Single Cell Transcriptomics

From Experimental Design to Data Analysis

Day 2

University of Maryland
Baltimore County

Workshop Outline

Part I: Single Cell Analysis – Theory

Part II: Introduction to R

Part III: Data Analysis and Visualization Workshop

Goals of this Workshop

Today

- Understand single cell data normalization and dimensionality reduction techniques
- Learn about single cell clustering and visualization techniques
- Learn how to generate and interpret tSNE and UMAP plots
- Get familiar working with R Studio
- Learn where to find and download published datasets

Overall

- Learn how to design and perform a “full stack” single cell RNA-seq experiment

Workshop Outline – Part I

Part I: Single Cell Analysis – Theory

- Data analysis overview
- Count matrices
- Data scaling and normalization
- Data quality control and filtering
- Principal component analysis (PCA) and clustering algorithms
- Understanding TSNE and UMAP plots
- Marker gene identification
- Extensions
    - Dataset integration & batch correction
    - Pseudotime analysis
    - Differential gene expression testing
    - Gene Ontology (GO) analysis

Workshop Outline – Part II/III

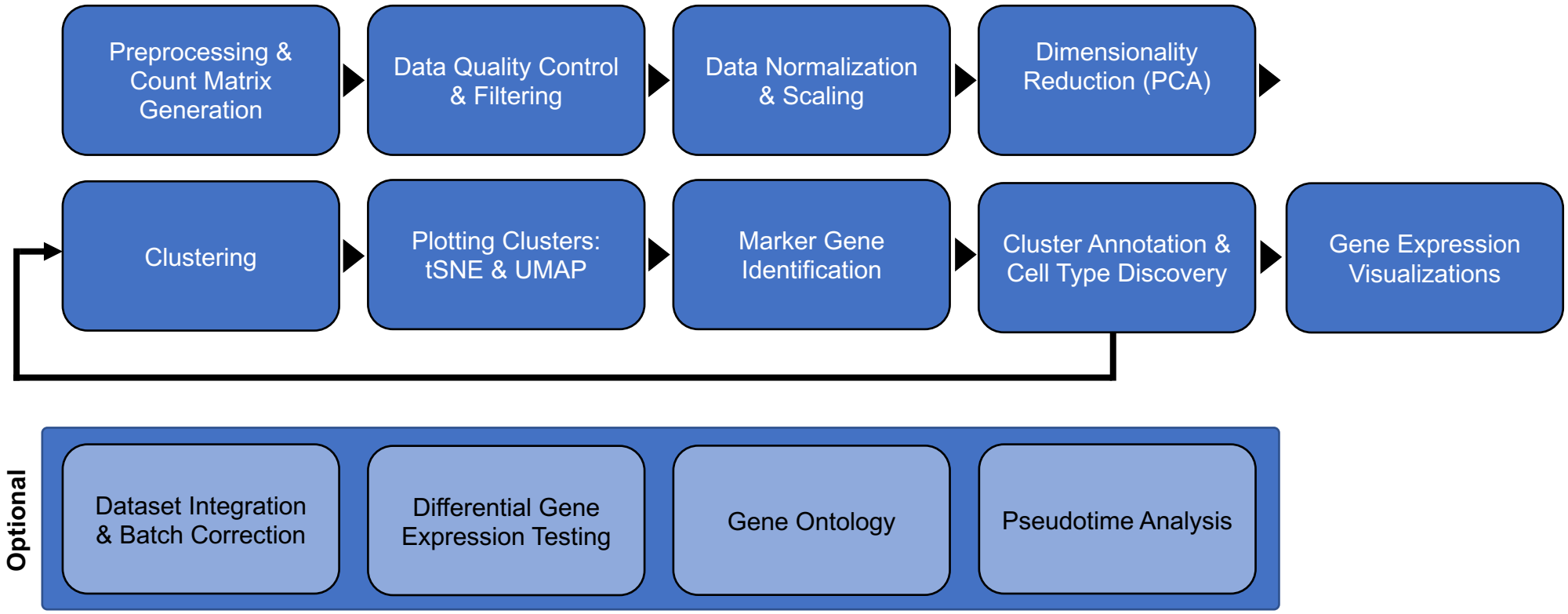
Part II: Introduction to R Statistical Computing

- Opening R Studio
- Installing R packages
- Syntax and data structures
- Functions and arguments

