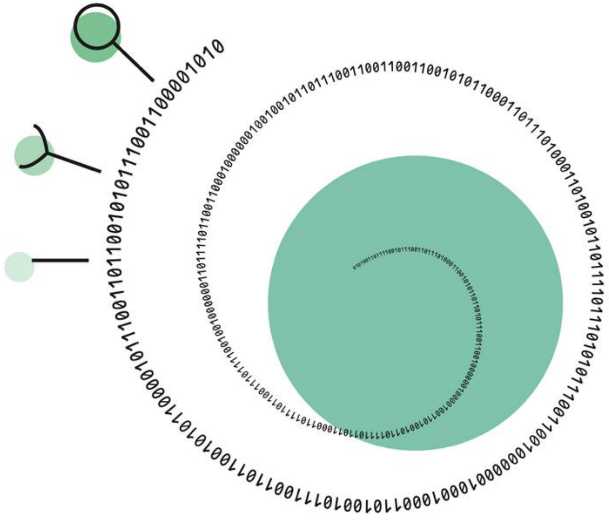


Single Dell Data Analysis Course

Data preprocessing

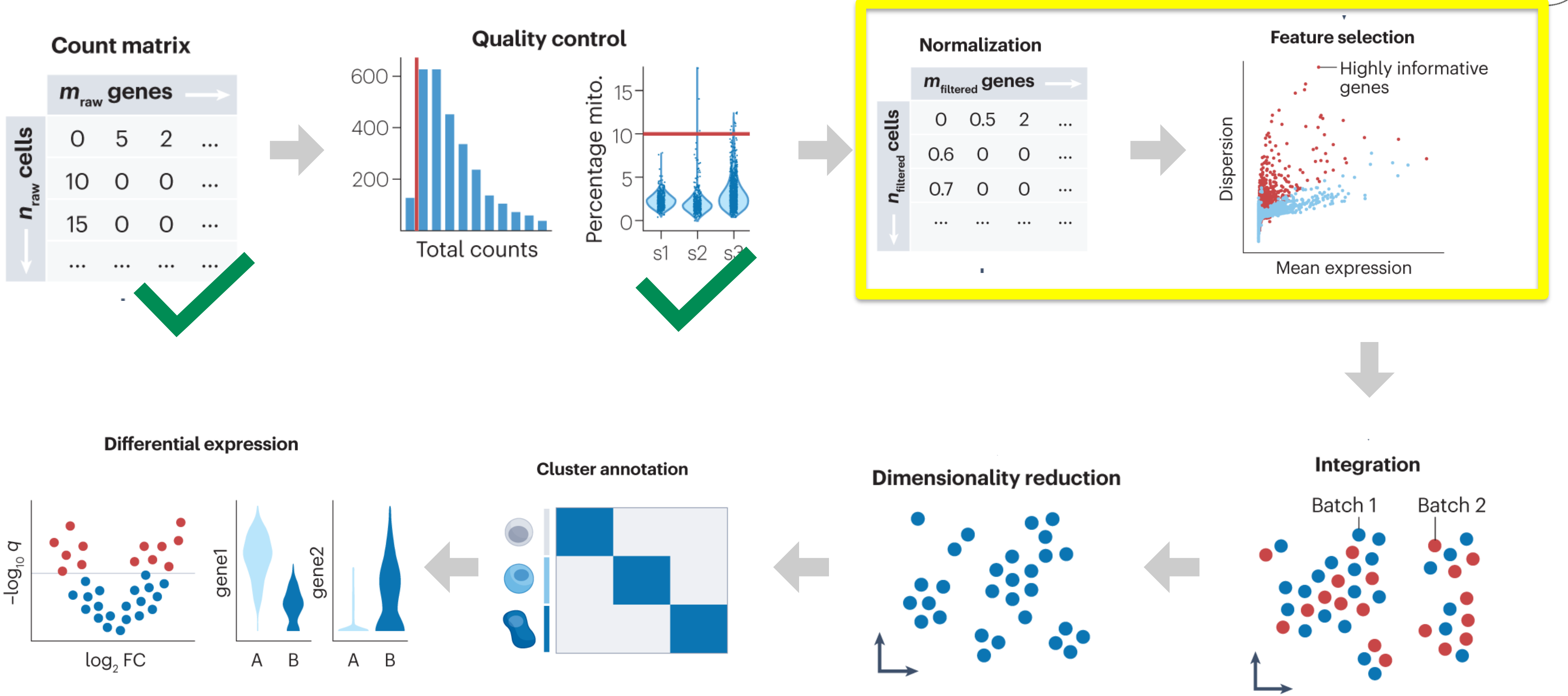
Lisa Buchauer

Professor of Systems Biology of Infectious Diseases

Department of Infectious Diseases and Intensive Care

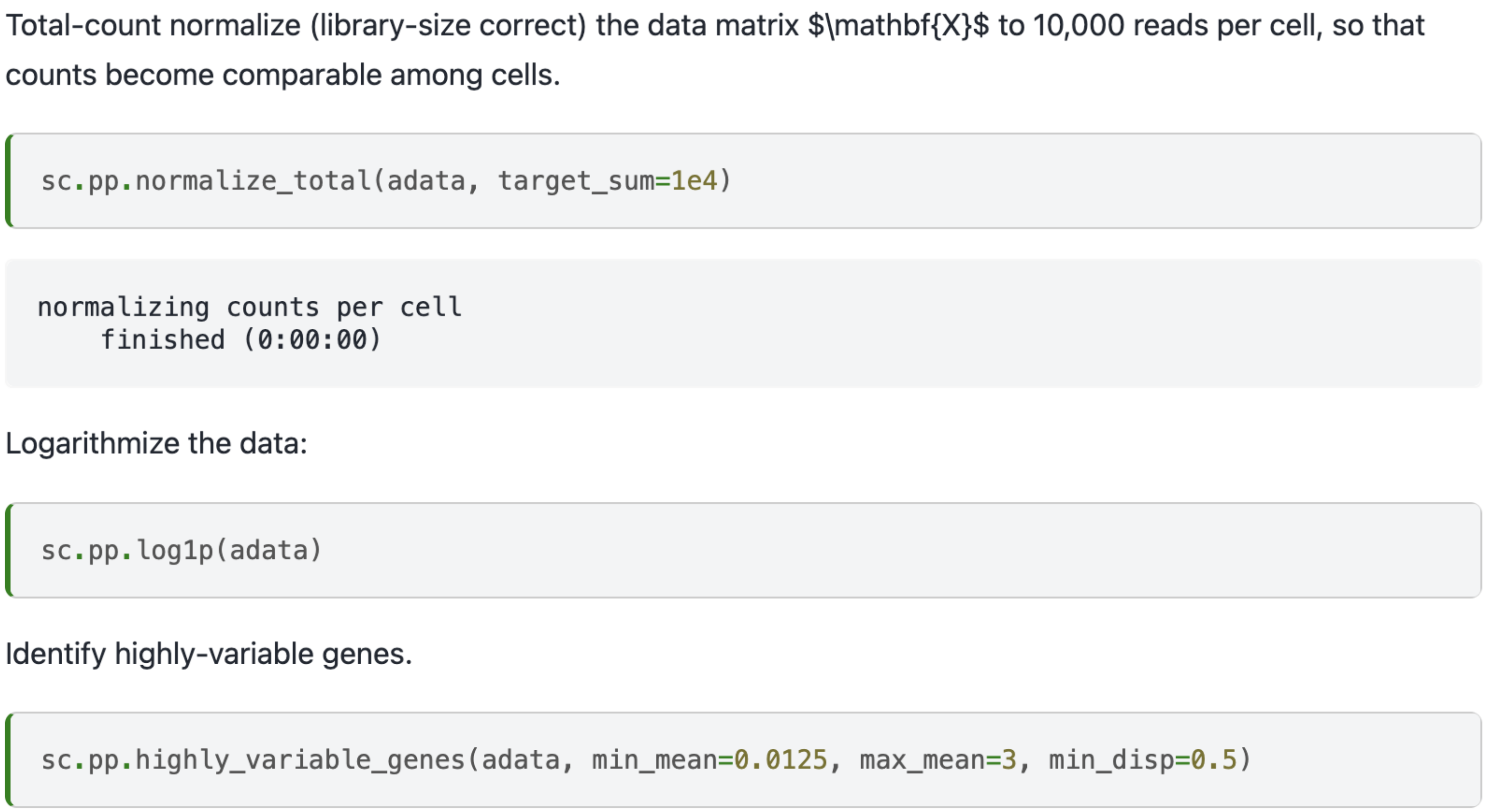
Charité - Universitätsmedizin Berlin

Processing overview



Heumos, L., Schaar, A.C., Lance, C. et al. Best practices for single-cell analysis across modalities. Nat Rev Genet 24, 550–572 (2023). https://doi.org/10.1038/s41576-023-00586-w

Three important lines of code



Total-count normalize (library-size correct) the data matrix $\mathbf{X}$ to 10,000 reads per cell, so that counts become comparable among cells.

1

```
sc.pp.normalize_total(adata, target_sum=1e4)
```

```
normalizing counts per cell
    finished (0:00:00)
```

Logarithmize the data:

2

```
sc.pp.log1p(adata)
```

Identify highly-variable genes.

3

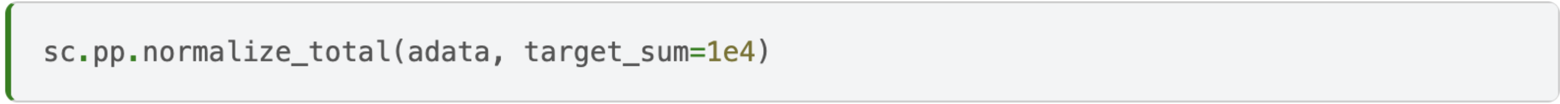
```
sc.pp.highly_variable_genes(adata, min_mean=0.0125, max_mean=3, min_disp=0.5)
```

https://scanpy-tutorials.readthedocs.io/en/latest/pbmc3k.html

1

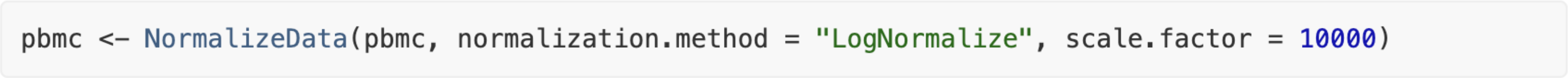
Normalizing data to mitigate library size effects

scanpy



```
sc.pp.normalize_total(adata, target_sum=1e4)
```

Seurat

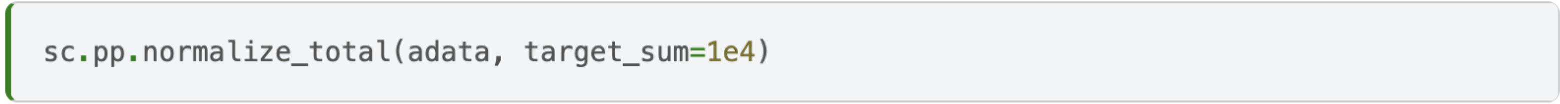


```
pbmc <- NormalizeData(pbmc, normalization.method = "LogNormalize", scale.factor = 10000)
```

