state
  - Cell cycle
  - Cell size
  - Sex, Age, ...
  - Etc..
- Technical variation
  - Cell quality
  - Library prep efficiency
  - Batch effects
  - Etc..

To identify cell types  
we would like to  
remove all other  
sources of variation.

# Genes with different distributions



# Cells with different distributions



# Normalization

- Want to make expression comparable across samples, cells and genes.
- Involves 3 main steps:
  - Scaling
  - Transformation
  - Removal of unwanted variation



# Scaling Normalization

- **Depth normalization** – for uneven sequencing depth
- **Gene length normalization** – for differences in gene detection due to gene length (full length methods)
- **Drop-out rate normalization** – for differences in RNA content / drop-out rates

**OBS!** After scaling we have relative amounts of the different genes, not absolute values.

# Depth normalization

- In most cases the amount of RNA – and of UMIs/reads differ between cells.
  - Can be biological, different celltypes have different RNA amount
  - Can be technical, RT-efficiency may differ between droplets/wells.
- NOTE! Also important to check for outlier genes that constitute large proportion of the reads!

# Bulk RNAseq methods

- **CPM**: Controls for sequencing depth when dividing by total count
- **RPKM/FPKM**: Controls for sequencing depth and gene length. Good for technical replicates, not good for sample-sample due to compositional bias. Assumes total RNA output is same in all samples.
- **TPM**: Similar to RPKM/FPKM. Corrects for sequencing depth and gene length. Also comparable between samples but no correction for compositional bias.

$$\text{CPM}_i = \frac{X_i}{\frac{N}{10^6}} = \frac{X_i}{N} \cdot 10^6$$

$$\text{FPKM}_i = \frac{X_i}{\left(\frac{\tilde{l}_i}{10^3}\right) \left(\frac{N}{10^6}\right)} = \frac{X_i}{\tilde{l}_i N} \cdot 10^9$$

$X_i$ : observed count

$l_i$ : length of the transcript

$N$  number of fragments sequenced

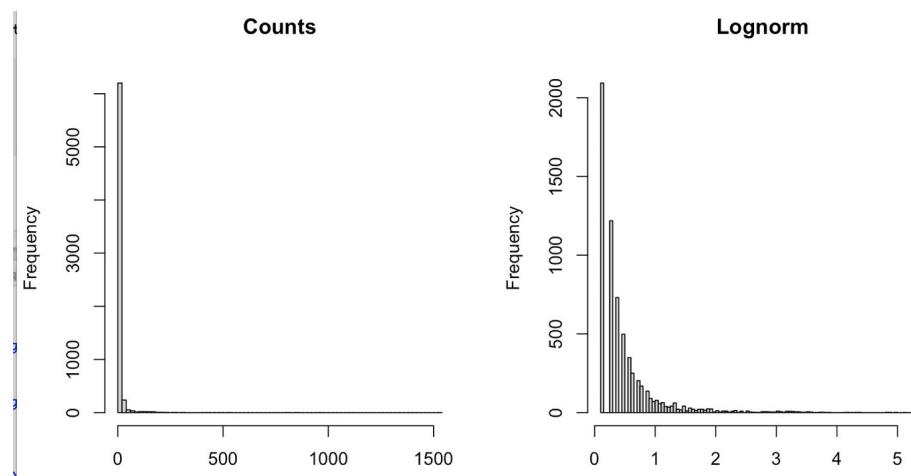
$$\text{TPM}_i = \frac{X_i}{\tilde{l}_i} \cdot \left( \frac{1}{\sum_j \frac{X_j}{\tilde{l}_j}} \right) \cdot 10^6$$

# Transformation Normalization

- Idea is to have a distribution of expression and variance in expression values that best captures biological variation.

# Logtransformation

- Log-transformed values approaches normal distribution for bulk RNAseq data
- For scRNAseq – more similar to zero-inflated binomial
- Still more similar to normal distribution than raw counts.

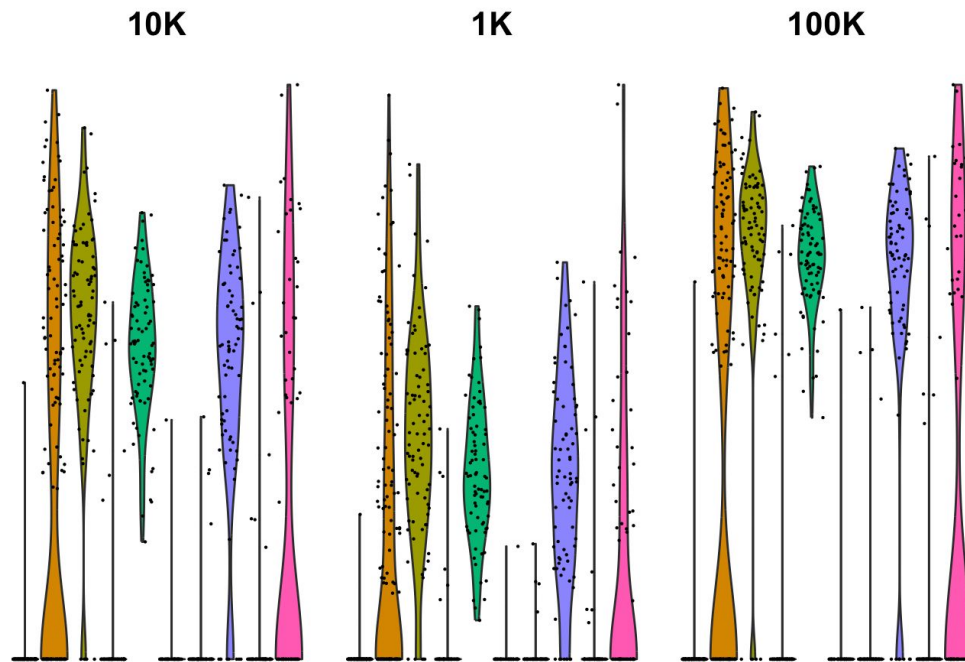


# Scaling factors and pseudocounts

Default scale factor is often 10 000, and pseudocount 1.

Relative counts:  $c/\text{sum}(c) * \text{scale.factor}$

Lognorm:  $\log(\text{RC}+1)$



# Other transforms

- Square root transform - is the variance stabilizing transformation for Poisson-distributed counts.
- Hyperbolic sine (arcsinh)

# Bulk RNAseq methods

- **TMM/RLE/MRN:** Improved assumption: The output between samples for a core set only of genes is similar. Corrects for compositional bias. RLE and MRN are very similar and correlates well with sequencing depth. `edgeR::calcNormFactors()` implements TMM, TMMwzp, RLE & UQ. `DESeq2::estimateSizeFactors` implements median ratio method (RLE). Does not correct for gene length.
- **VST/RLOG/VOOM:** Variance is stabilised across the range of mean values. For use in exploratory analyses. `vst()` and `rlog()` functions from *DESeq2*. `voom()` function from *Limma* converts data to normal distribution.



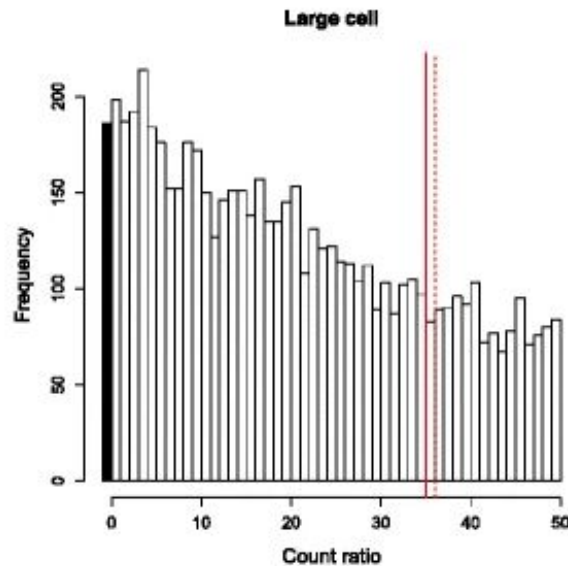
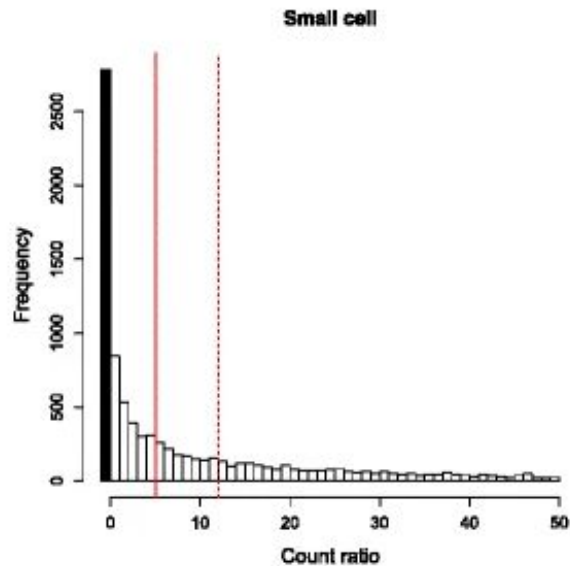
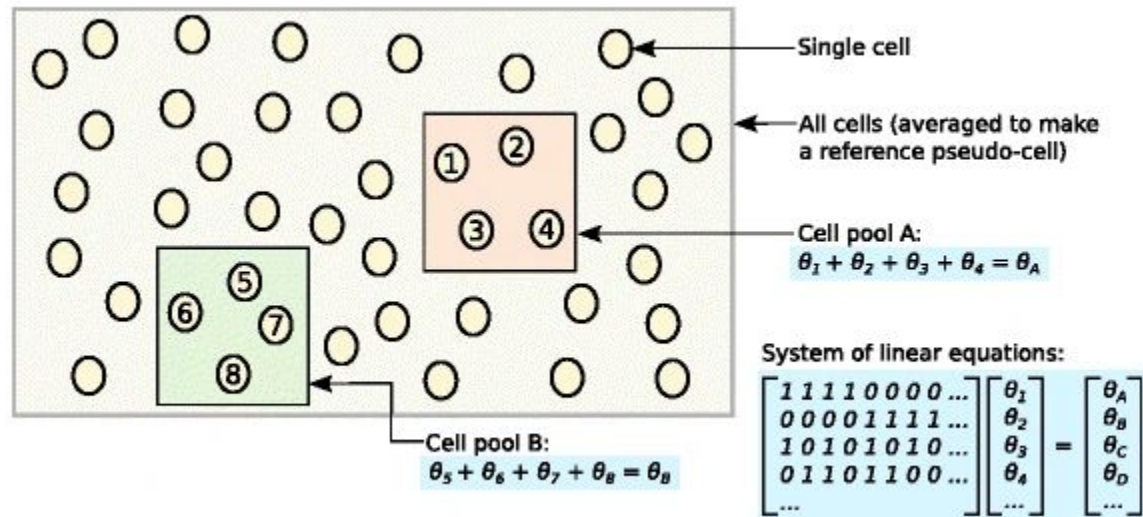
# Depth normalization and logtransformation in practice:

- Lognorm: divide by sequencing depth \* a scale factor and log-transform the data
- Scater **normalize** – uses total counts or provided size factors. Default `return_log = TRUE`. scale factor = 1M.
- Seurat **NormalizeData** – returns log-normalized data with `scale.factor = 10K` by default.
- Scanpy **normalize\_total** – normalize by sequencing depth. **OBS!** scale factor default median of all cells.
  - then need to run **log1p**.

# scRNAseq normalization methods

- Deconvolution/Scran (Pooling-Across-Cells)
- SCnorm (Expression-Depth Relation)
- SCTransform
- Census
- Sanity
- ZINB-WaVE
- scVI
- BASiCS
- More.....
- Dino
- Normalizr
- DCA
- SAVER
- Magic

# Deconvolution



Lun et al. Genome Biol. 2016

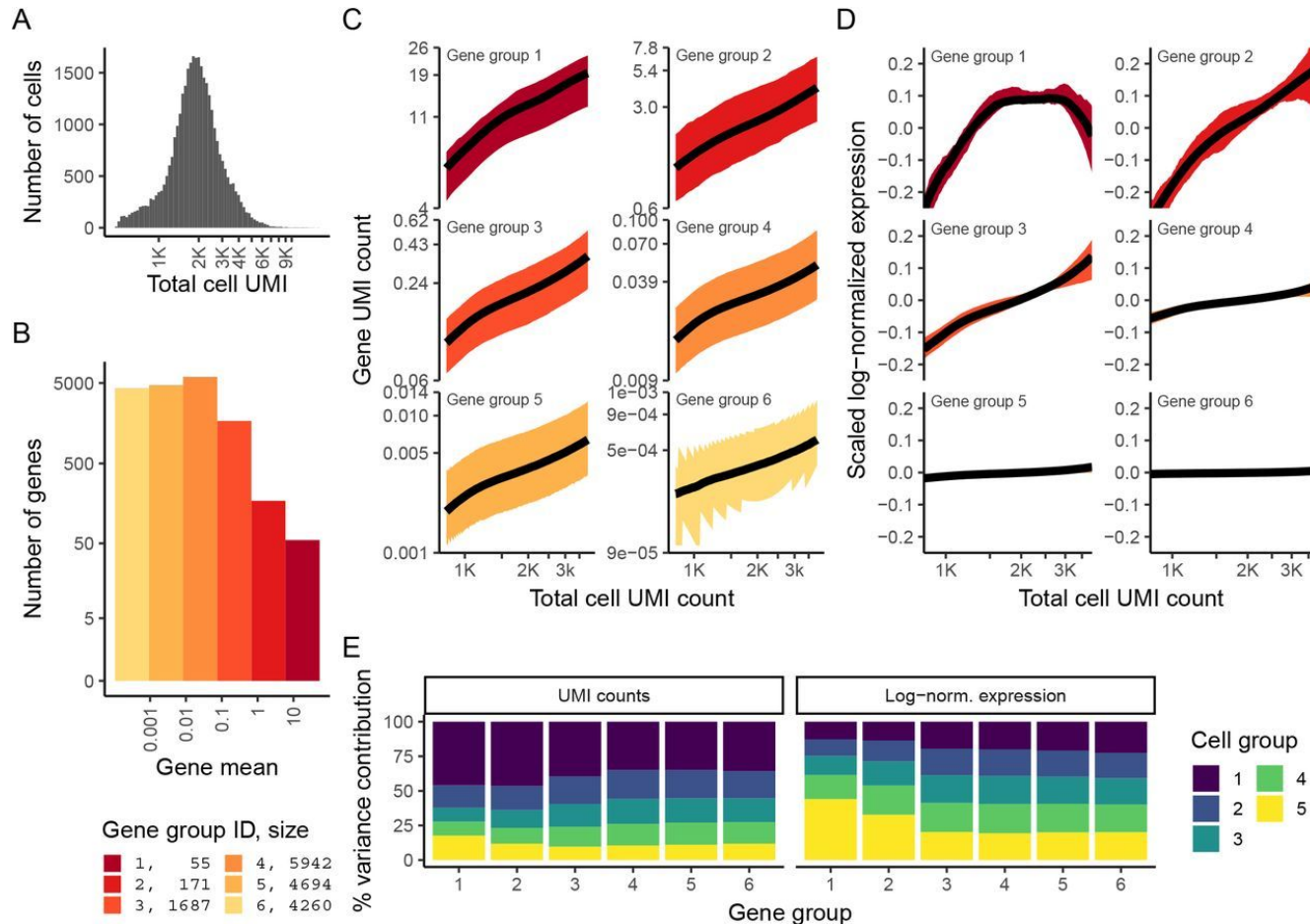
# Scran - computeSumFactors

- Deconvolution with all cells
  - The assumption is that most genes are not differentially expressed (DE) between cells,
- Deconvolution within clusters (FastCluster beforehand)
  - Size factors computed within each cluster and rescaled by normalization between clusters.
  - When many genes are DE between clusters in a heterogeneous population.
- computeSumFactors – will also remove low abundance genes

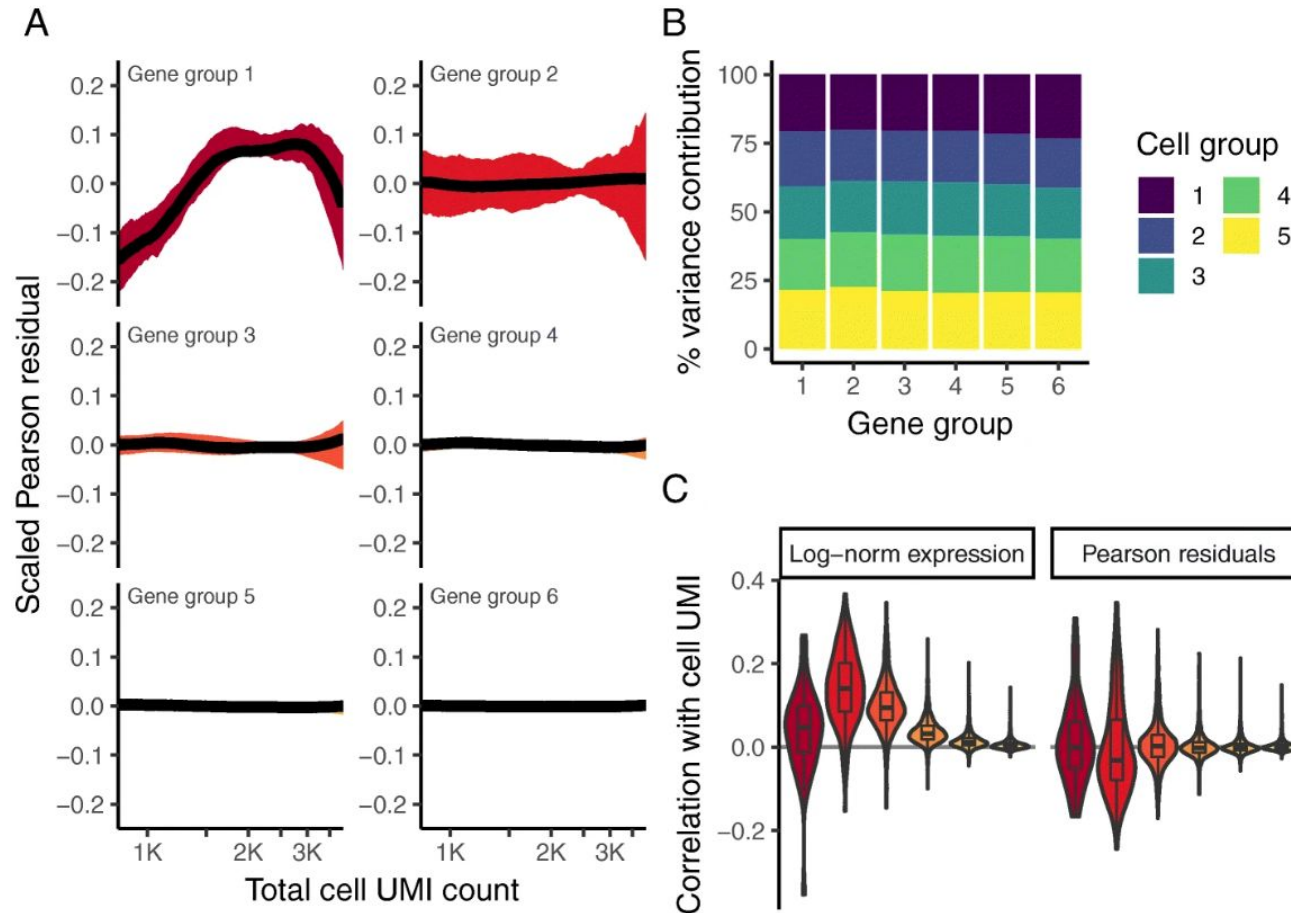
# Normalization with gene groups

- Global scale factors may lead to overcorrection for weakly and moderately expressed genes and undercorrection for highly expressed genes.
- It will also differ a lot between cells with high/low total counts.
- Solution: Do normalization for genes at different expression levels – SCNorm & SCTransform