Part III: Data Analysis and Visualization Workshop

Part I: Single Cell Analysis - Theory

Single Cell Experiment Outline



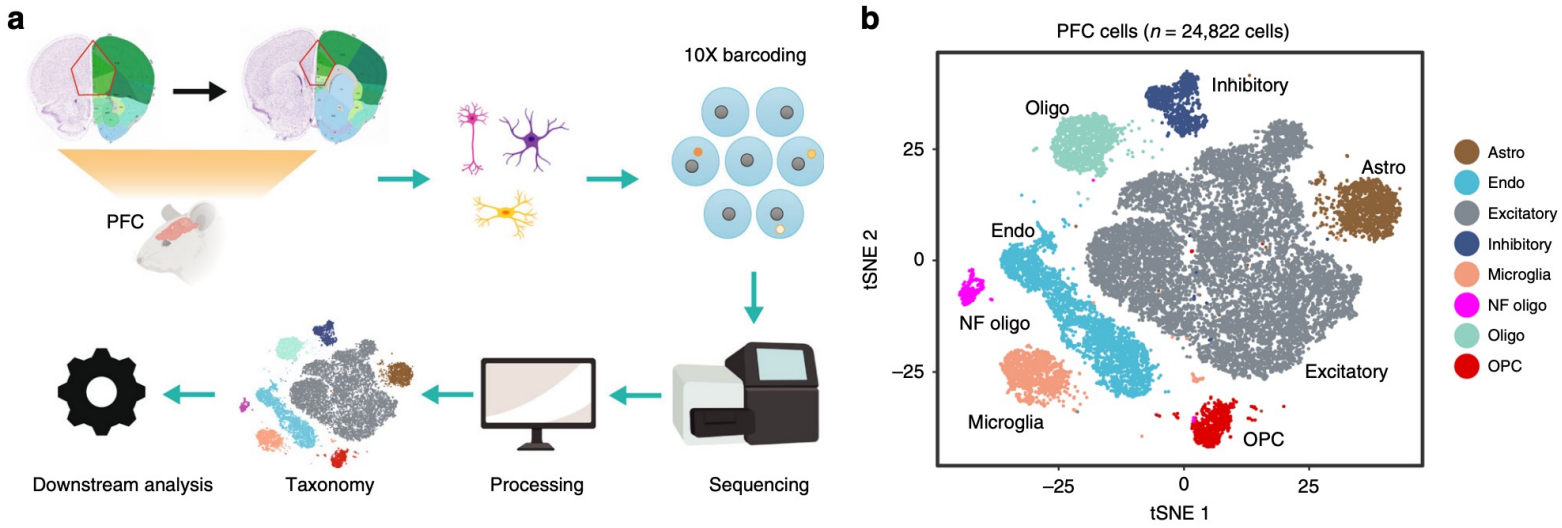
Published dataset we will explore today

ARTICLE

https://doi.org/10.1038/s41467-019-12054-3 OPEN

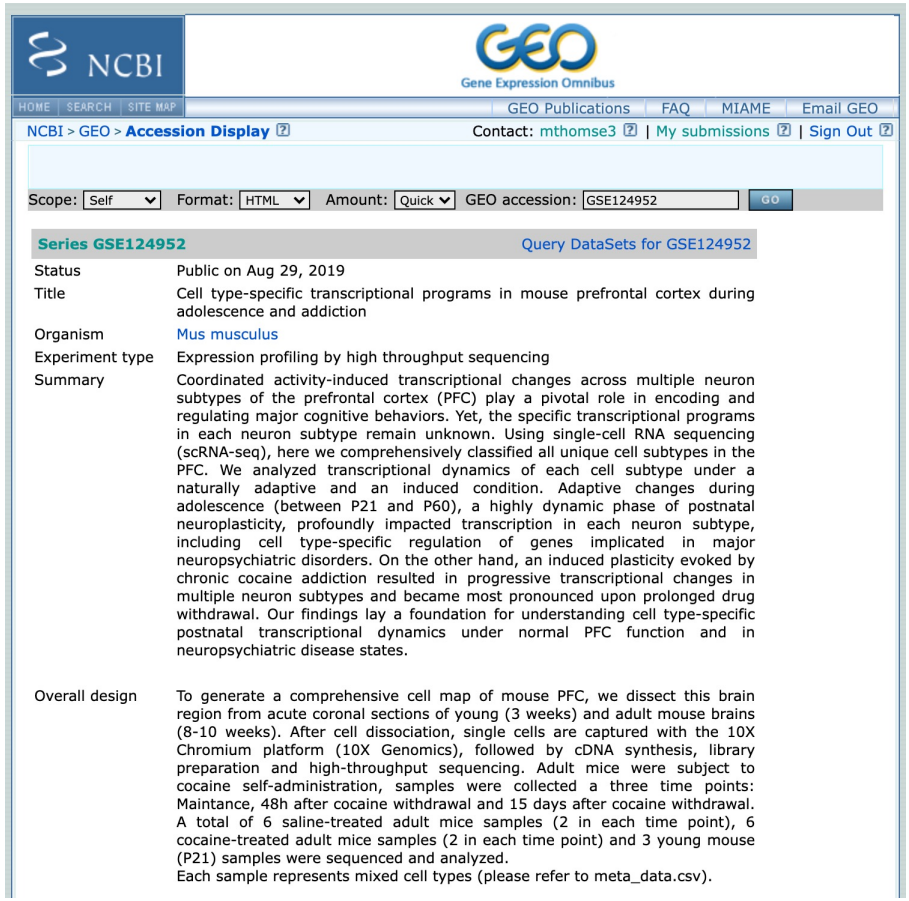
Cell type-specific transcriptional programs in mouse prefrontal cortex during adolescence and addiction