1

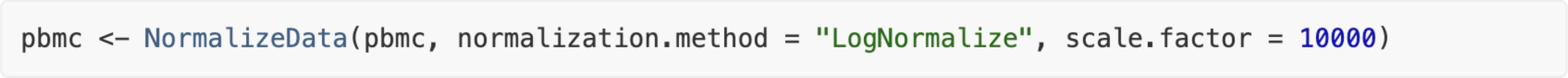
Normalizing data to mitigate library size effects

scanpy



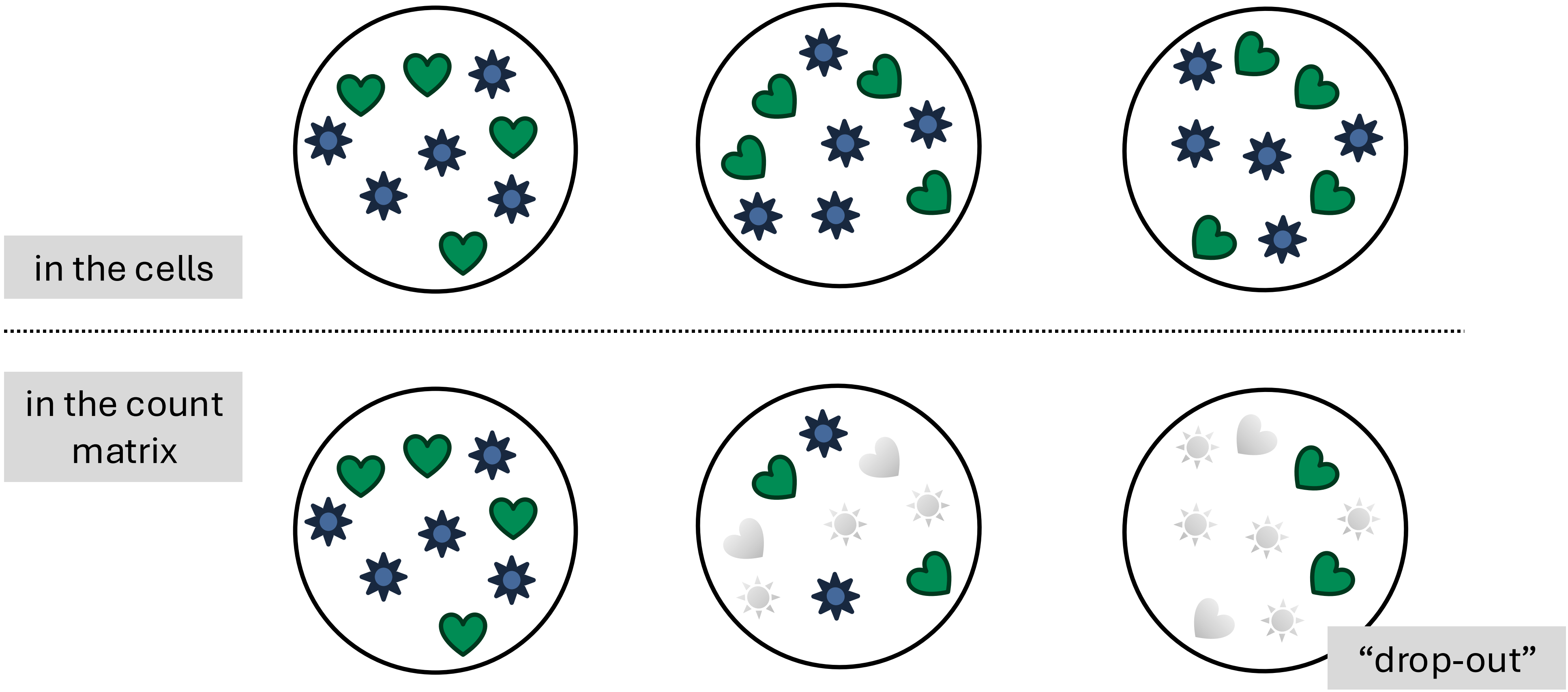
```
sc.pp.normalize_total(adata, target_sum=1e4)
```

Seurat



```
pbmc <- NormalizeData(pbmc, normalization.method = "LogNormalize", scale.factor = 10000)
```

Why?



in the cells

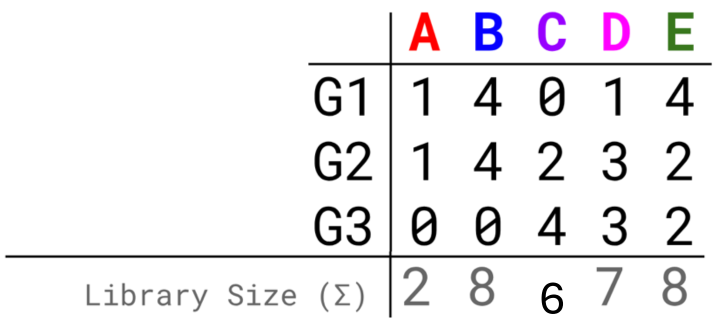
in the count matrix

“drop-out”

1

Normalizing data to mitigate library size effects

How?



	A	B	C	D	E
G1	1	4	0	1	4
G2	1	4	2	3	2
G3	0	0	4	3	2
Library Size (Σ)	2	8	6	7	8

target_sum = 10

(count / library size) x target sum

A|G1: (½)*10 = 5

B|G2: (4/8)*10 = 5

	A	B	C	D	E
G1	5	5	0	1.43	5
G2	5	5	3.33	4.29	2.5
G3	0	0	6.67	4.29	2.5
Sum	10	10	10	10	10

1

Normalizing data to mitigate library size effects

How?

	A	B	C	D	E
G1	1	4	0	1	4
G2	1	4	2	3	2
G3	0	0	4	3	2
Library Size (Σ)	2	8	6	7	8

target_sum = 10

(count / library size) x target sum

A|G1: (½)*10 = 5

B|G2: (4/8)*10 = 5

	A	B	C	D	E
G1	5	5	0	1.43	5
G2	5	5	3.33	4.29	2.5
G3	0	0	6.67	4.29	2.5
Sum	10	10	10	10	10

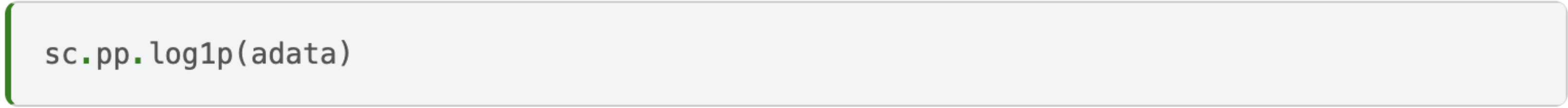
- 10k counts are often used as target
- Alternative: normalize to median library size of original data

- Relies on the assumption that every cell originally had the same amount of RNA
- ! Actual variation in count number may be due to both technical AND biological effects

2

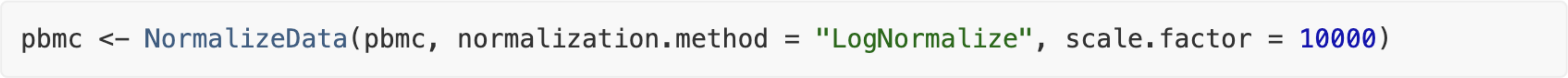
Taking the log to stabilize the variance

scanpy



```
sc.pp.log1p(adata)
```

Seurat

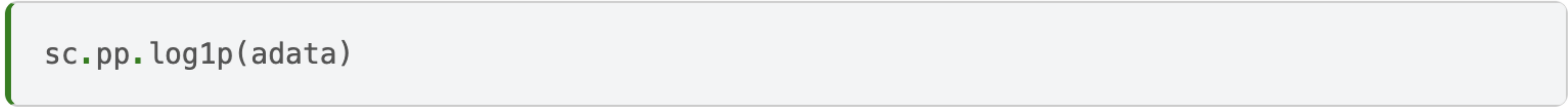


```
pbmc <- NormalizeData(pbmc, normalization.method = "LogNormalize", scale.factor = 10000)
```

2

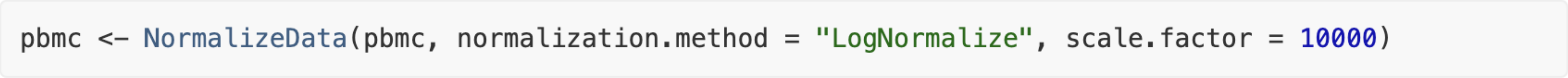
Taking the log to stabilize the variance

scanpy



```
sc.pp.log1p(adata)
```

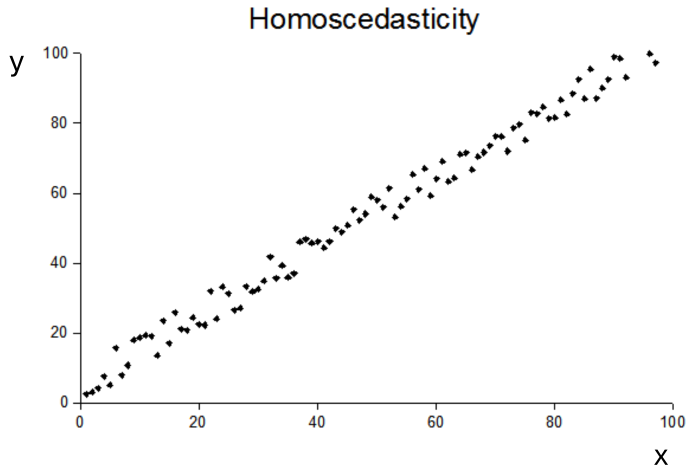
Seurat

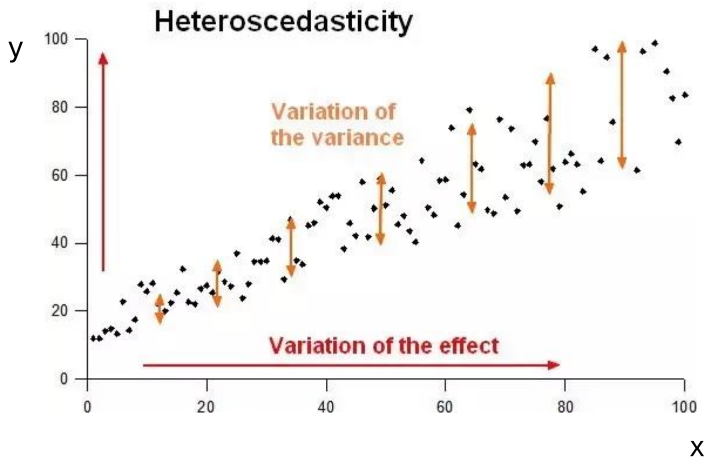


```
pbmc <- NormalizeData(pbmc, normalization.method = "LogNormalize", scale.factor = 10000)
```

Why?

Many downstream methods like identification of highly variable genes, dimension reduction and clustering require (or at least perform a lot better) with **homoscedastic** data.



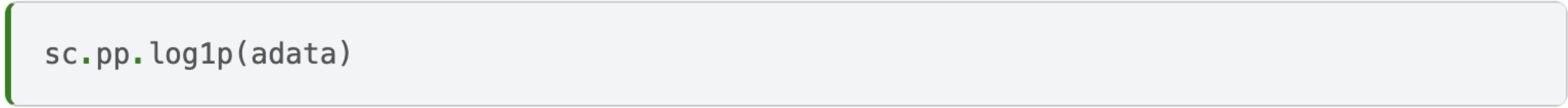


https://en.wikipedia.org/wiki/Homoscedasticity_and_heteroscedasticity

2

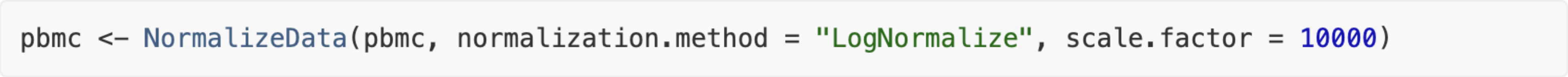
Taking the log to stabilize the variance

scanpy



```
sc.pp.log1p(adata)
```

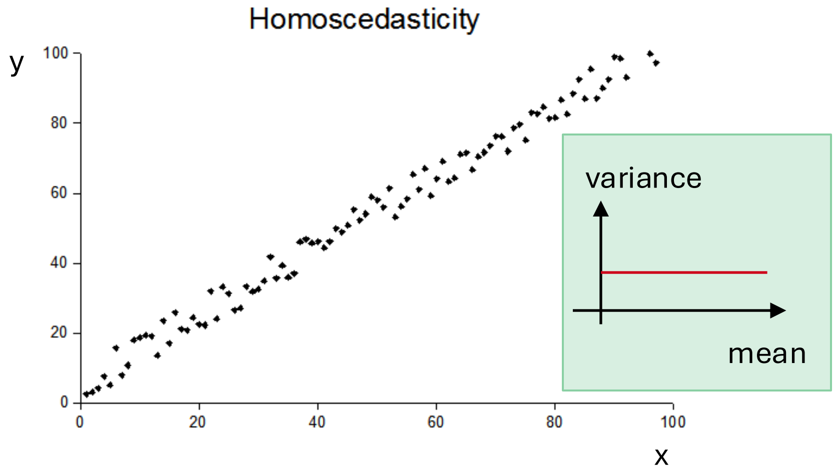
Seurat

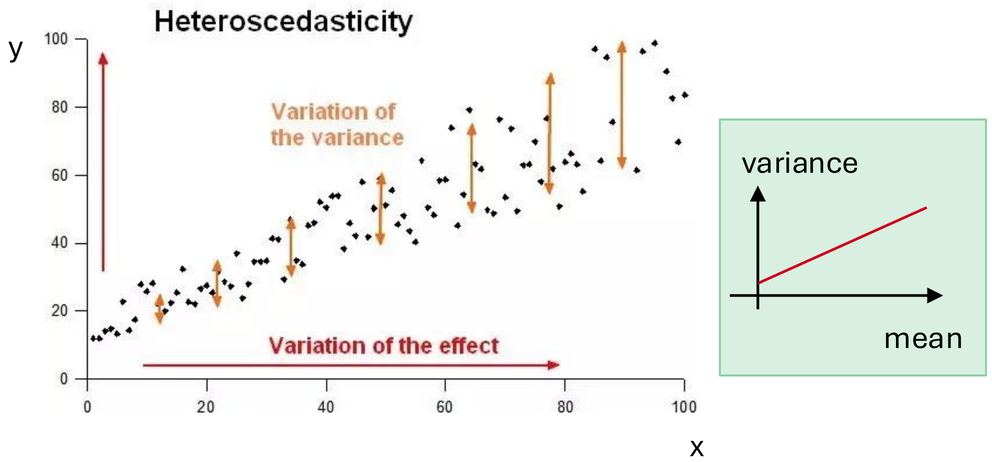


```
pbmc <- NormalizeData(pbmc, normalization.method = "LogNormalize", scale.factor = 10000)
```

Why?