# SCTransform (Seurat)



# SCTransform (Seurat)



Pearson residuals from regularized negative binomial (NB) regression

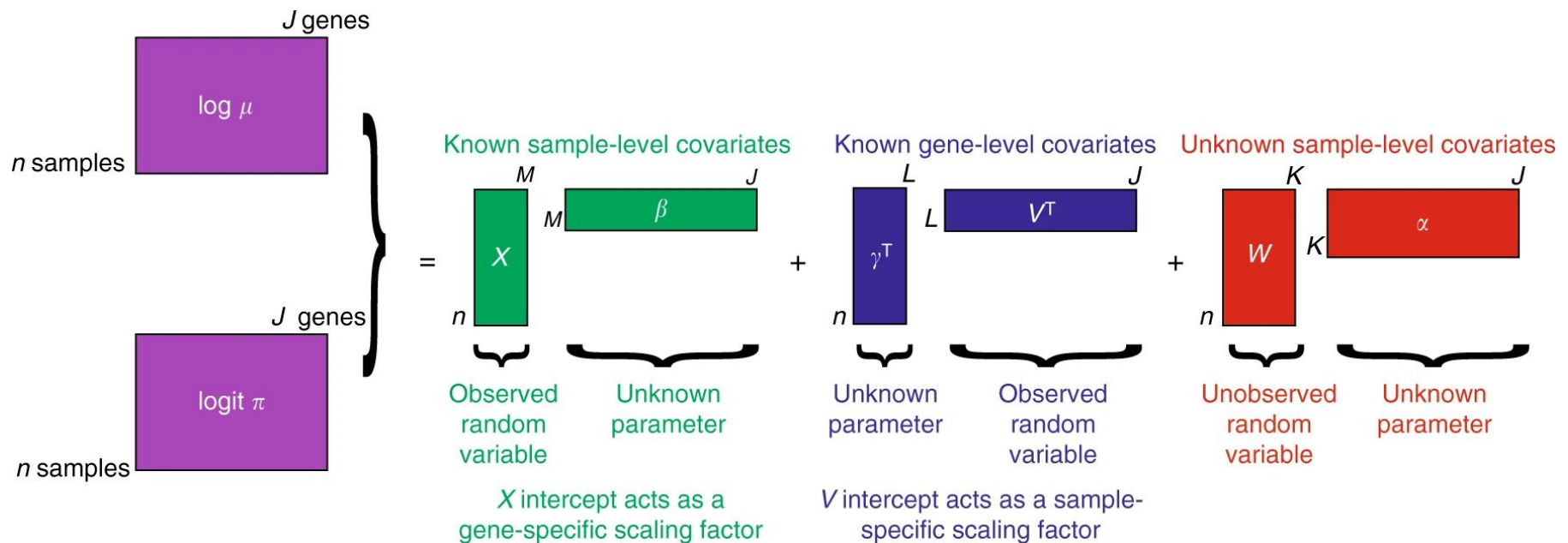
# SCTransform (Seurat)

- OBS! SCTransform function in Seurat also does variable gene selection in the same step with a slightly different method than the default in Seurat.
- But you can also specify which genes to run it on.
- You can also run regression of other parameters in the same step.
- Should be run per sample not with all data together.



# Zero-Inflated Negative Binomial-based Wanted Variation Extraction (ZINB-WaVE) - NewWave.

- Normalization and batch correction in one go
- Both gene-level and sample-level covariates
- Extension of the RUV model



# Comparison of transformations for single-cell RNA-seq data

Raw counts

$Y$

Delta method

$\log(Y/s + 1)$

GLM residual

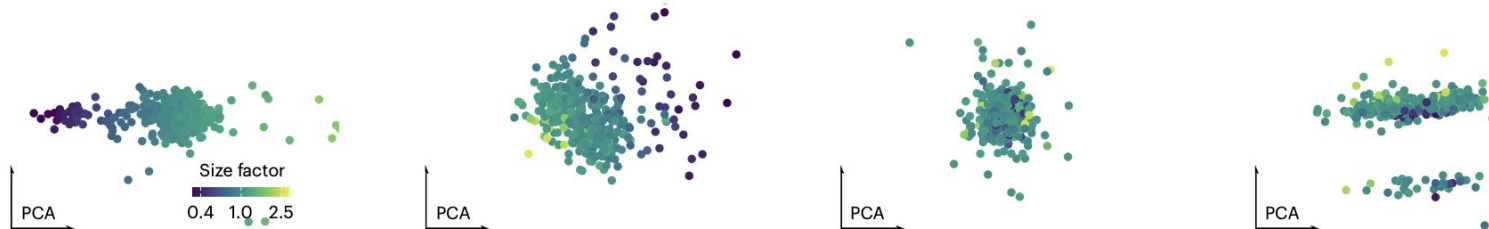
$$\frac{Y - \mu}{\sqrt{\mu + \alpha\mu^2}}$$

Latent expression

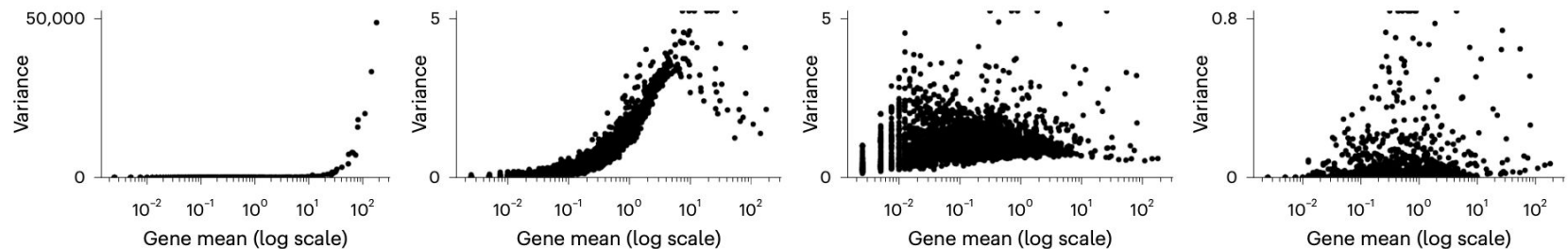
$Y \sim \text{Poisson}(M)$

$M \sim \text{logNormal}(\mu, \sigma^2)$

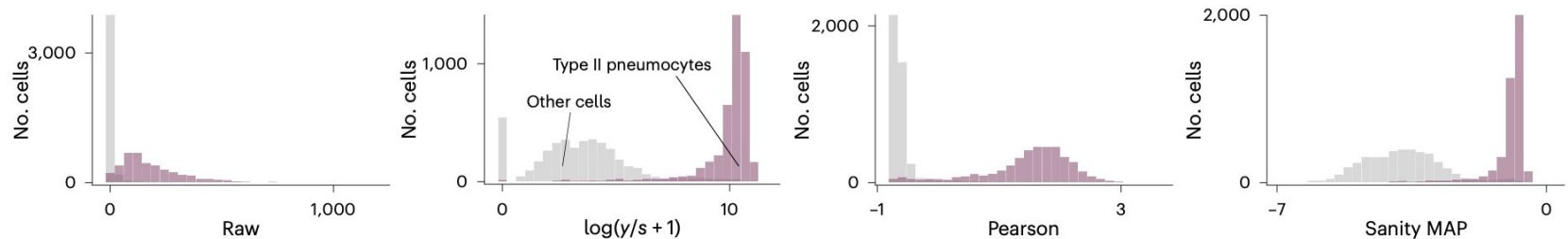
**a** Confounding effect of size factors on PCA embedding of droplets encapsulating a homogeneous RNA solution