Aritra Bhattacherjee[1,2,3,5], Mohamed Nadhir Djekidel[1,2,3,5], Renchao Chen[1,2,3,5], Wenqiang Chen[1,2,3,5], Luis M. Tuesta[1,2,3] & Yi Zhang[1,2,3,4]



Downloading data from the Gene Expression Omnibus (GEO)

The complete set of raw data can be found at: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE124952



NCBI

Gene Expression Omnibus

HOME | SEARCH | SITE MAP | GEO Publications | FAQ | MIAME | Email GEO

NCBI > GEO > **Accession Display** Contact: mthomse3 | My submissions | Sign Out

Scope: Self Format: HTML Amount: Quick GEO accession: GSE124952 GO

Series GSE124952 Query DataSets for GSE124952

Status	Public on Aug 29, 2019
Title	Cell type-specific transcriptional programs in mouse prefrontal cortex during adolescence and addiction
Organism	Mus musculus
Experiment type	Expression profiling by high throughput sequencing
Summary	Coordinated activity-induced transcriptional changes across multiple neuron subtypes of the prefrontal cortex (PFC) play a pivotal role in encoding and regulating major cognitive behaviors. Yet, the specific transcriptional programs in each neuron subtype remain unknown. Using single-cell RNA sequencing (scRNA-seq), here we comprehensively classified all unique cell subtypes in the PFC. We analyzed transcriptional dynamics of each cell subtype under a naturally adaptive and an induced condition. Adaptive changes during adolescence (between P21 and P60), a highly dynamic phase of postnatal neuroplasticity, profoundly impacted transcription in each neuron subtype, including cell type-specific regulation of genes implicated in major neuropsychiatric disorders. On the other hand, an induced plasticity evoked by chronic cocaine addiction resulted in progressive transcriptional changes in multiple neuron subtypes and became most pronounced upon prolonged drug withdrawal. Our findings lay a foundation for understanding cell type-specific postnatal transcriptional dynamics under normal PFC function and in neuropsychiatric disease states.
Overall design	To generate a comprehensive cell map of mouse PFC, we dissect this brain region from acute coronal sections of young (3 weeks) and adult mouse brains (8-10 weeks). After cell dissociation, single cells are captured with the 10X Chromium platform (10X Genomics), followed by cDNA synthesis, library preparation and high-throughput sequencing. Adult mice were subject to cocaine self-administration, samples were collected a three time points: Maintance, 48h after cocaine withdrawal and 15 days after cocaine withdrawal. A total of 6 saline-treated adult mice samples (2 in each time point), 6 cocaine-treated adult mice samples (2 in each time point) and 3 young mouse (P21) samples were sequenced and analyzed. Each sample represents mixed cell types (please refer to meta_data.csv).

Samples (15)
Less...