Many downstream methods like identification of highly variable genes, dimension reduction and clustering expect (or at least perform a lot better with) **homoscedastic** data.



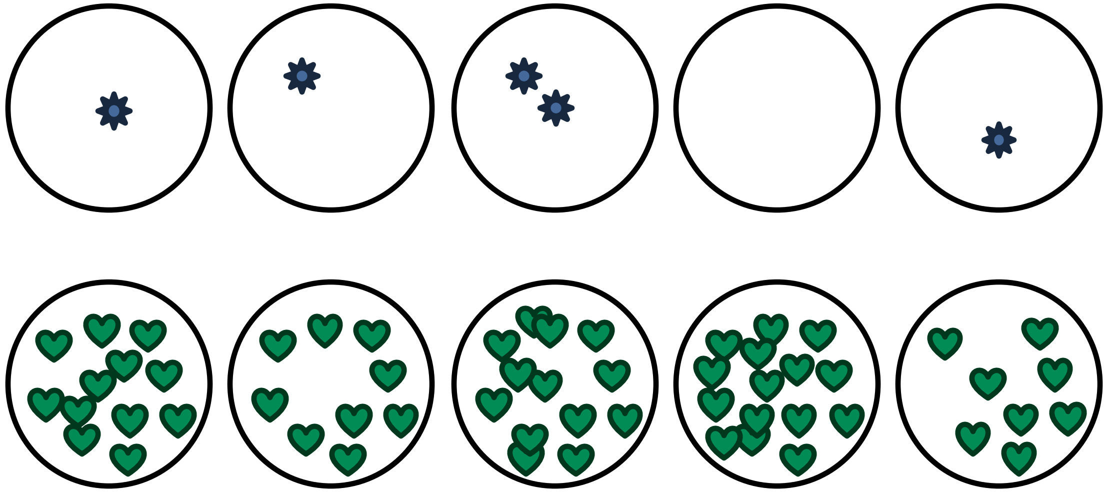


https://en.wikipedia.org/wiki/Homoscedasticity_and_heteroscedasticity

2

Taking the log to stabilize the variance

Why?



Lowly expressed gene

Mean = 1 count

Variance = 0.4 counts

Highly expressed gene

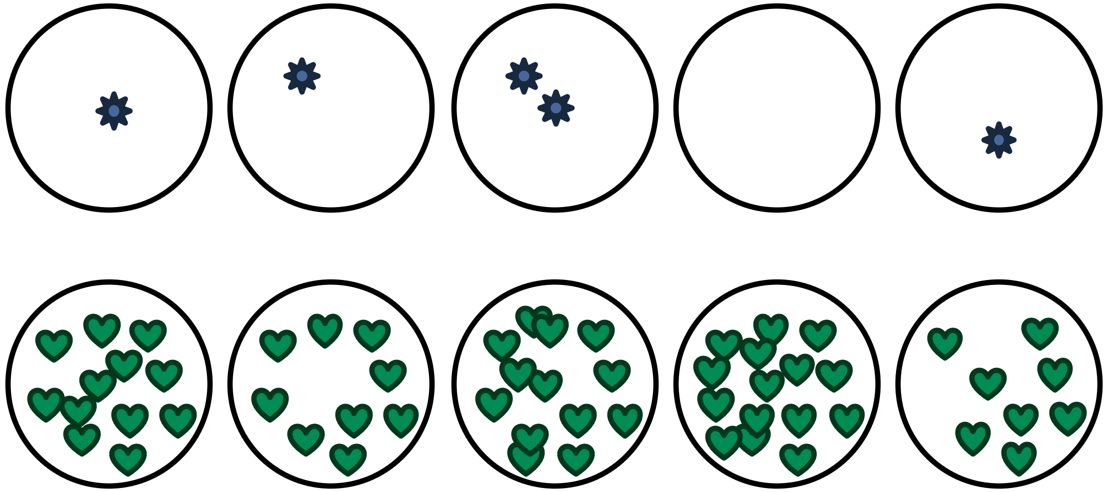
Mean = 11.4 count

Variance = 6.64 counts

2

Taking the log to stabilize the variance

Why?



Lowly expressed gene

Mean = 1 count

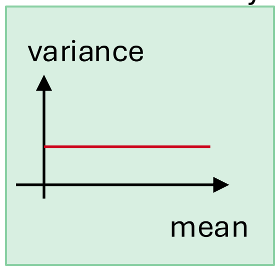
Variance = 0.4 counts

Highly expressed gene

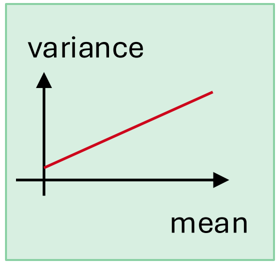
Mean = 11.4 count

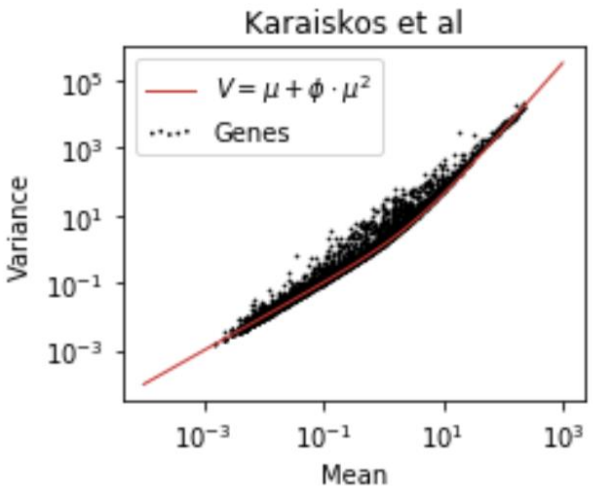
Variance = 6.64 counts

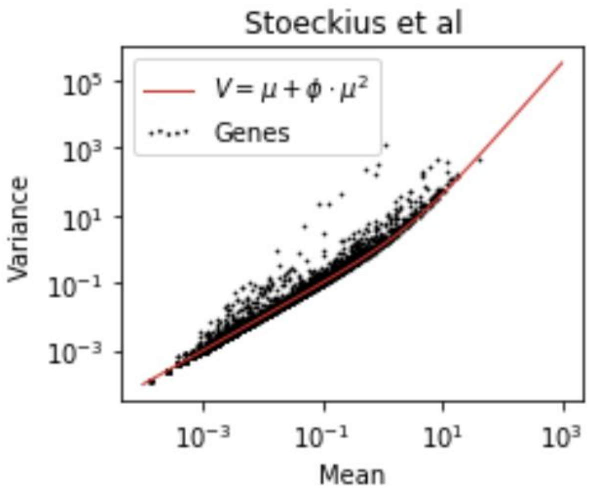
homoscedasticity



heteroscedasticity







Heteroscedastic!

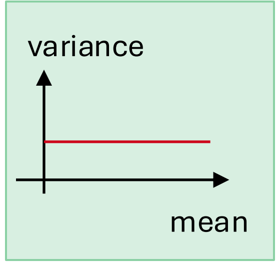
https://www.nxn.se/valent/2017/10/15/variance-stabilizing-scrna-seq-counts

2

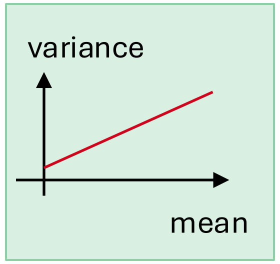
Taking the log to stabilize the variance

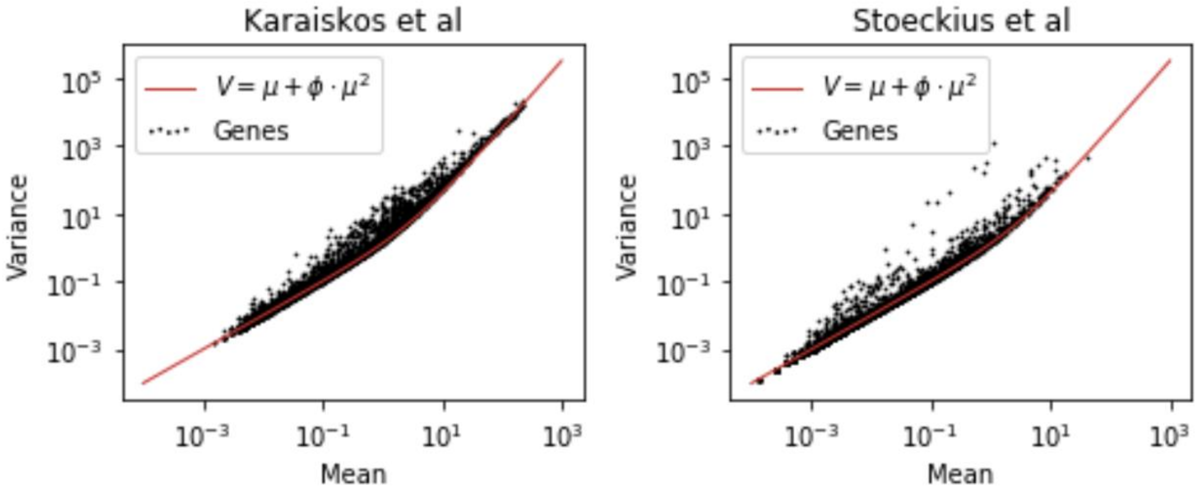
Why?

homoscedasticity

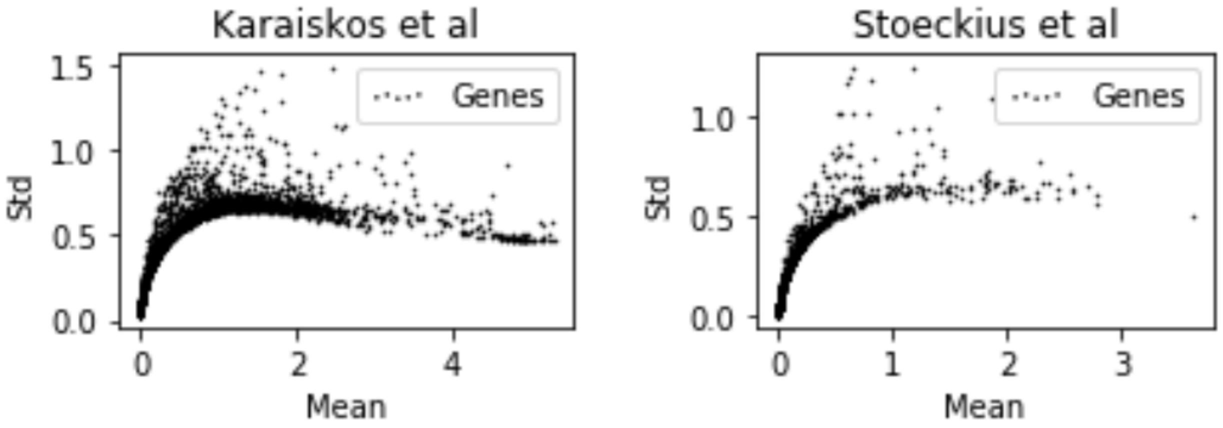


heteroscedasticity





log-transform counts



https://www.nxn.se/valent/2017/10/15/variance-stabilizing-scrna-seq-counts

2

Taking the log to stabilize the variance

How?

	A	B	C	D	E
G1	5	5	0	1.43	5
G2	5	5	3.33	4.29	2.5
G3	0	0	6.67	4.29	2.5
Sum	10	10	10	10	10

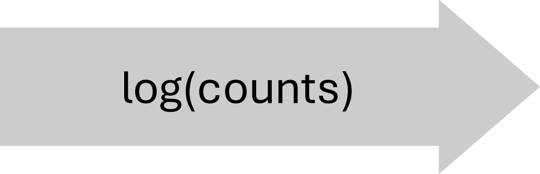
log(counts)

2

Taking the log to stabilize the variance

How?