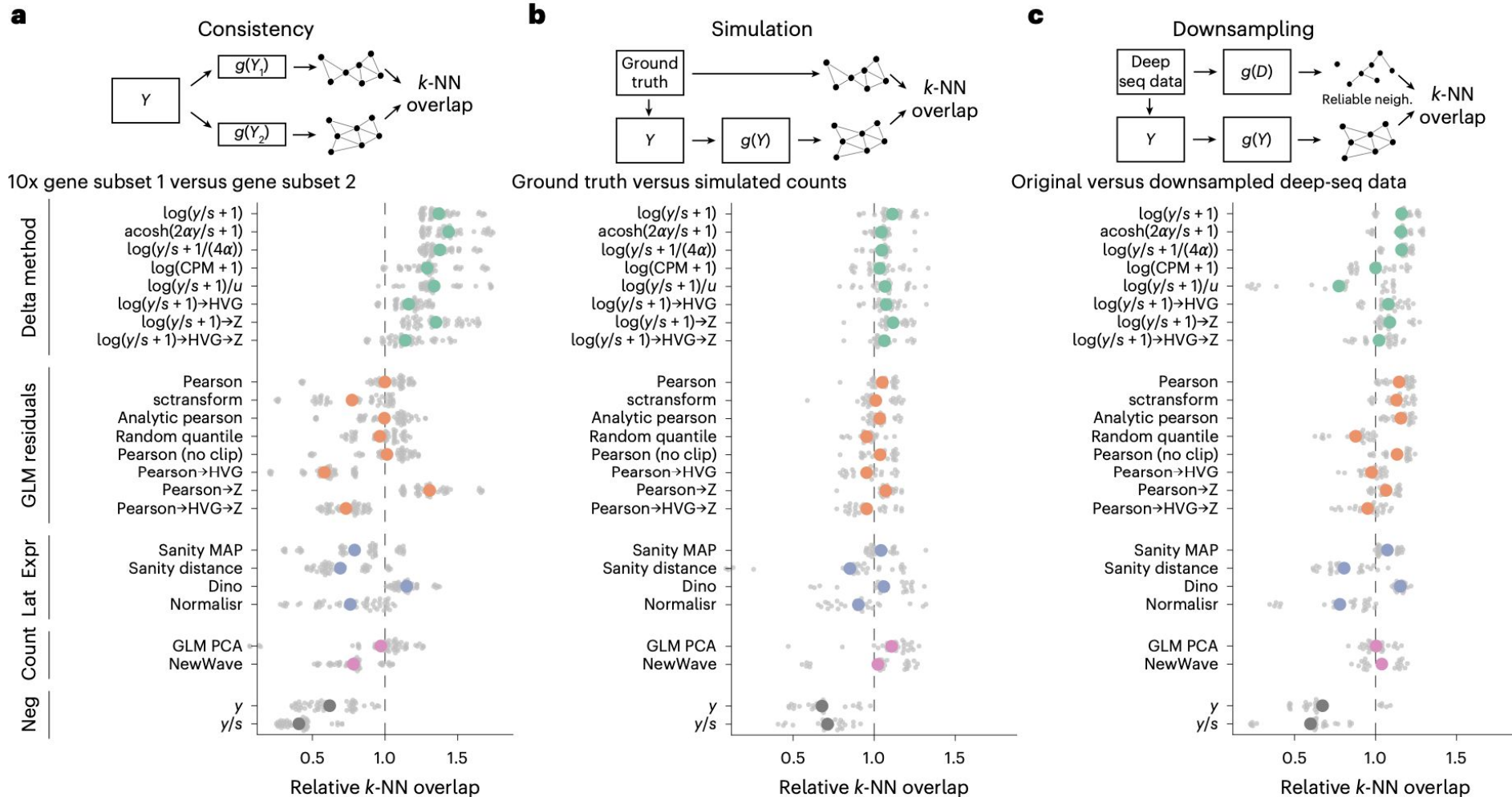
**b** Mean-variance relation for 2,597 genes of the 10x hematopoietic cell dataset



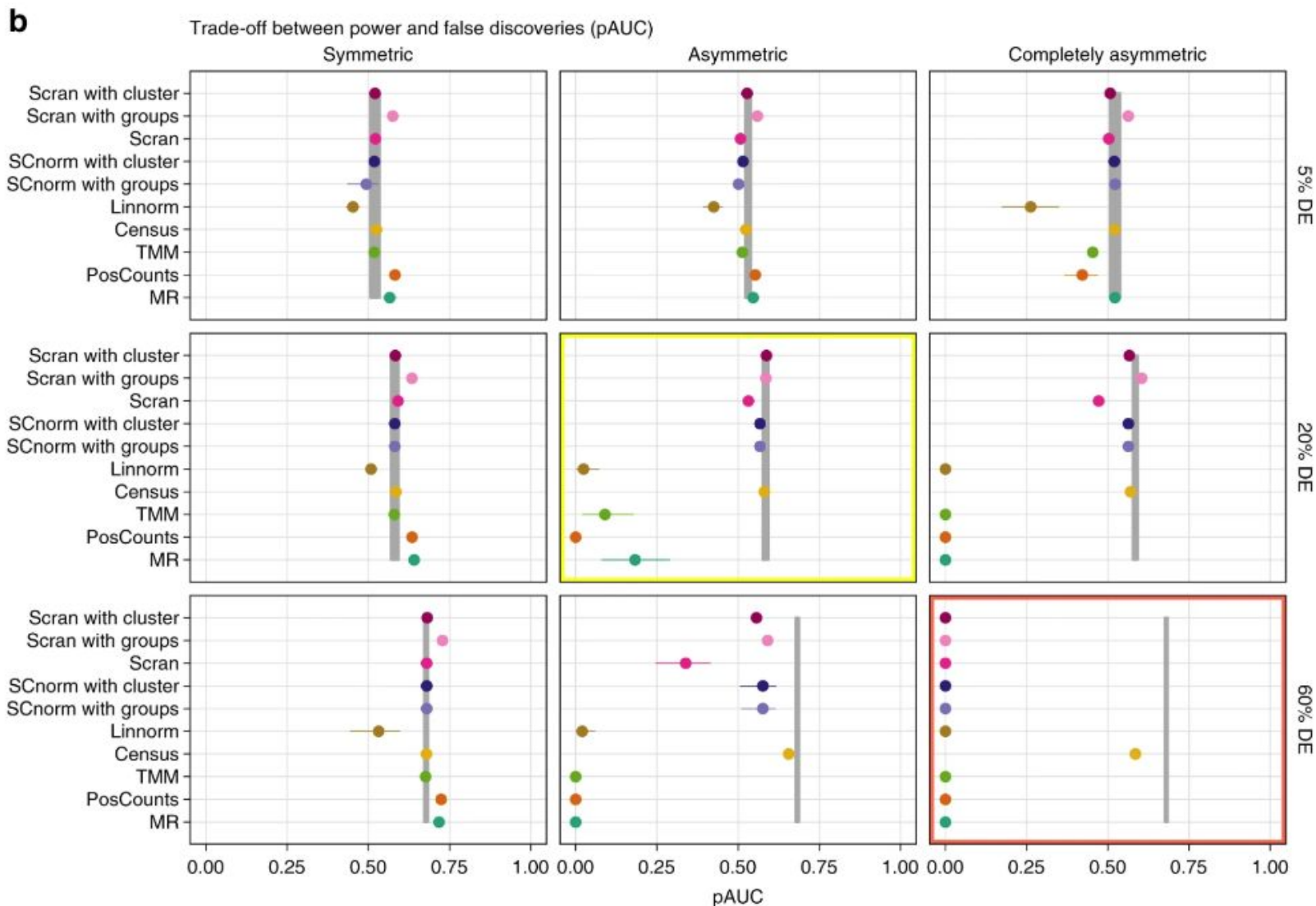
**c** Distribution of a single gene (*Sftpc*) with a bimodal expression pattern in lung epithelium



# Comparison of transformations for single-cell RNA-seq data



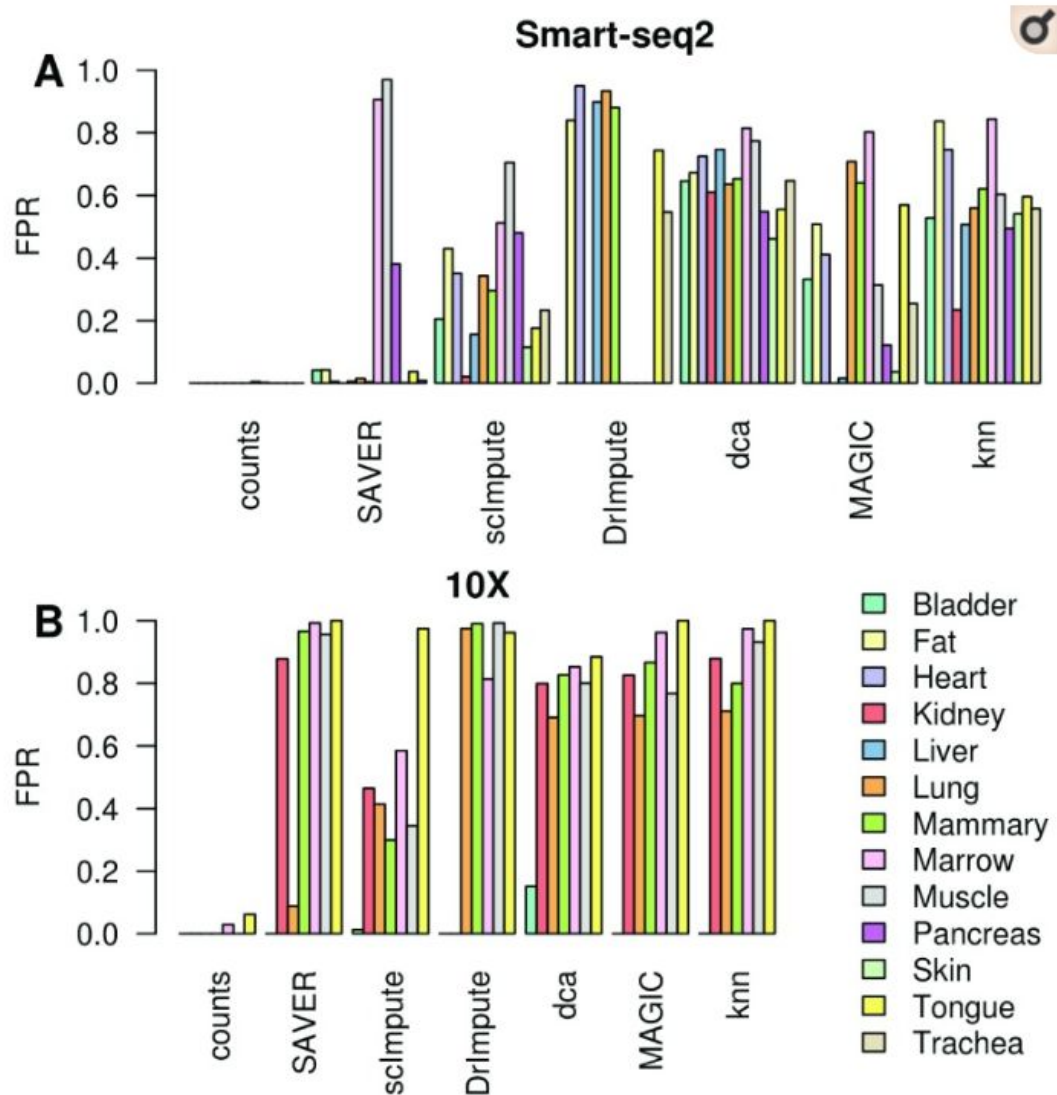
# DE with different normalizations



# Imputation

- scRNAseq has a lot of zeros in expression matrix
- Common for GWAS data to impute SNPs
- Many methods published:
  - SAVER
  - DrImpute
  - scImpute
  - MAGIC
  - Knn-smooth
  - Deep count autoencoder

# Imputation can introduce false correlations



# Scaling data – Z-score transformation

- Z-score transformation - linearly transform data to a mean of zero and a standard deviation of 1 - also called **centering** and **scaling**
- PCA or any other type of analysis will be dominated by highly expressed genes with high variance.
- It can be wise to center and scale each gene before performing PCA, some methods only do centering.