GSM3559978 PFC Adult scRNA-Seq Sample1
GSM3559979 PFC Adult scRNA-Seq Sample2 ←
GSM3559980 PFC Adult scRNA-Seq Sample3 ←
GSM3559981 PFC Adult scRNA-Seq Sample4
GSM3559982 PFC Adult scRNA-Seq Sample5 ←
GSM3559983 PFC Adult scRNA-Seq Sample6
GSM3559984 PFC Adult scRNA-Seq Sample7 ←
GSM3559985 PFC Adult scRNA-Seq Sample8
GSM3559986 PFC Adult scRNA-Seq Sample9
GSM3559987 PFC Adult scRNA-Seq Sample10
GSM3559988 PFC Adult scRNA-Seq Sample11
GSM3559989 PFC Adult scRNA-Seq Sample12
GSM3559990 PFC P21 scRNA-Seq Sample1
GSM3559991 PFC P21 scRNA-Seq Sample2
GSM3559992 PFC P21 scRNA-Seq Sample3

4 “control” samples

Relations

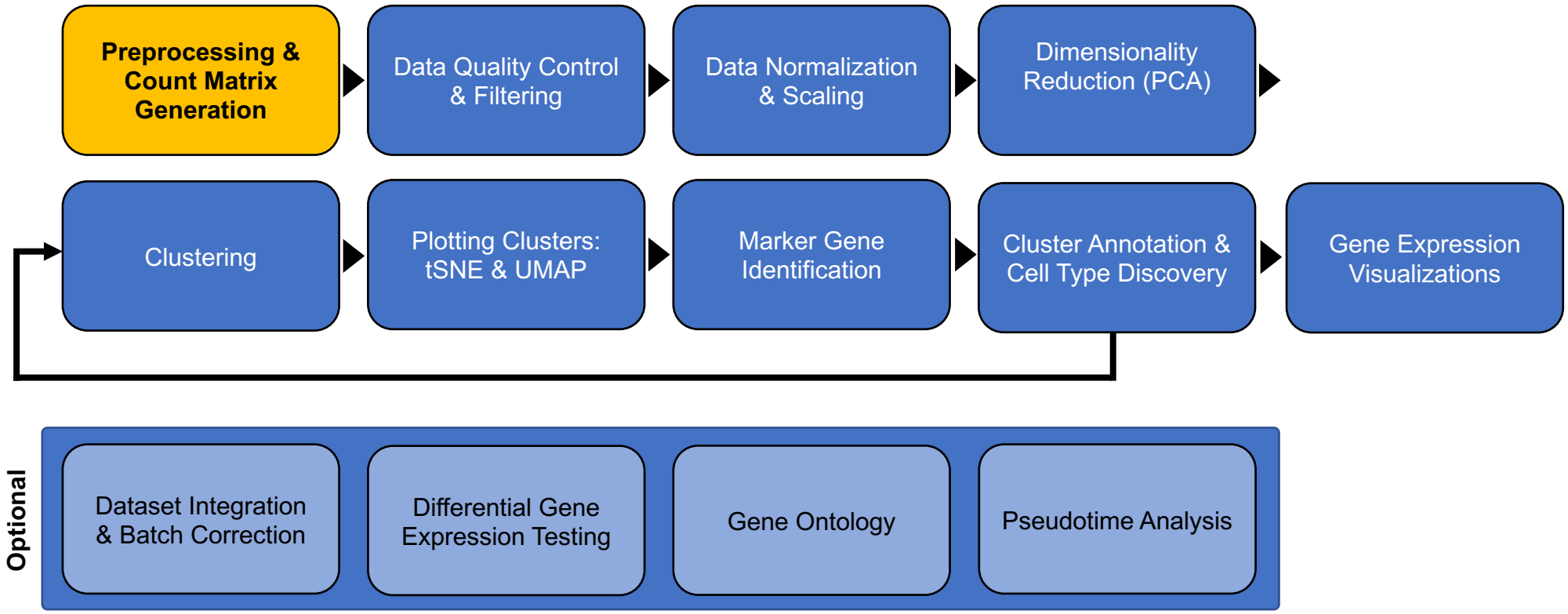
BioProject	PRJNA514448
SRA	SRP178464

Download family	Format
SOFT formatted family file(s)	SOFT
MINiML formatted family file(s)	MINiML
Series Matrix File(s)	TXT

Supplementary file	Size	Download	File type/resource
GSE124952_RAW.tar	363.7 Mb	(http)(custom)	TAR (of MTX, TSV)
GSE124952_expression_matrix.csv.gz	88.0 Mb	(ftp)(http)	CSV
GSE124952_meta_data.csv.gz	861.1 Kb	(ftp)(http)	CSV