	A	B	C	D	E
G1	5	5	0	1.43	5
G2	5	5	3.33	4.29	2.5
G3	0	0	6.67	4.29	2.5
Sum	10	10	10	10	10



log(counts)

Value Error

2

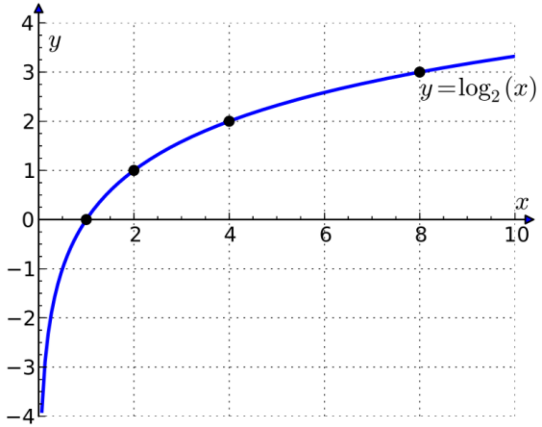
Taking the log to stabilize the variance

How?

	A	B	C	D	E
G1	5	5	0	1.43	5
G2	5	5	3.33	4.29	2.5
G3	0	0	6.67	4.29	2.5
Sum	10	10	10	10	10

log(counts)

Value Error



https://en.wikipedia.org/wiki/Binary_logarithm

2

Taking the log to stabilize the variance

How?

	A	B	C	D	E
G1	5	5	0	1.43	5
G2	5	5	3.33	4.29	2.5
G3	0	0	6.67	4.29	2.5
Sum	10	10	10	10	10

log(counts + 1)

	A	B	C	D	E
G1	1.8	1.8	0	0.9	1.8
G2	1.8	1.8	1.5	1.7	0.9
G3	0	0	2.0	1.7	0.9

2

Taking the log to stabilize the variance

How?

	A	B	C	D	E
G1	5	5	0	1.43	5
G2	5	5	3.33	4.29	2.5
G3	0	0	6.67	4.29	2.5
Sum	10	10	10	10	10

log(counts + 1)

	A	B	C	D	E
G1	1.8	1.8	0	0.9	1.8
G2	1.8	1.8	1.5	1.7	0.9
G3				1.7	0.9

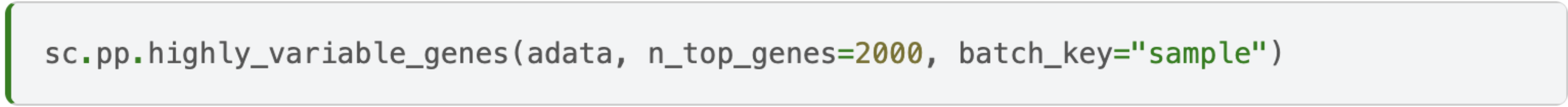
Lognormalized data, ready for downstream processing ☑

- Logarithmic transformation is the most common choice for this task
- Many alternatives exist, but performance differences are minor

3

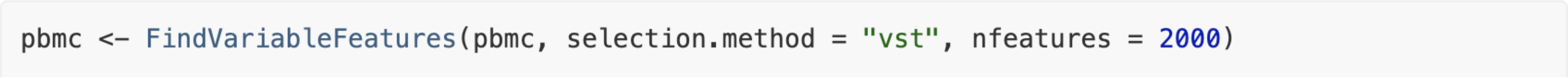
Finding highly variable genes

scanpy



```
sc.pp.highly_variable_genes(adata, n_top_genes=2000, batch_key="sample")
```

Seurat

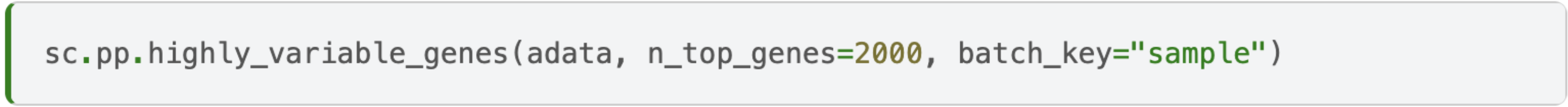


```
pbmc <- FindVariableFeatures(pbmc, selection.method = "vst", nfeatures = 2000)
```

3

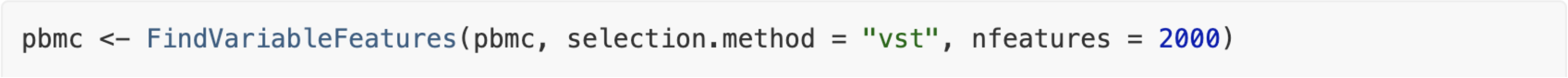
Finding highly variable genes

scanpy



```
sc.pp.highly_variable_genes(adata, n_top_genes=2000, batch_key="sample")
```

Seurat



```
pbmc <- FindVariableFeatures(pbmc, selection.method = "vst", nfeatures = 2000)
```

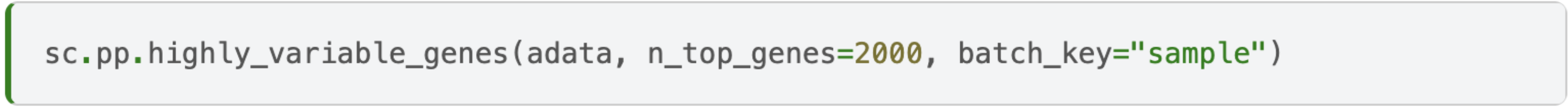
Why?

	A	B	C	D	E
G4	0	0	0	0	0
G5	1.8	1.9	0.2	0.2	2.1
G6	0.5	0.5	1.0	0.9	0.4
G7	0	0	0.1	0	0
G8	1.6	1.5	1.6	1.5	1.7

3

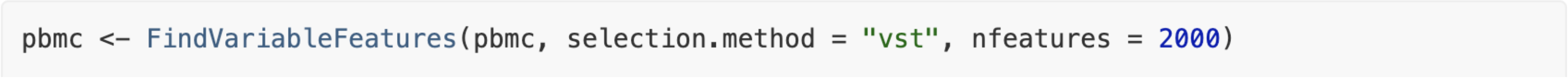
Finding highly variable genes

scanpy



```
sc.pp.highly_variable_genes(adata, n_top_genes=2000, batch_key="sample")
```

Seurat



```
pbmc <- FindVariableFeatures(pbmc, selection.method = "vst", nfeatures = 2000)
```

Why?

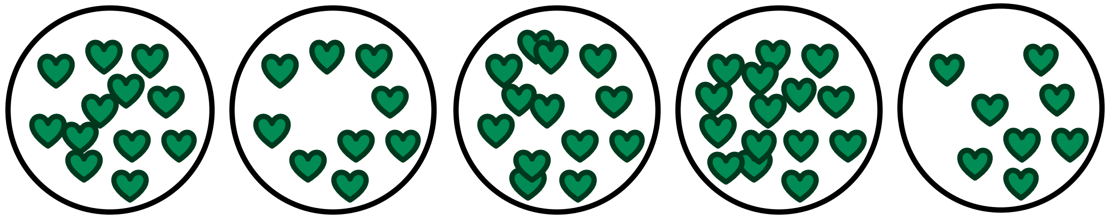
	A	B	C	D	E
G4	0	0	0	0	0
G5	1.8	1.9	0.2	0.2	2.1
G6	0.5	0.5	1.0	0.9	0.4
G7	0	0	0.1	0	0
G8	1.6	1.5	1.6	1.5	1.7

- Genes that are hardly expressed at all and/or do not vary a lot across cells are less valuable for analysis
- Masking them increases computational efficiency while simultaneously reducing analysis noise

3

Finding highly variable genes

How?



Highly expressed gene

Mean = 11.4 count

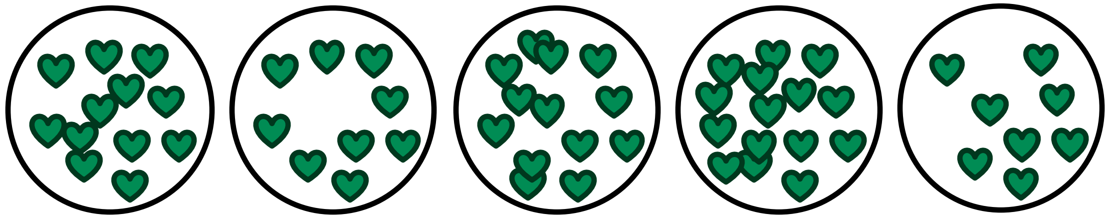
Variance = 6.64 counts

Naïve idea: Let’s just take the genes with the highest variance across cells.

3

Finding highly variable genes

How?

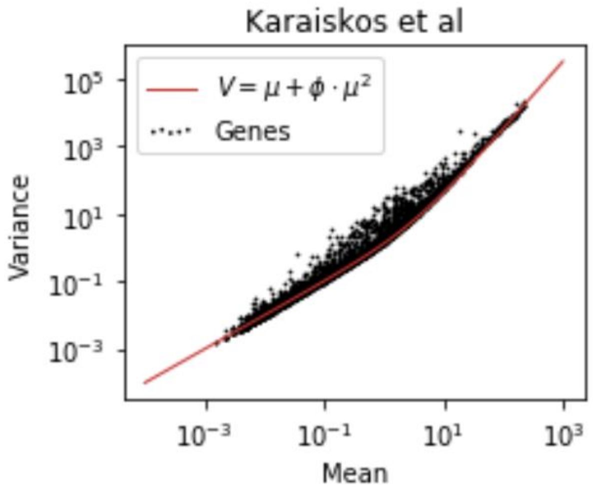


Highly expressed gene

Mean = 11.4 count

Variance = 6.64 counts

~~Naïve idea: Let’s just take the genes with the highest variance across cells.~~



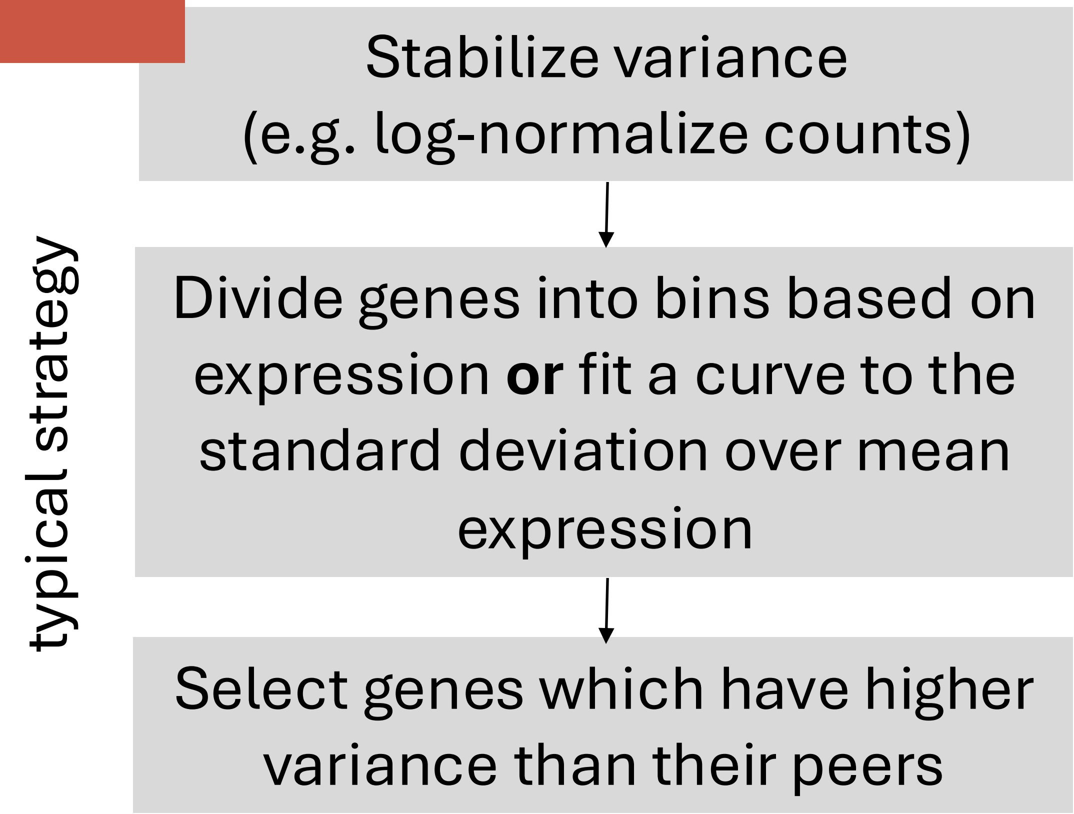
Heteroscedastic!

- Genes with higher expression also have higher variance by default
- To find the interesting genes, we need to compare their variability with that of similarly expressed genes

3

Finding highly variable genes

How?