# What normalization should you use?

- Normalization has big impact on differential gene expression, but not as much on clustering
- **In most cases it is enough to do sequence depth normalization and log-transformation.**
- When working with highly similar subtypes of the same celltype, or with celltypes of very different sizes, individual size factors could help.
- Binning by gene level (SCTransform) helps to remove the effect of different gene detection across cells.

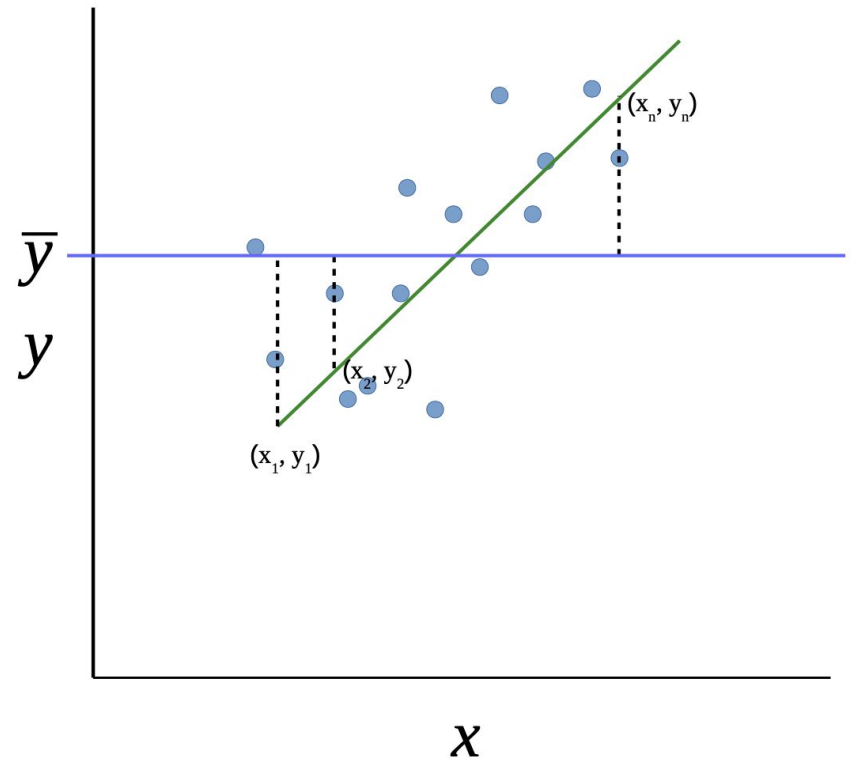


# Confounding factors

- Any source of variation that you do not expect to give separation of the cell types.
  - Cell cycle
  - Cell size
  - Sequencing depth
  - Cell quality
  - Batch
  - More...

# Linear regression

- Fit a line to the gene expression vs variable of interest
- Calculate residuals
- Remove variance explained by the variable of interest by taking the residuals.
- Multiple linear regression if multiple factors.



# Other tools to remove unwanted variance

- RUVseq() or svaseq()
- Linear models with e.g. removeBatchEffect() in limma or scater
- ComBat() in sva
- Tools like SCTransform, ZIMB-WaVE does regression in the same step.

# What confounders should you remove?

- Percent mitochondrial reads – often correlates with quality of cell
- Sequencing depth / nUMI
- Gene detection rate – relates to amount of RNA per cell.
- Cell cycle
- Batch effects (Sample, dataset, sort date, sex, etc.)
  - in most cases it is better to use an integration tool.

# What confounders should you remove?

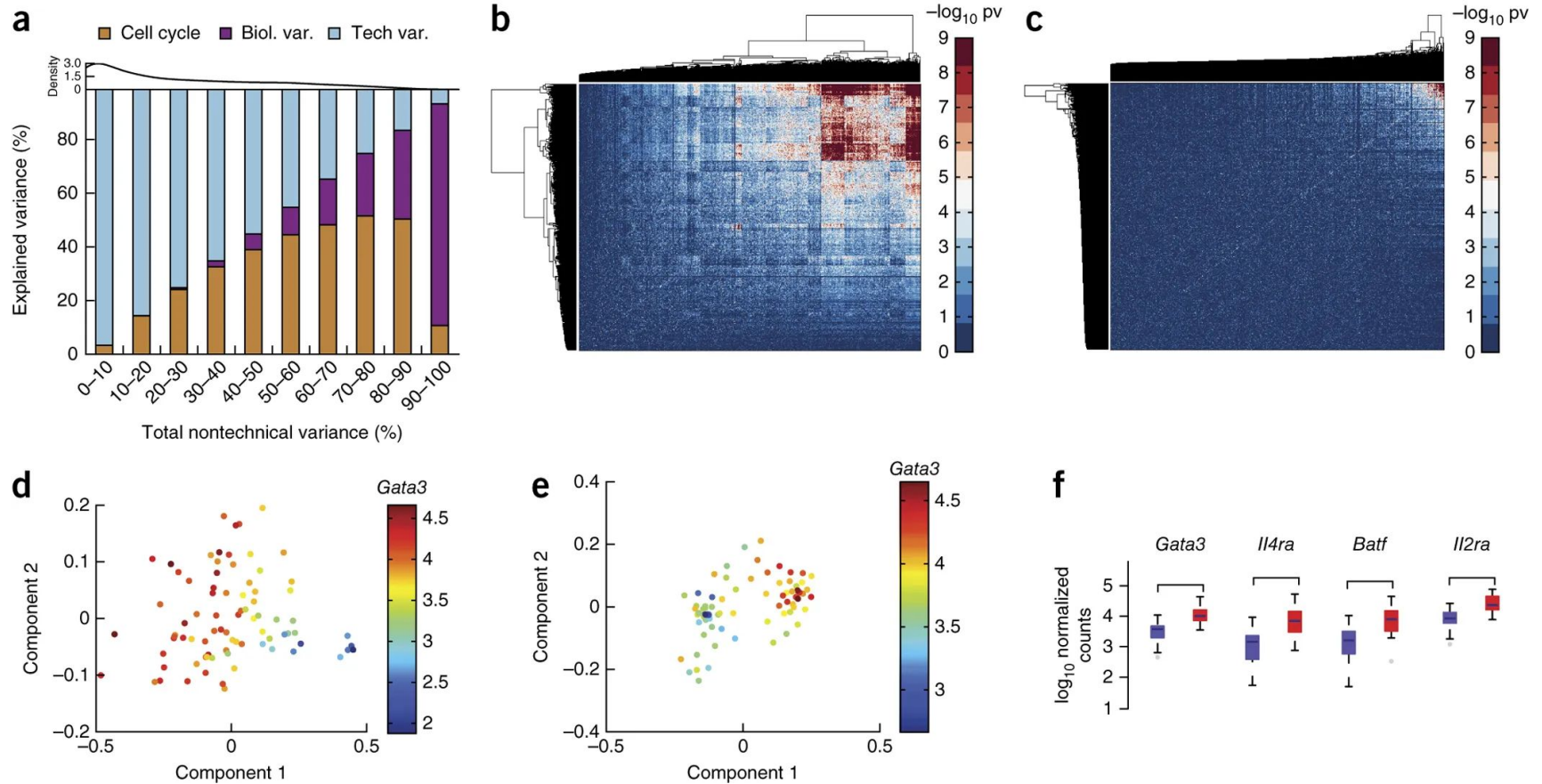
**ALWAYS** check QC parameters in PCA/tSNE/UMAP and see how they influence your data.

**BUT**, be careful that your confounders are not related to your biological question!

## Scaling and regression in practice

- Seurat **ScaleData**: does Z-score transformation and regression of variables in **vars.to.regress**. Can use **linear** (default), **poisson** or **negbinom** models.
- Scrn: runs scaling but not centering automatically in PCA step. **trendVar** function estimates unwanted variation either with a **design** matrix or with **block** factors. **decomposeVar** or **denoisePCA** to remove unwanted variation.
- Scanpy: **pp.regress\_out** and **pp.scale** functions.