← Count matrix download

Single Cell Experiment Outline

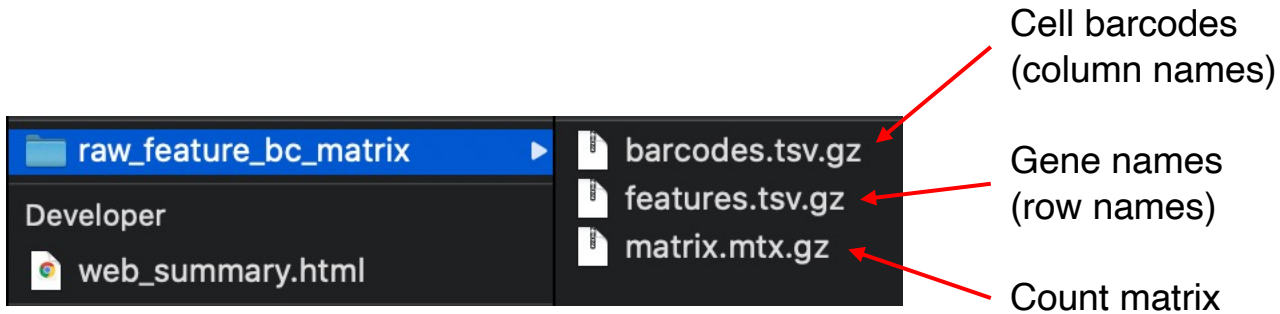


“Digital Gene Expression” A.K.A. Count Matrices

Standard Count Matrix

Cell:	1	2	···	N
GENE 1	1	2		14
GENE 2	4	27		8
GENE 3	0	0		1
⋮	⋮	⋮		⋮
GENE M	6	2		0

10X Genomics Count Matrix

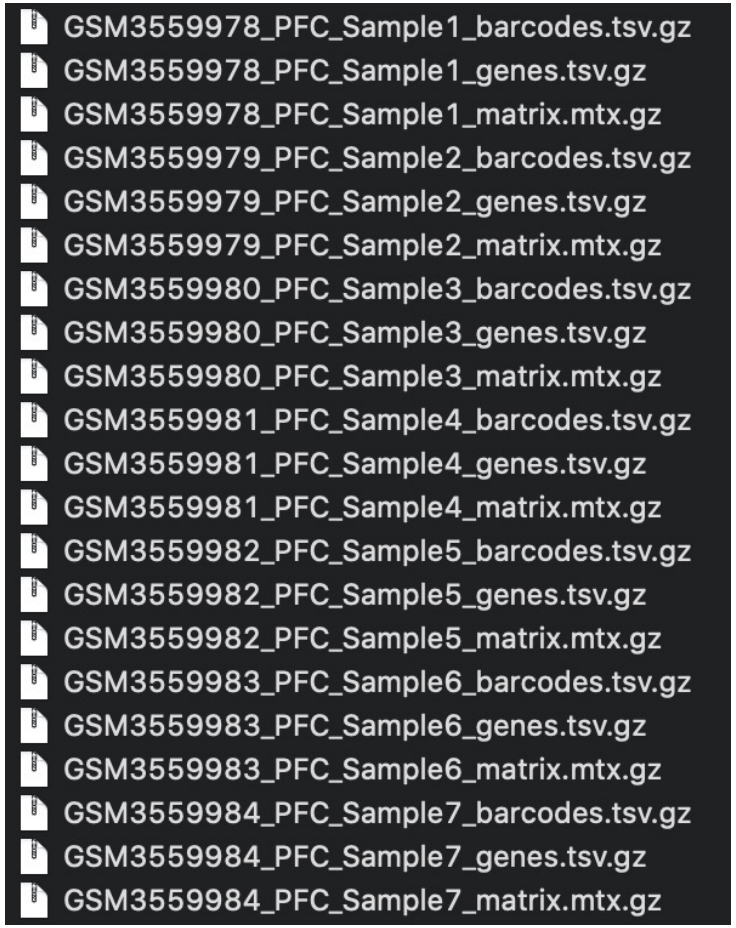


After running CellRanger, the raw count matrix will be exported to a folder named “raw_feature_bc_matrix” in the “outs” folder defined when you ran CellRanger