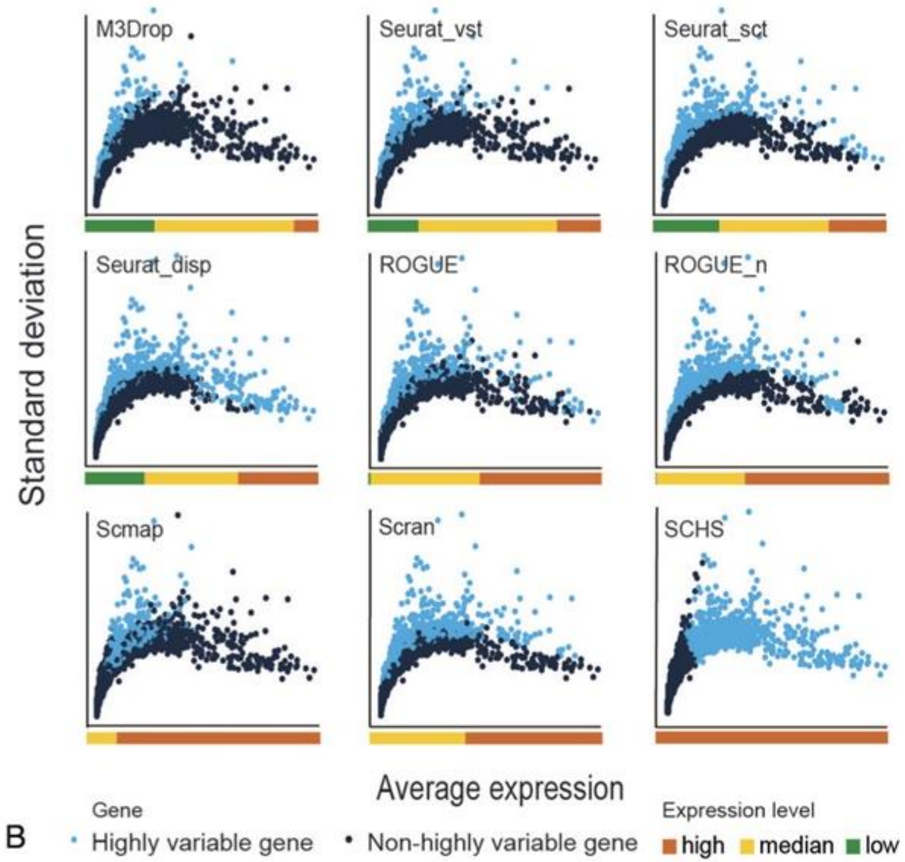


typical strategy

Stabilize variance
(e.g. log-normalize counts)

Divide genes into bins based on expression **or** fit a curve to the standard deviation over mean expression

Select genes which have higher variance than their peers

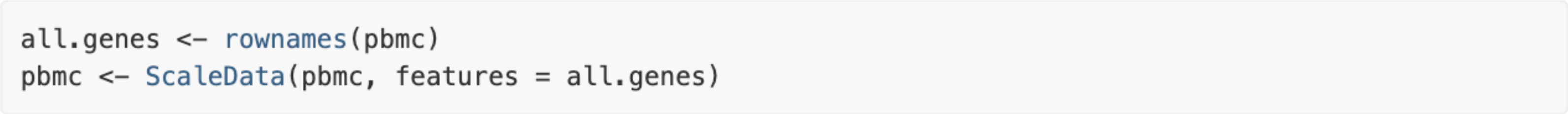


Zhang, Yinan; Xie, Xiaowei; Wu, Peng∗; Zhu, Ping∗. SIEVE: identifying robust single cell variable genes for single-cell RNA sequencing data. Blood Science 3(2):p 35-39, April 2021. | DOI: 10.1097/BS9.0000000000000072

4?

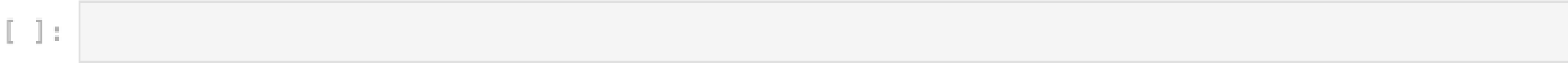
Scaling / z-scoring / standardization brings all gene expression values onto a common scale

All Seurat tutorials:



```
all.genes <- rownames(pbmc)
pbmc <- ScaleData(pbmc, features = all.genes)
```

Most scanpy tutorials:

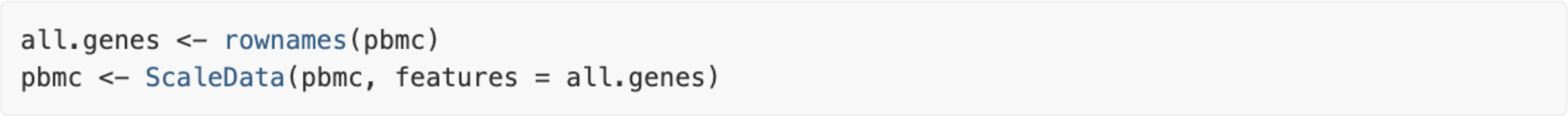


```
[ ]:
```

4?

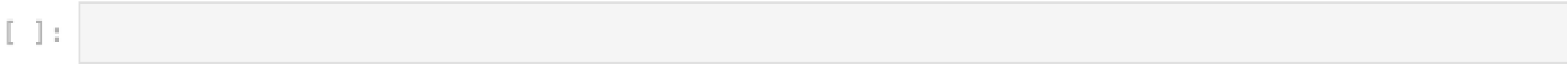
Scaling / z-scoring / standardization brings all gene expression values onto a common scale

All Seurat tutorials:



```
all.genes <- rownames(pbmc)
pbmc <- ScaleData(pbmc, features = all.genes)
```

Most scanpy tutorials:



```
[ ]:
```

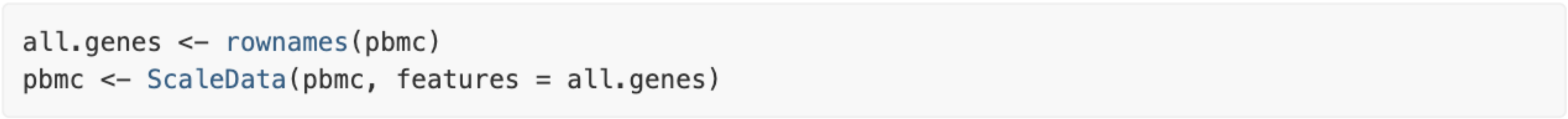
$$x = \frac{x - mean(\vec{x})}{std(x)}$$

- Subtract mean
- Divide by standard deviation
- → all values are centered around 0 and have unit variance

4?

Scaling / z-scoring / standardization brings all gene expression values onto a common scale

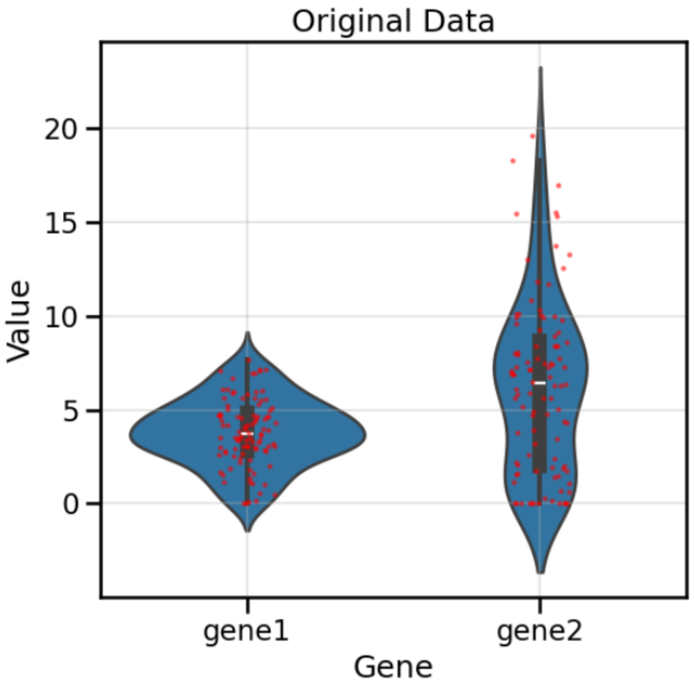
All Seurat tutorials:



```
all.genes <- rownames(pbmc)
pbmc <- ScaleData(pbmc, features = all.genes)
```

Most scanpy tutorials:

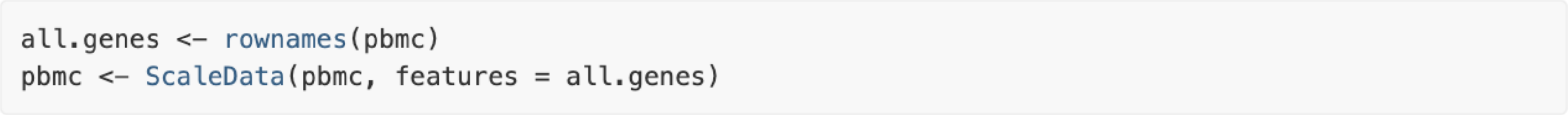
```
[ ]:
```



4?

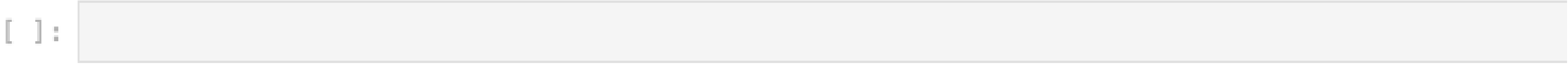
Scaling / z-scoring / standardization brings all gene expression values onto a common scale

All Seurat tutorials:

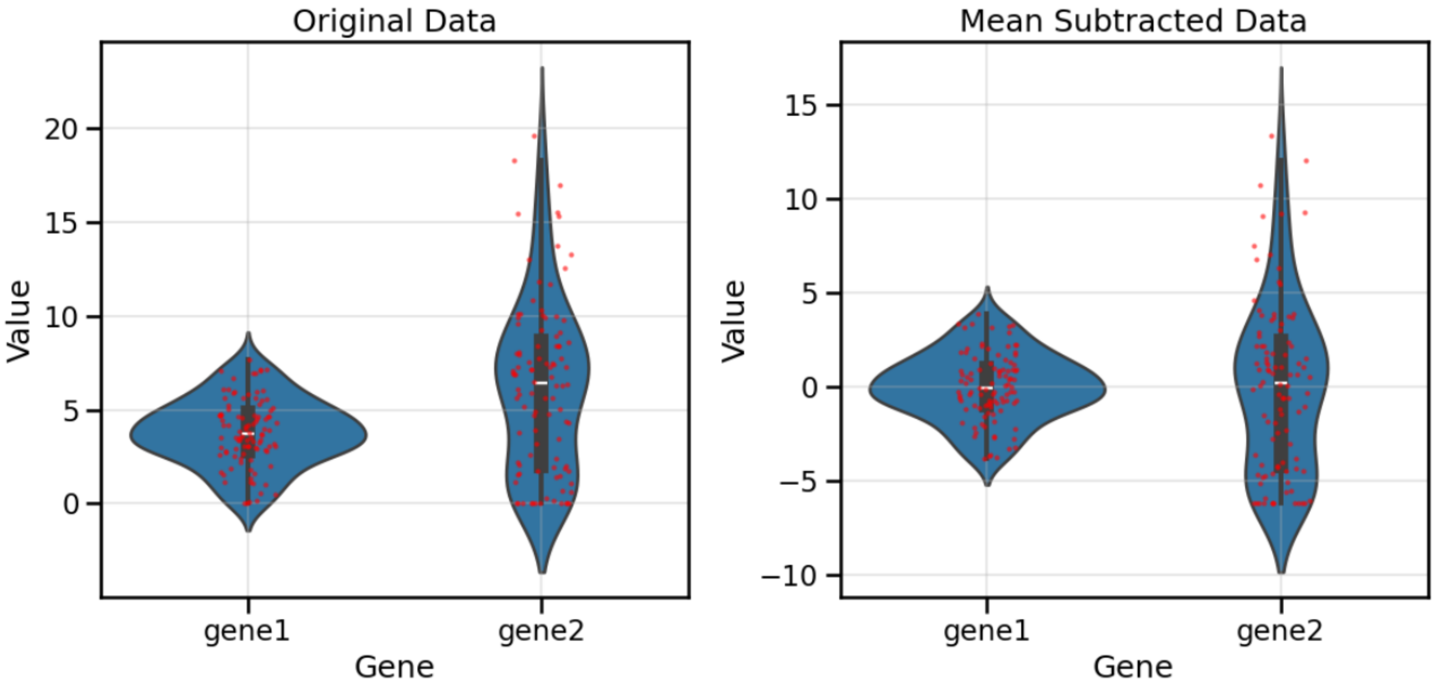


```
all.genes <- rownames(pbmc)
pbmc <- ScaleData(pbmc, features = all.genes)
```

Most scanpy tutorials:



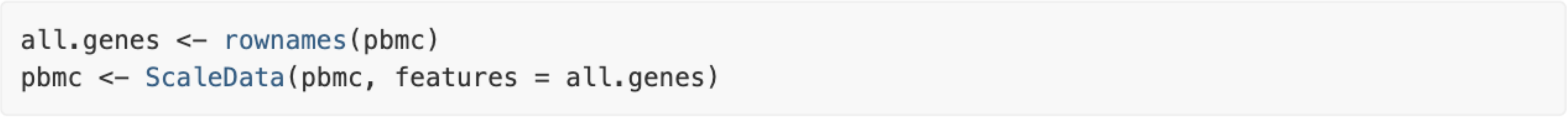
```
[ ]:
```



4?