# Cell cycle effect



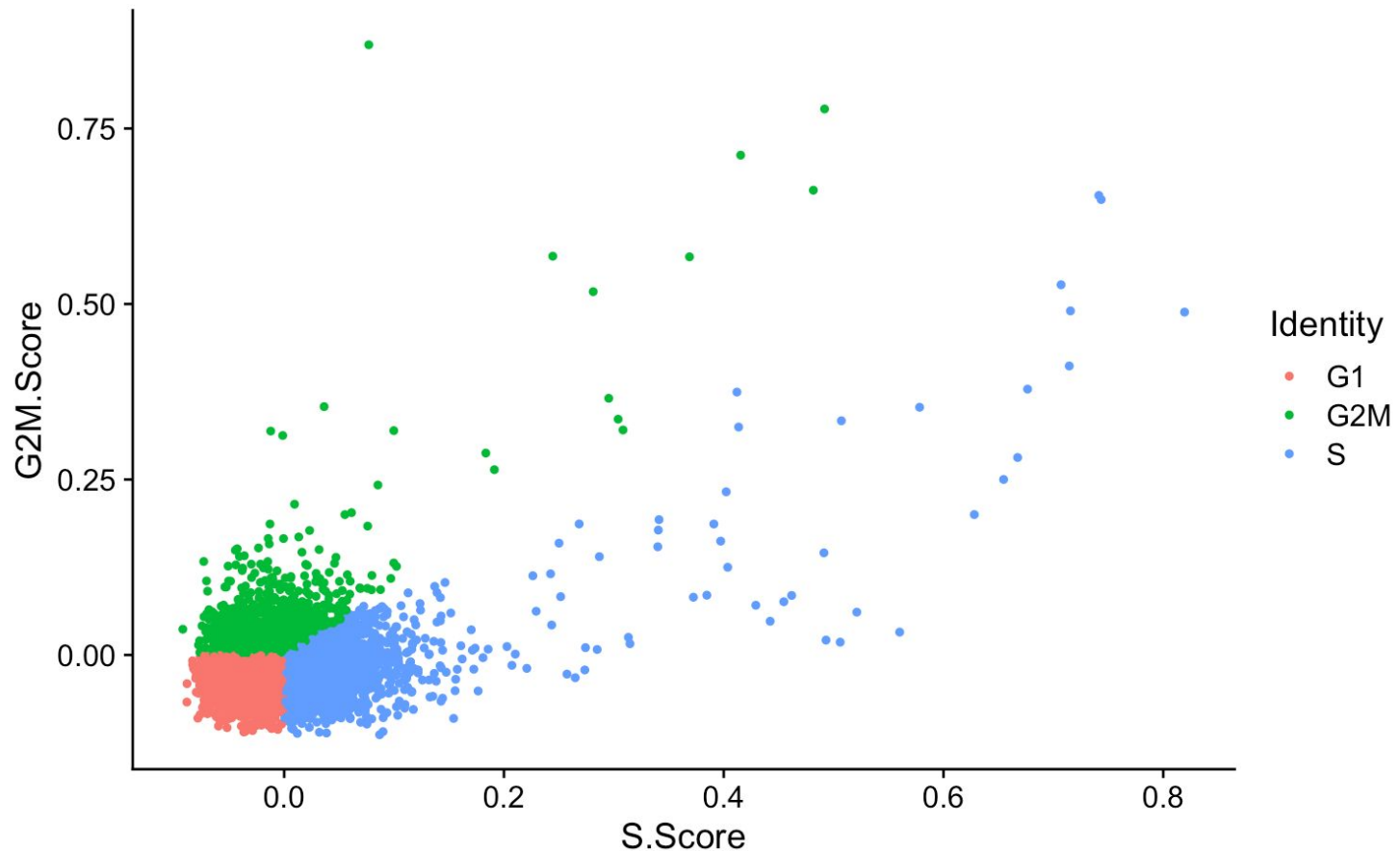
# Predict cell cycle stage / scores

- Seurat – **CellCycleScoring** – builds on G2M- & S-phase human gene lists from Tirosh et al. paper
- Scrn – **cyclone** function – trained on mouse cell cycle sorted cells. Uses relative expression of pairs of genes.
- Scanpy - **tl.score\_genes\_cell\_cycle** – uses same gene list as Seurat



# OBS! Seurat/Scanpy "Phase" predictions use a fixed cutoff.

0.44



```
FeatureScatter(data, "S.Score", "G2M.Score", group.by =  
"Phase")
```

# Cell cycle removal

- Regression on cell cycle scores.
  - Either with S.Score and G2M.Score
  - Or with  $\text{Diff} = \text{S.Score} - \text{G2M.Score}$
- scLVM - Designed for cell-cycle variation correction. Also has correction of other confounding variables.
- ccRemover (stable version from CRAN). “ccRemover outperforms scLVM slightly.”
- Oscope
- reCAT

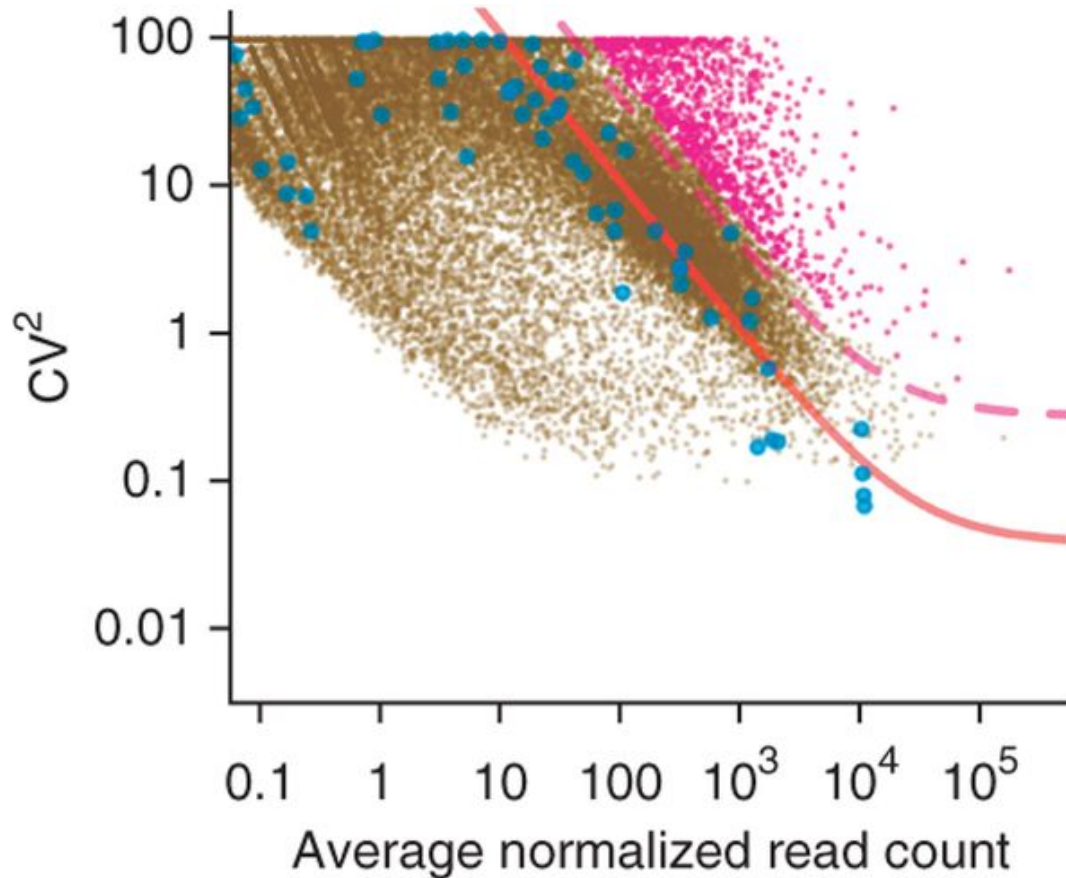
# Selecting genes

- Excluding invariable genes that do not contribute informative/interesting information
  - Improved signal to noise ratio
  - Reduced computational requirements
- Highly variable genes (HVGs)
- Correlated gene pairs/groups
- Top PCA loadings

# Variable gene selection

- **Genes which behave differently from a null model describing technical noise**
  - Mean-variance trend: genes with higher than expected variance
  - Coefficient of variation (Brennecke et al. 2013)
- **High dropout genes**
  - Number of zeros unexpectedly high compared to null model

# Highly variable genes (HVGs)



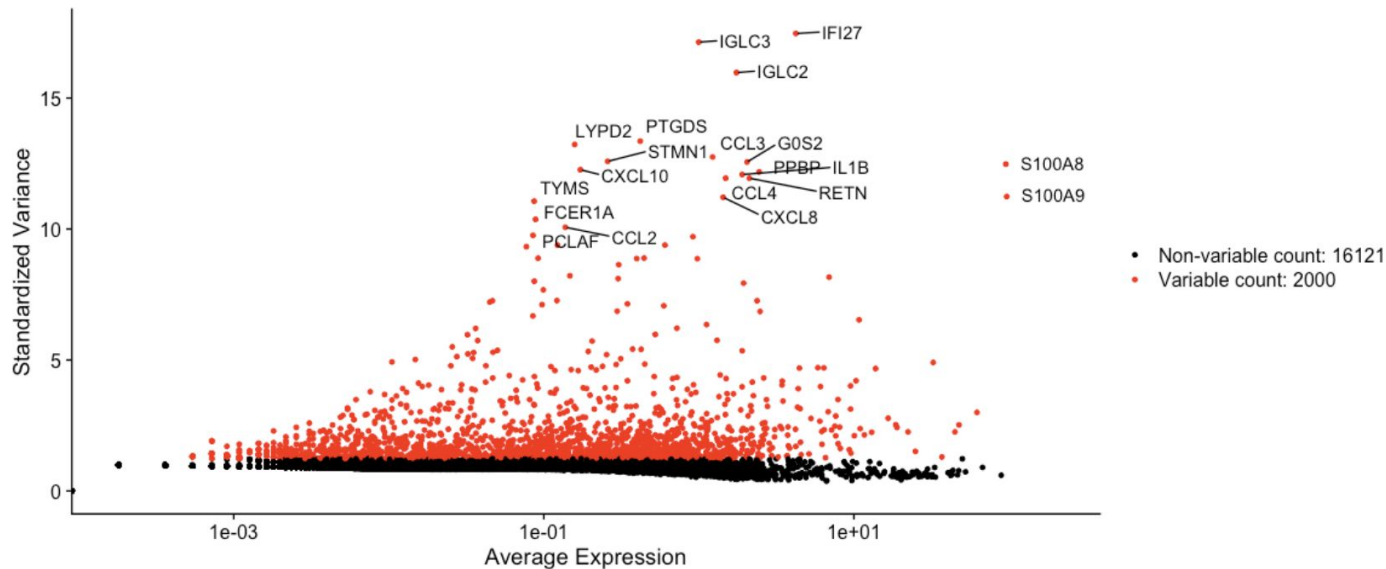
$$CV = \frac{var}{mean} = \frac{\sigma}{\mu}$$

Fit a gamma  
generalized linear  
model

No ERCCs?  
-> estimate technical  
noise based on  
all genes

# Variable gene selection in practise:

- Seurat: FindVariableFeatures - default **vst** method
- Fits a line to the relationship of  $\log(\text{variance})$  and  $\log(\text{mean})$  using local polynomial regression (loess). Then standardizes the feature values using the observed mean and expected variance. Feature variance is then calculated on the standardized values after clipping to a maximum.