Note: in older versions of CellRanger, ”features.tsv.gz” may be named “genes.tsv.gz”

[Source: https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/output/matrices]

Quick note about data organization

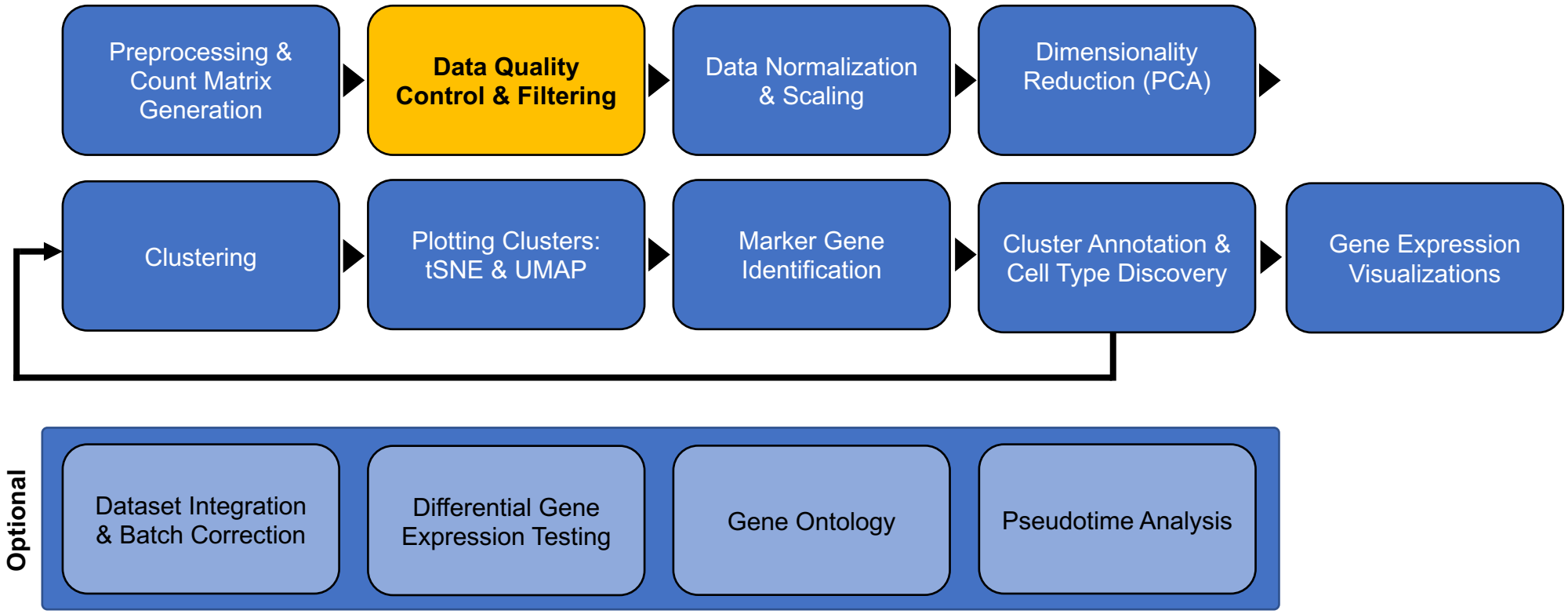


Downloading raw/processed data from GEO can come in a wide variety of formats determined by the authors when they upload. For this data, we need to simply reorganize/rename the files from the left to the structure on the right. The file structure on the right side will be easily recognized by Seurat when we import the data.

```
raw-data/
    |___pfc-sample2/
        |___barcodes.tsv.gz
        |___features.tsv.gz
        |___matrix.mtx.gz
    |___pfc-sample3/
        |___barcodes.tsv.gz
        |___features.tsv.gz
        |___matrix.mtx.gz
    |___pfc-sample5/
        |___barcodes.tsv.gz
        |___features.tsv.gz
        |___matrix.mtx.gz
    |___pfc-sample6/
        |___barcodes.tsv.gz
        |___features.tsv.gz
        |___matrix.mtx.gz
```

(note “genes” is changed to “features”)

Single Cell Experiment Outline



Quality Control Metrics

Goal: filter out low-quality cells and retain high-quality cells

Genes per cell ('nGene' or 'nFeature_RNA')
UMIs per cell ('nUMI' or 'nCount_RNA')