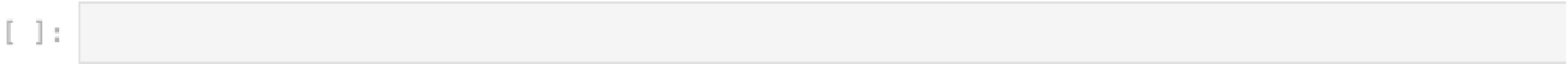
Scaling / z-scoring / standardization brings all gene expression values onto a common scale

All Seurat tutorials:

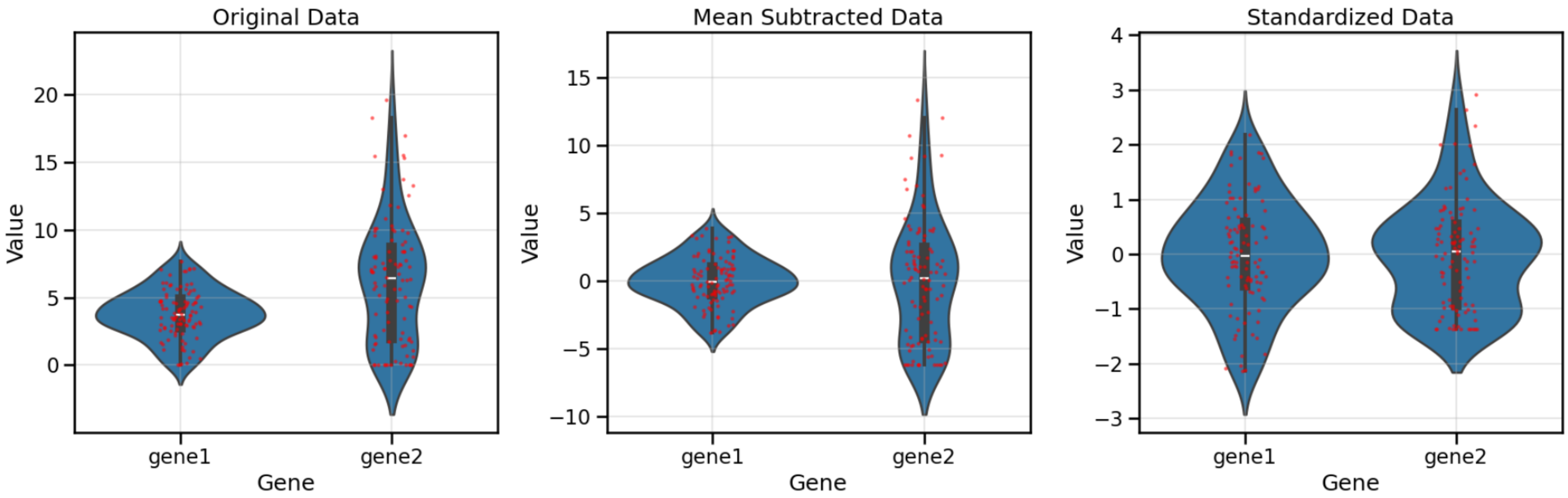


```
all.genes <- rownames(pbmc)
pbmc <- ScaleData(pbmc, features = all.genes)
```

Most scanpy tutorials:

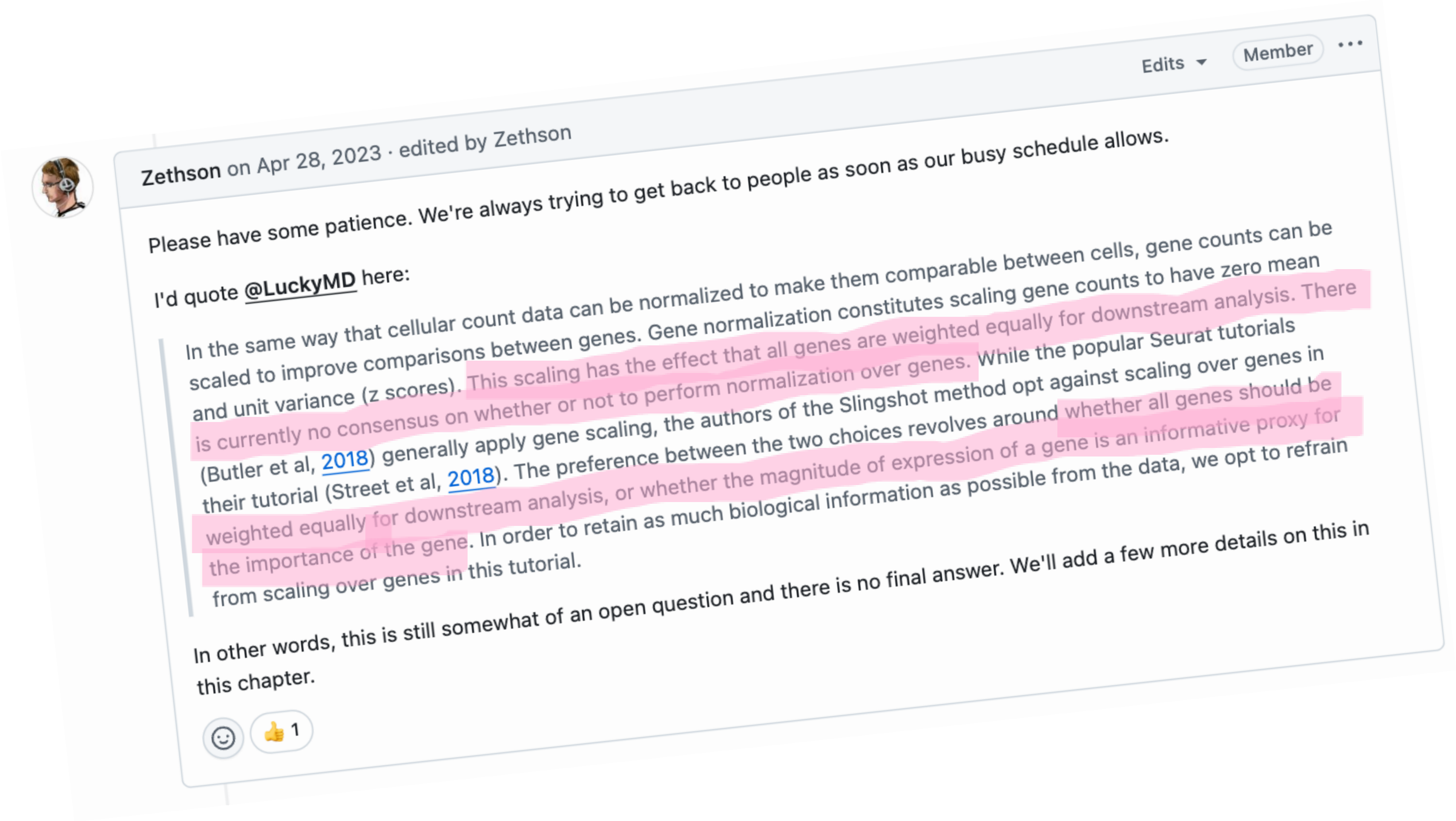


```
[ ]:
```



4?

To scale or not to scale?



Zethson on Apr 28, 2023 · edited by Zethson

Please have some patience. We're always trying to get back to people as soon as our busy schedule allows.

I'd quote **@LuckyMD** here:

> In the same way that cellular count data can be normalized to make them comparable between cells, gene counts can be scaled to improve comparisons between genes. Gene normalization constitutes scaling gene counts to have zero mean and unit variance (z scores). This scaling has the effect that all genes are weighted equally for downstream analysis. There is currently no consensus on whether or not to perform normalization over genes. While the popular Seurat tutorials (Butler et al, 2018) generally apply gene scaling, the authors of the Slingshot method opt against scaling over genes in their tutorial (Street et al, 2018). The preference between the two choices revolves around whether all genes should be weighted equally for downstream analysis, or whether the magnitude of expression of a gene is an informative proxy for the importance of the gene. In order to retain as much biological information as possible from the data, we opt to refrain from scaling over genes in this tutorial.

In other words, this is still somewhat of an open question and there is no final answer. We'll add a few more details on this in this chapter.

☺ 👍 1

https://github.com/theislab/single-cell-best-practices/issues/185

4?

To scale or not to scale?

No!

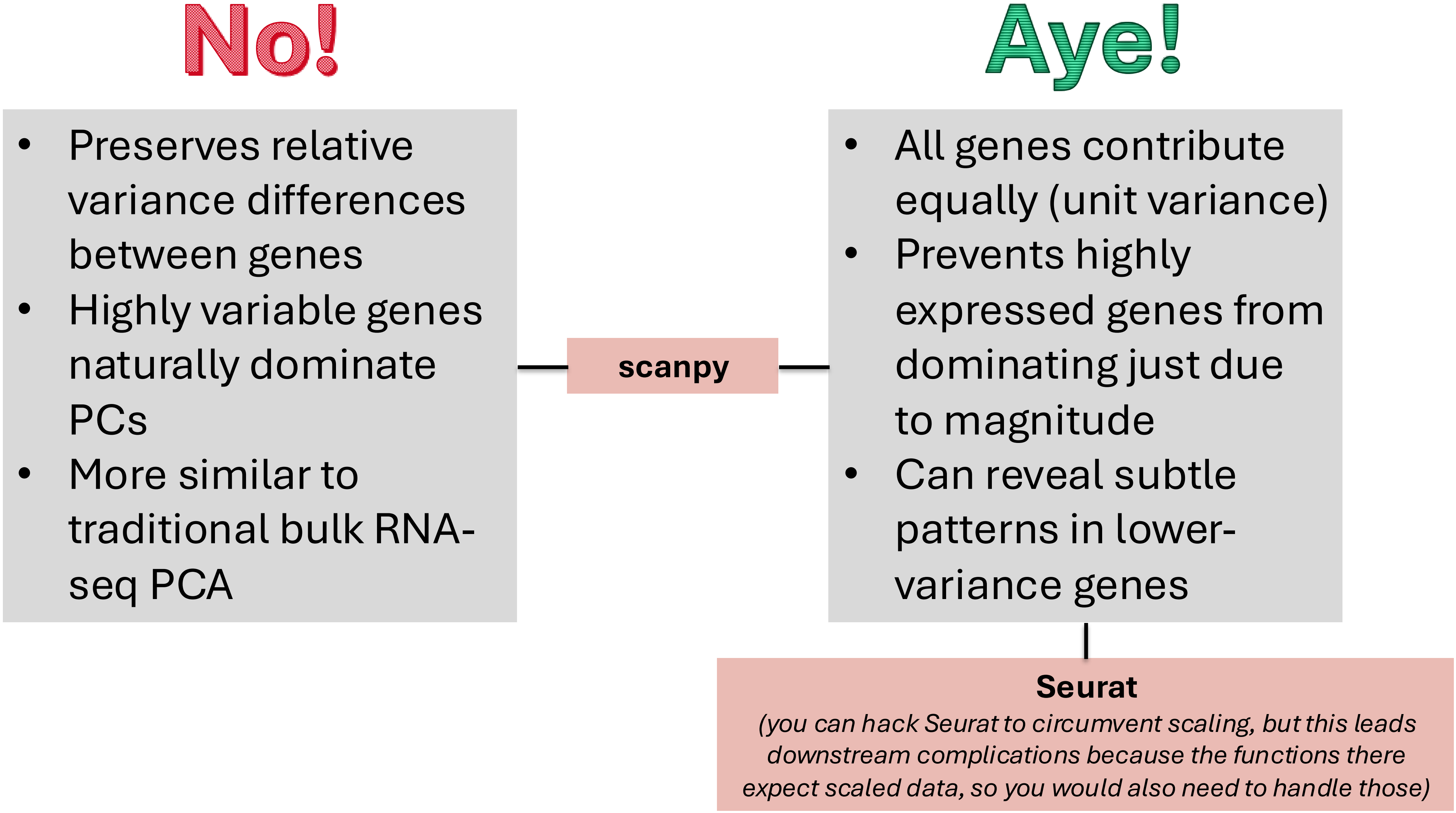
- Preserves relative variance differences between genes
- Highly variable genes naturally dominate PCs
- More similar to traditional bulk RNA-seq PCA

Aye!