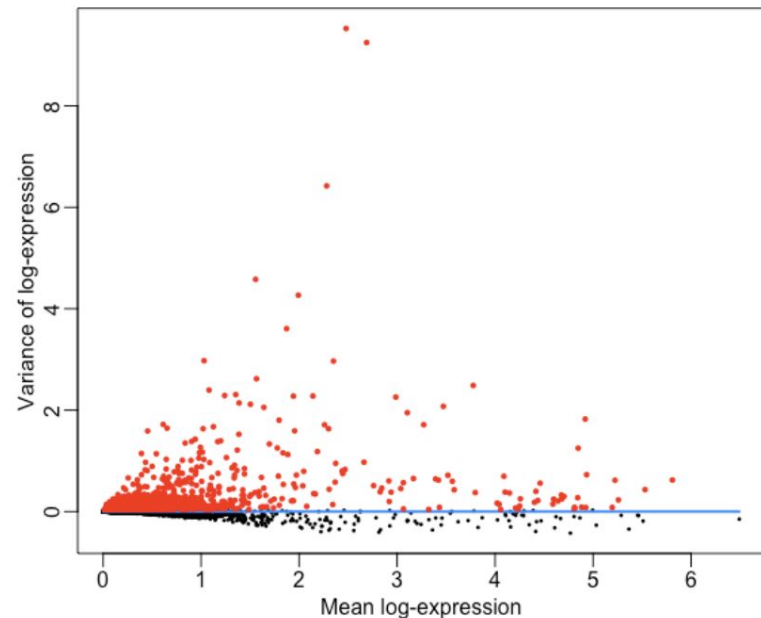
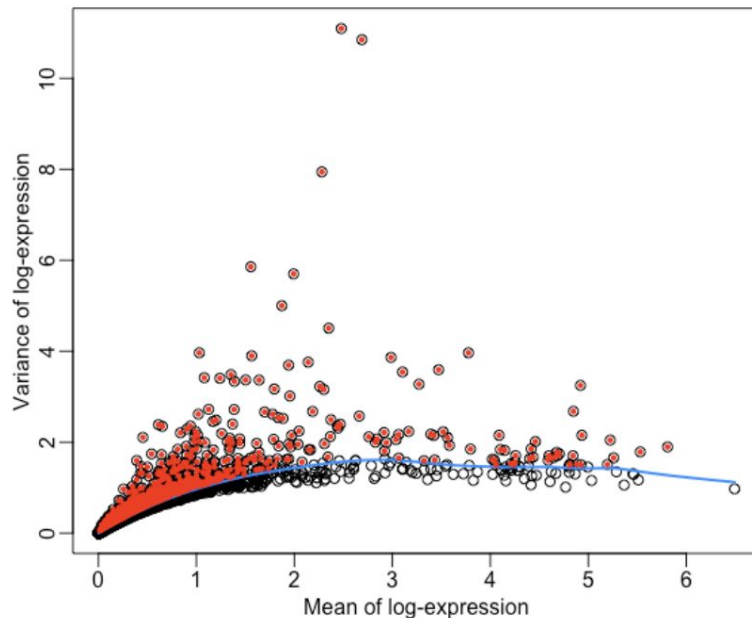
## Variable gene selection in practise:

**NOTE!** If you run **SCTransform** on a Seurat object it will automatically run HVG selection with its own method (based on Pearson residuals)

**vst** uses the raw counts to define the variance.

# Variable gene selection in practise:

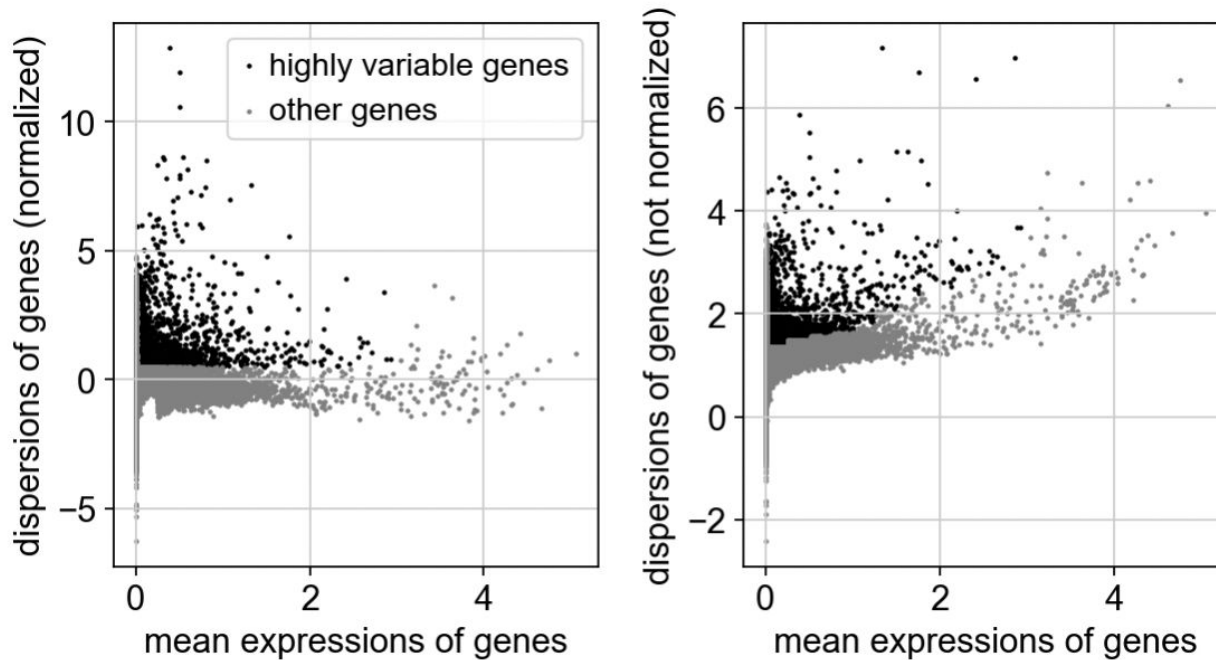
- Scran: ModelGeneVar & getTopHVGs
- Model the variance of the log-expression profiles for each gene, decomposing it into technical and biological components based on a fitted mean-variance trend.
- Can include blocking parameters in the design.





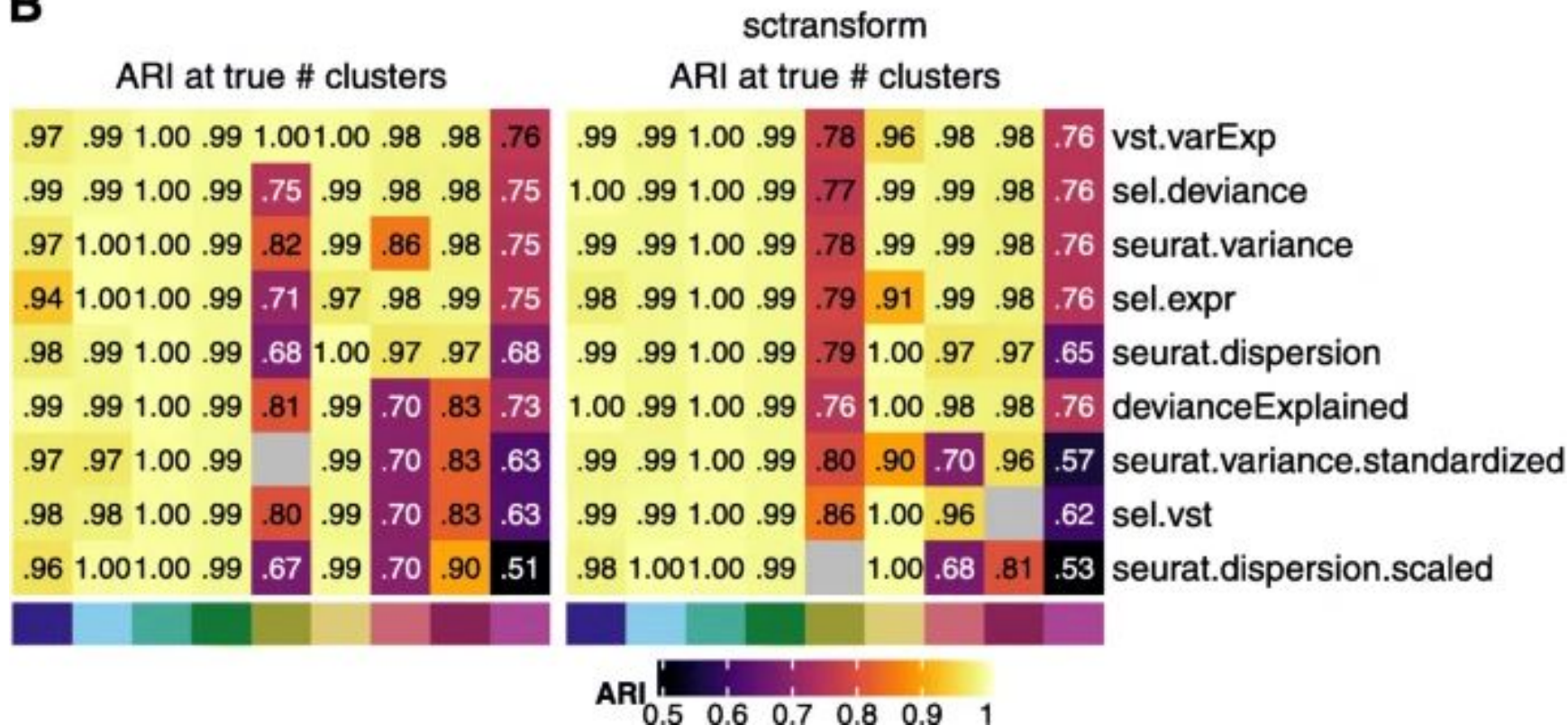
# Variable gene selection in practise:

- Scanpy: `sc.pp.highly_variable_genes`
- Implements same methods as Seurat
- Can specify “batch\_key” and calculate per batch then combine the values.

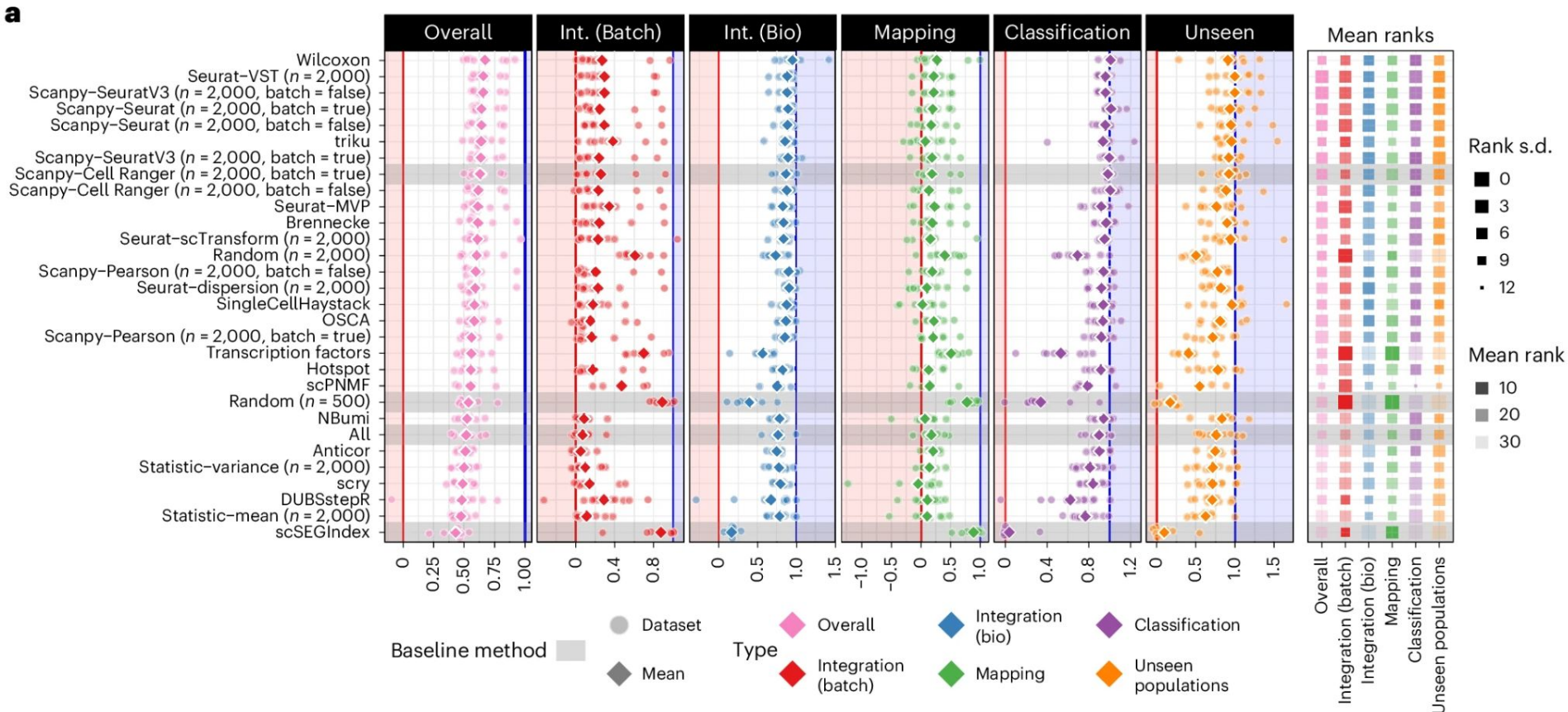


# Evaluation of HVGs for clustering

**B**



# Evaluation of HVGs for integration



# How many genes should you choose?

Too few variable genes:

- Do not capture all biological signal.

Too many variable genes:

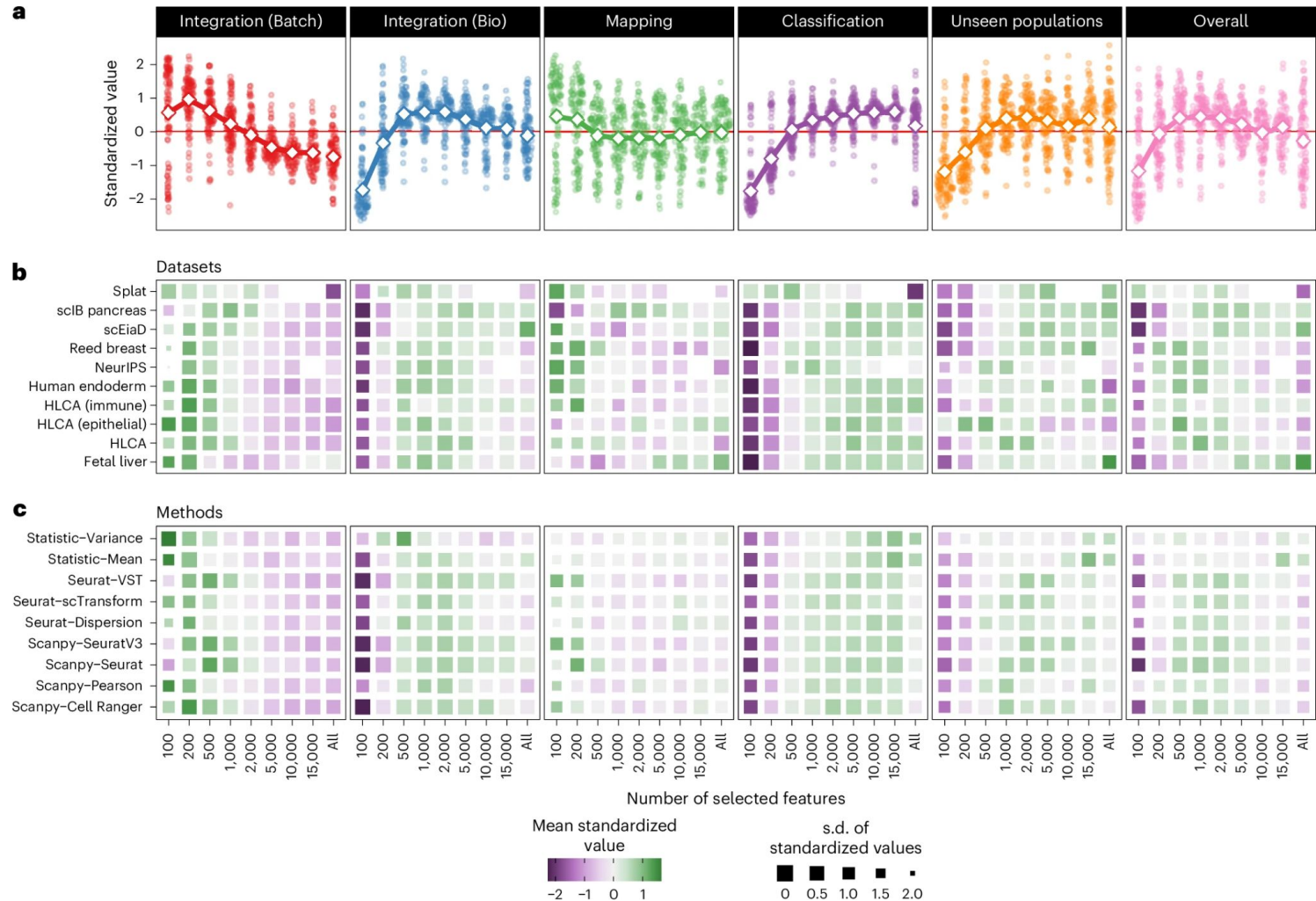
- Risk of also including technical noise.
- Higher computational cost.

No number fits all datasets!

- Single celltype - use few HVGs (1000-2000)
- Many celltypes - use more HVGs (3000-5000)



# How many genes should you choose?



(Zappia et al. *Nature Methods* 2025)

# Conclusions

- Normalization has impact on differential gene expression.
- Many different methods to remove unwanted variance – often an important step!
- Selection of variable genes is important to remove noise in the data. Always subset genes before running PCA/clustering.
- Always aim for same sequencing depth in all samples – to avoid at least one confounding factor.

# Do not worry!

If you have distinct celltypes – the clustering will be similar regardless of how you treat the data.

But, for subclustering of similar celltypes normalization and removal of confounders may be crucial.