- All genes contribute equally (unit variance)
- Prevents highly expressed genes from dominating just due to magnitude
- Can reveal subtle patterns in lower-variance genes

4?

To scale or not to scale?



No!

- Preserves relative variance differences between genes
- Highly variable genes naturally dominate PCs
- More similar to traditional bulk RNA-seq PCA

scanpy

Aye!

- All genes contribute equally (unit variance)
- Prevents highly expressed genes from dominating just due to magnitude
- Can reveal subtle patterns in lower-variance genes

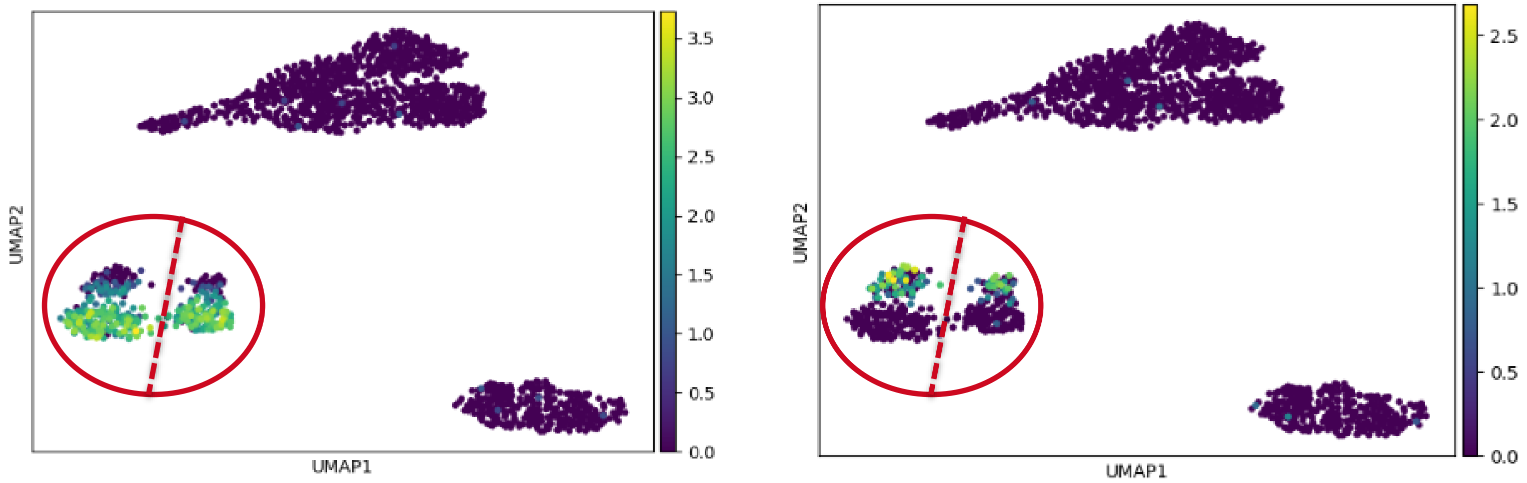
Seurat

(you can hack Seurat to circumvent scaling, but this leads downstream complications because the functions there expect scaled data, so you would also need to handle those)

4?

Example of effects upcoming in workshop data

Split appears only **without** scaling!



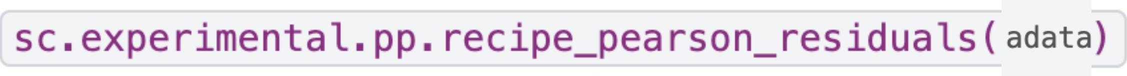
- Non-scaling approach reveals a “duplicate” population
- The difference between the twins is just one highly expressed gene
- Through scaling, the importance of this gene is lessened and

https://github.com/theislab/single-cell-best-practices/issues/185

1 2 3 4

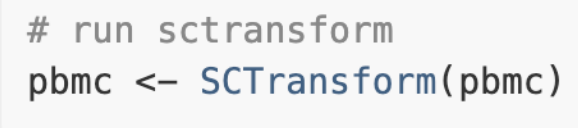
Alternative: Pearson Residuals (SCTransform)

scanpy



```
sc.experimental.pp.recipe_pearson_residuals(adata)
```

Seurat



```
# run sctransform
pbmc <- SCTransform(pbmc)
```

Replace normalization, log-transformation and highly variable gene search

Why?

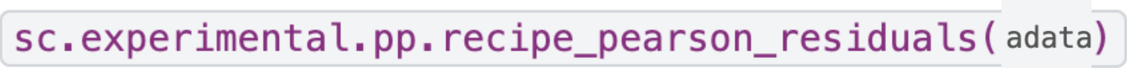
Total gene expression variability = Technical variability + Biological variability

https://scanpy-tutorials.readthedocs.io/en/latest/tutorial_pearson_residuals.html
https://satijalab.org/seurat/articles/sctransform_vignette

1 2 3 4

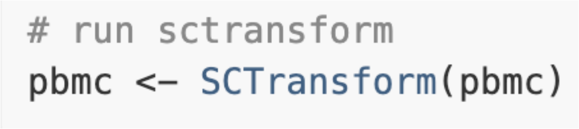
Alternative: Pearson Residuals (SCTransform)

scanpy



```
sc.experimental.pp.recipe_pearson_residuals(adata)
```

Seurat



```
# run sctransform
pbmc <- SCTransform(pbmc)
```

Replace normalization, log-transformation and highly variable gene search

Why?

Total gene expression variability = Technical variability + Biological variability

Biological variability = Total gene expression variability - Technical variability

measure

stats model

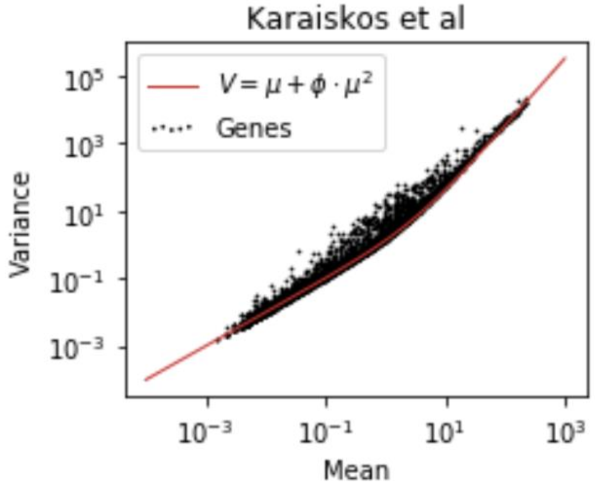
https://scanpy-tutorials.readthedocs.io/en/latest/tutorial_pearson_residuals.html
https://satijalab.org/seurat/articles/sctransform_vignette

1 2 3 4

Alternative: Pearson Residuals (SCTransform)

How?

Biological variability = Total gene expression variability measure - Technical variability stats model



Negative binomial distribution describes the technical noise of single cell data.

Raw count

Expected count under NB model

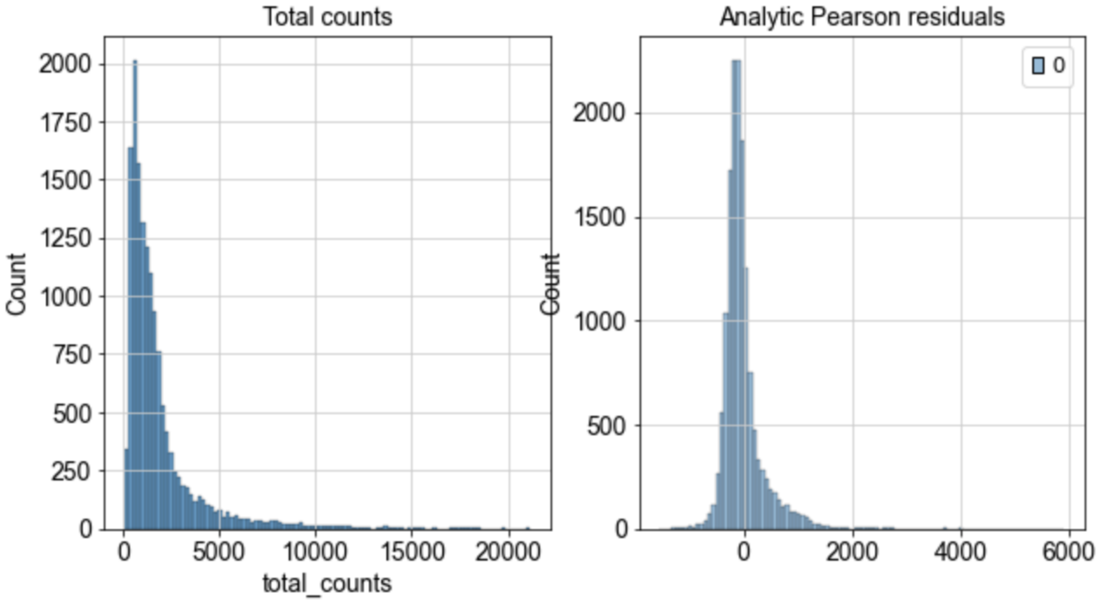
Pearson residual

$$Z_{cg} = \frac{X_{cg} - \hat{\mu}_{cg}}{\sqrt{\hat{\mu}_{cg} + \hat{\mu}_{cg}^2/\theta}}$$

Expected std under NB model

Lause, J., Berens, P. & Kobak, D. Analytic Pearson residuals for normalization of single-cell RNA-seq UMI data. Genome Biol 22, 258 (2021). https://doi.org/10.1186/s13059-021-02451-7

Alternative: Pearson Residuals (SCTransform)



→ after transformation into Pearson Residuals

Basic Interpretation

- **Pearson residual = 0**: The observed count matches exactly what the model expected
- **Positive residual (> 0)**: The observed count is higher than expected
- **Negative residual (< 0)**: The observed count is lower than expected

Magnitude Interpretation

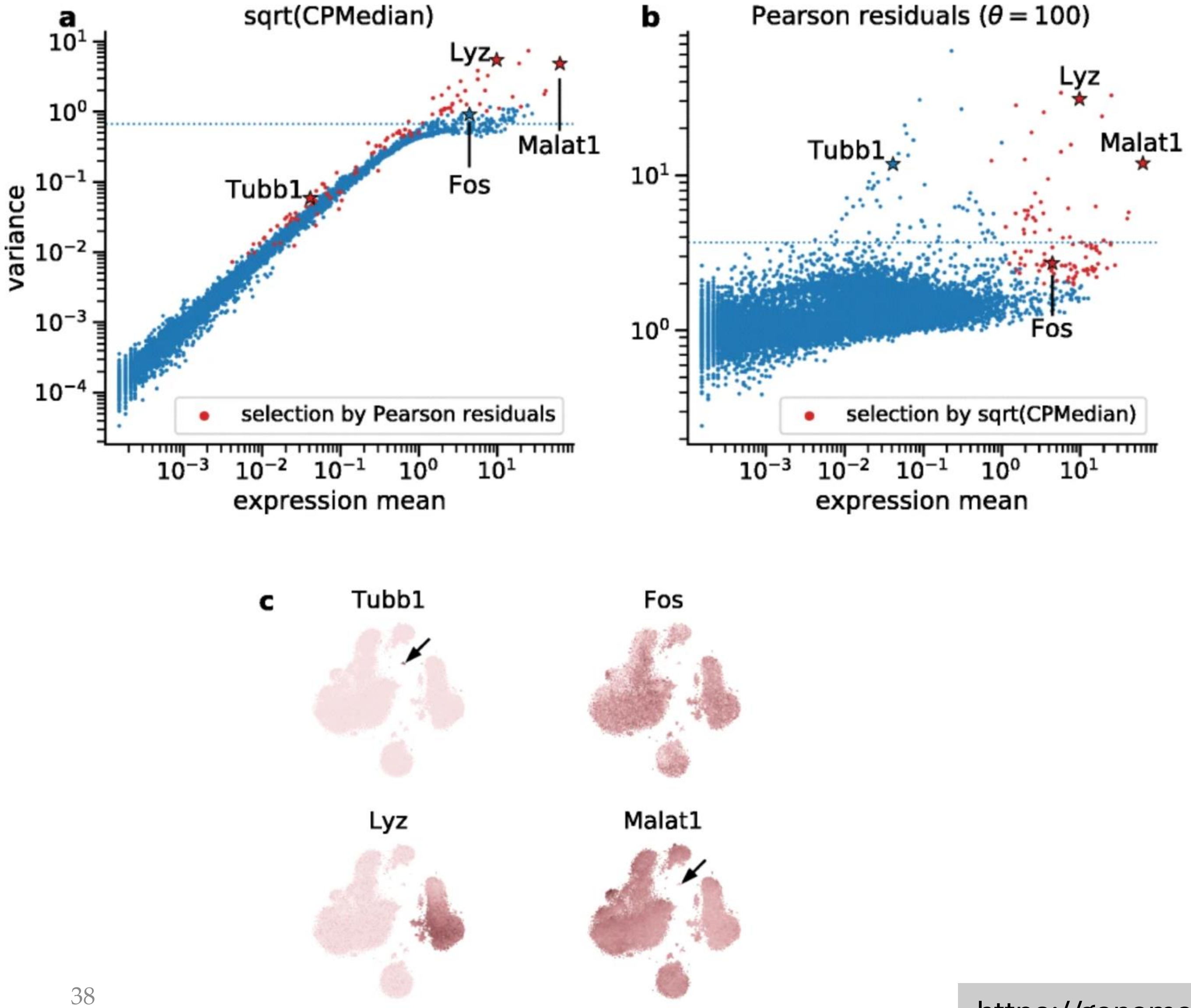
The magnitude of a Pearson residual indicates the strength of the deviation:

- **|residual| < 2**: Minor deviation, likely just random noise
- **|residual| > 3**: Strong deviation, highly likely to be biologically significant
- **|residual| > 5**: Extreme deviation, almost certainly represents a real biological signal

https://www.sc-best-practices.org/preprocessing_visualization/normalization.html

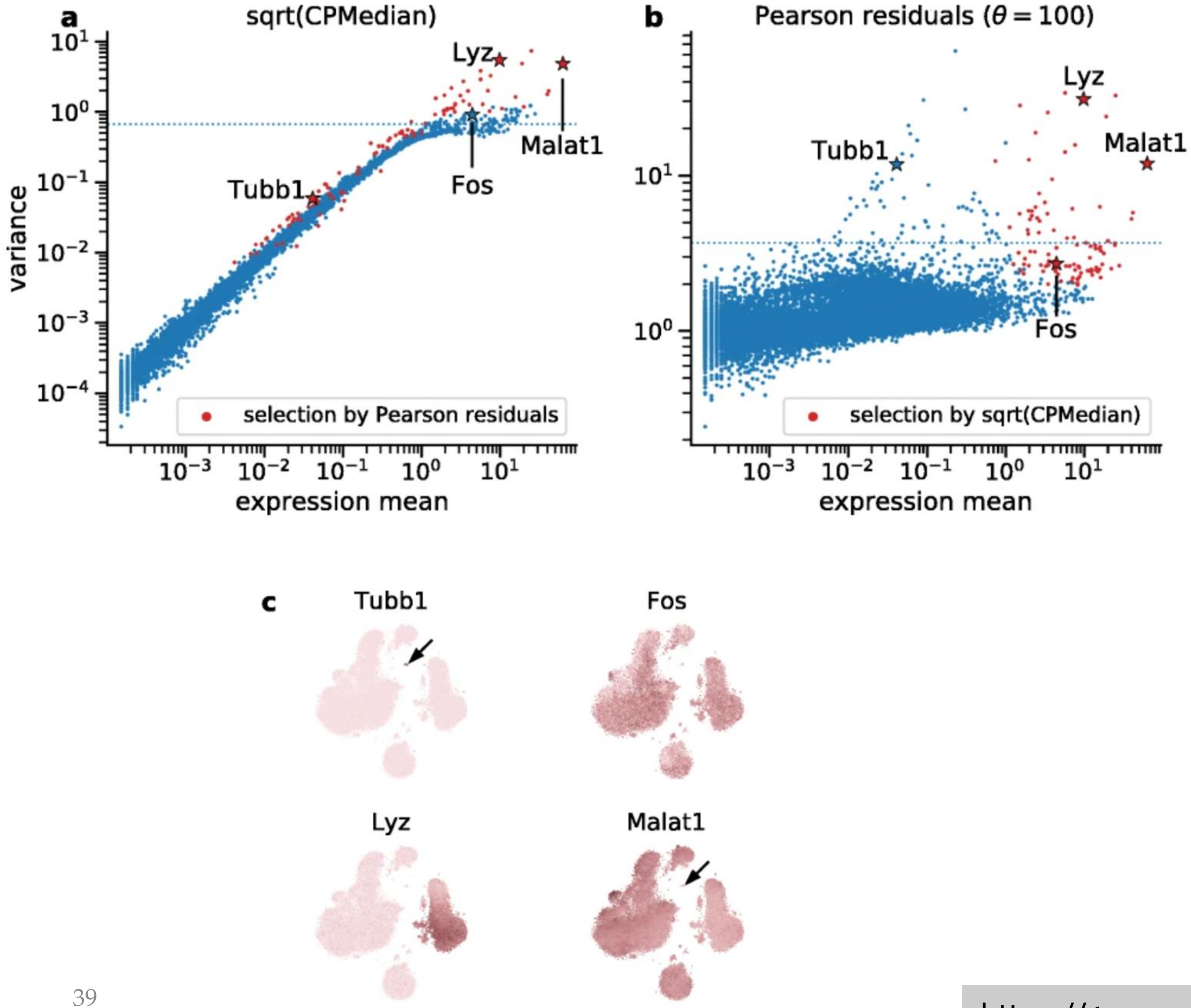
1 2 3 4

Alternative: Pearson Residuals (SCTransform)



https://genomebiology.biomedcentral.com/articles/10.1186/s13059-021-02451-7

Alternative: Pearson Residuals (SCTransform)

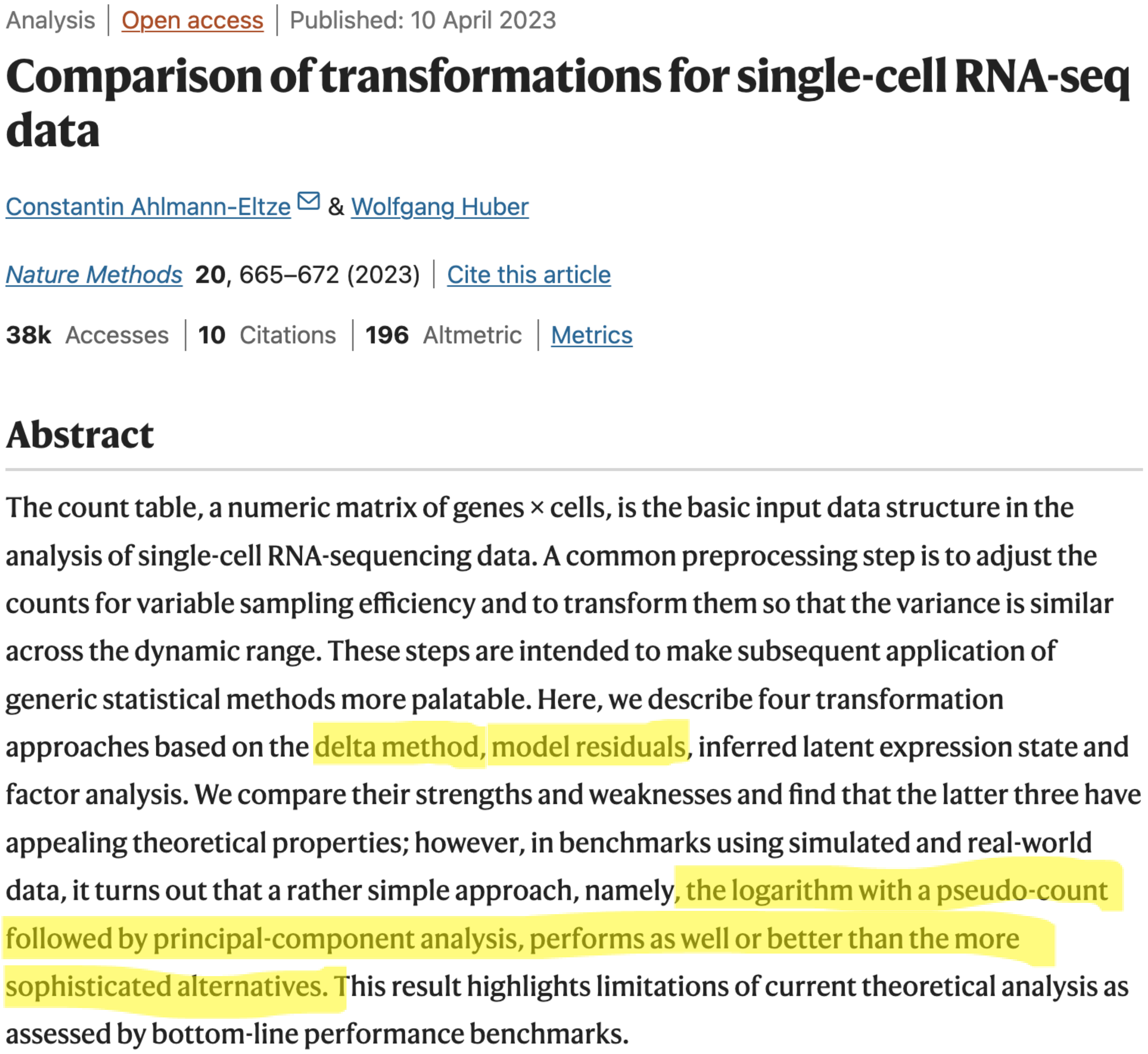


- Highly variable gene selection via Pearson Residuals can identify genes relevant for tiny cell populations
- Downstream analyses (e.g. clustering) can benefit from this
- Calculation is relatively expensive and can be prohibitive for large datasets

https://genomebiology.biomedcentral.com/articles/10.1186/s13059-021-02451-7

1 2 3 4

Pearson Residuals are not a must



Analysis | Open access | Published: 10 April 2023

Comparison of transformations for single-cell RNA-seq data

Constantin Ahlmann-Eltze & Wolfgang Huber

Nature Methods **20**, 665–672 (2023) | Cite this article

38k Accesses | **10** Citations | **196** Altmetric | Metrics

Abstract

The count table, a numeric matrix of genes × cells, is the basic input data structure in the analysis of single-cell RNA-sequencing data. A common preprocessing step is to adjust the counts for variable sampling efficiency and to transform them so that the variance is similar across the dynamic range. These steps are intended to make subsequent application of generic statistical methods more palatable. Here, we describe four transformation approaches based on the delta method, model residuals, inferred latent expression state and factor analysis. We compare their strengths and weaknesses and find that the latter three have appealing theoretical properties; however, in benchmarks using simulated and real-world data, it turns out that a rather simple approach, namely, the logarithm with a pseudo-count followed by principal-component analysis, performs as well or better than the more sophisticated alternatives. This result highlights limitations of current theoretical analysis as assessed by bottom-line performance benchmarks.

Ahlmann-Eltze, C., Huber, W. Comparison of transformations for single-cell RNA-seq data. Nat Methods 20, 665–672 (2023). https://doi.org/10.1038/s41592-023-01814-1

To scale or not to scale?

Standard Workflows:

Scanpy (typical):

python

```
sc.pp.normalize_total(adata) # normalize to counts per 10k sc.pp.log1p(adata) # log(1+x) transform sc.tl.pca(adata, zero_center=True) # PCA with centering
```

No scaling to unit variance!

Seurat (typical):

r

```
NormalizeData(obj) # similar log normalization ScaleData(obj) # centers AND scales to unit variance RunPCA(obj) # expects already-centered data
```

Your Situation:

You're feeding **log-normalized but unscaled** data to Seurat's PCA, which likely **doesn't center** because it assumes ScaleData() was already run. This would explain the difference!

To test your hypothesis:

1.Check if Seurat's RunPCA() has a centering parameter (I believe it doesn't by default)
2.You could manually center your data before feeding it to Seurat:
3.r
4.*# After NormalizeData, manually center without scaling* data.centered <- sweep(data, 1, rowMeans(data), "-") *# Then run PCA on this*
5.Or check what scanpy's PCA looks like with zero_center=False - it should look more like your Seurat result

Why the Different Philosophies?

Scanpy approach (log-norm + center, no scaling):

- Preserves relative variance differences between genes
- Highly variable genes naturally dominate PCs
- More similar to traditional bulk RNA-seq PCA

Seurat approach (log-norm + scale):

- All genes contribute equally (unit variance)
- Prevents highly expressed genes from dominating just due to magnitude
- Can reveal subtle patterns in lower-variance genes

The difference in plots you're seeing is likely because:

- Scanpy: PCA on centered but unscaled log-counts
- Seurat (your hack): PCA on uncentered, unscaled log-counts

The centering makes a huge difference, as we discussed earlier - without it, PC1 often just captures mean expression level rather than biological variation.

Ahlmann-Eltze, C., Huber, W. Comparison of transformations for single-cell RNA-seq data. Nat Methods 20, 665–672 (2023). https://doi.org/10.1038/s41592-023-01814-1