Low

- Poor mRNA capture
- Low mRNA content
- Degraded mRNA in sample
- Ambient RNA (not a 'real' cell)

High

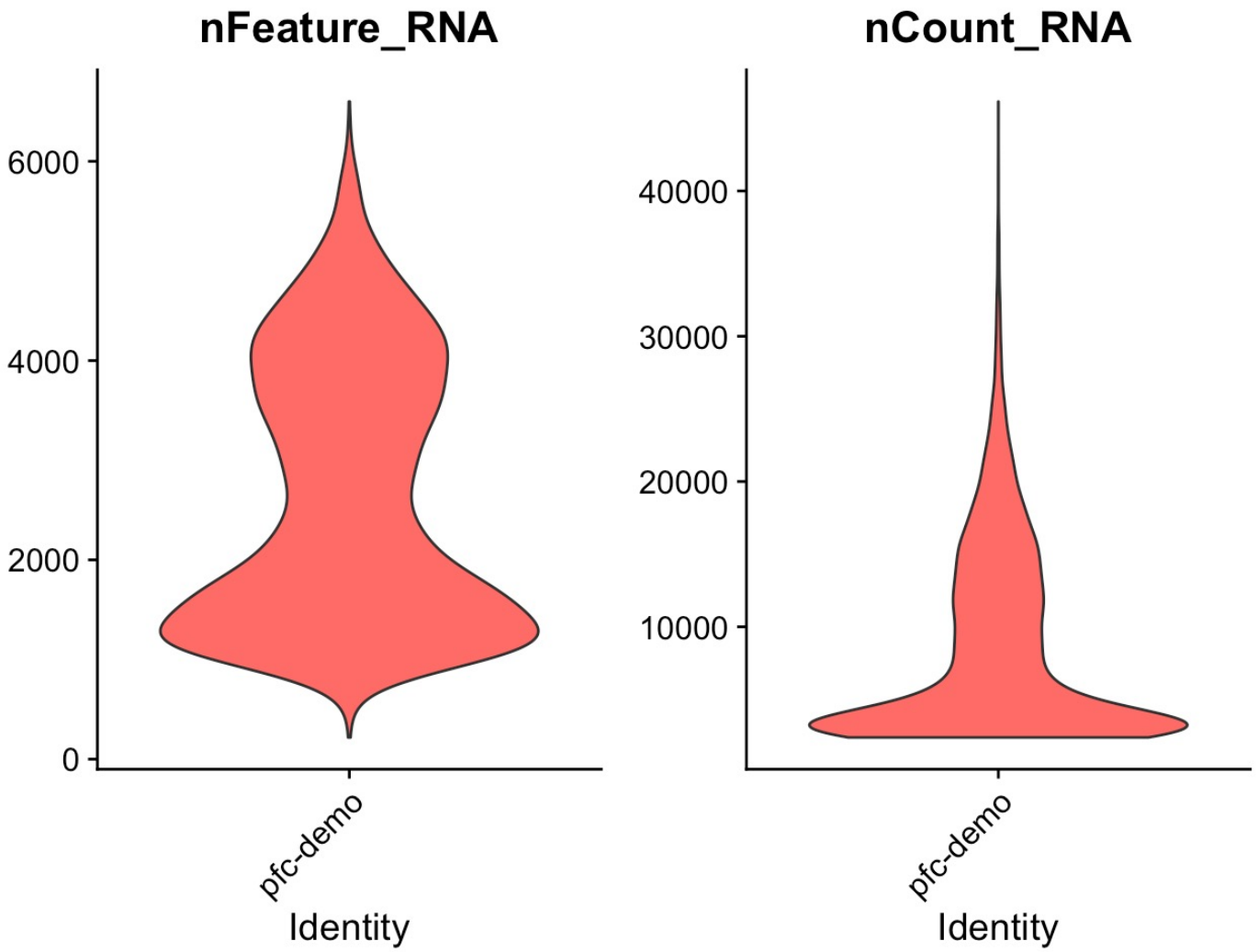
- Doublets

**Note: I rarely use a high cutoff in initial QC filtering and instead prefer to remove doublets during clustering – high nFeature_RNA is still a strong indicator of doublets at that stage too.

Typically want at least > 500 genes/cell, ideally > 1000 genes/cell

Quality Control Metrics – Visualizing nFeature_RNA & nCount_RNA

"Violin" Plots



Quality Control Metrics

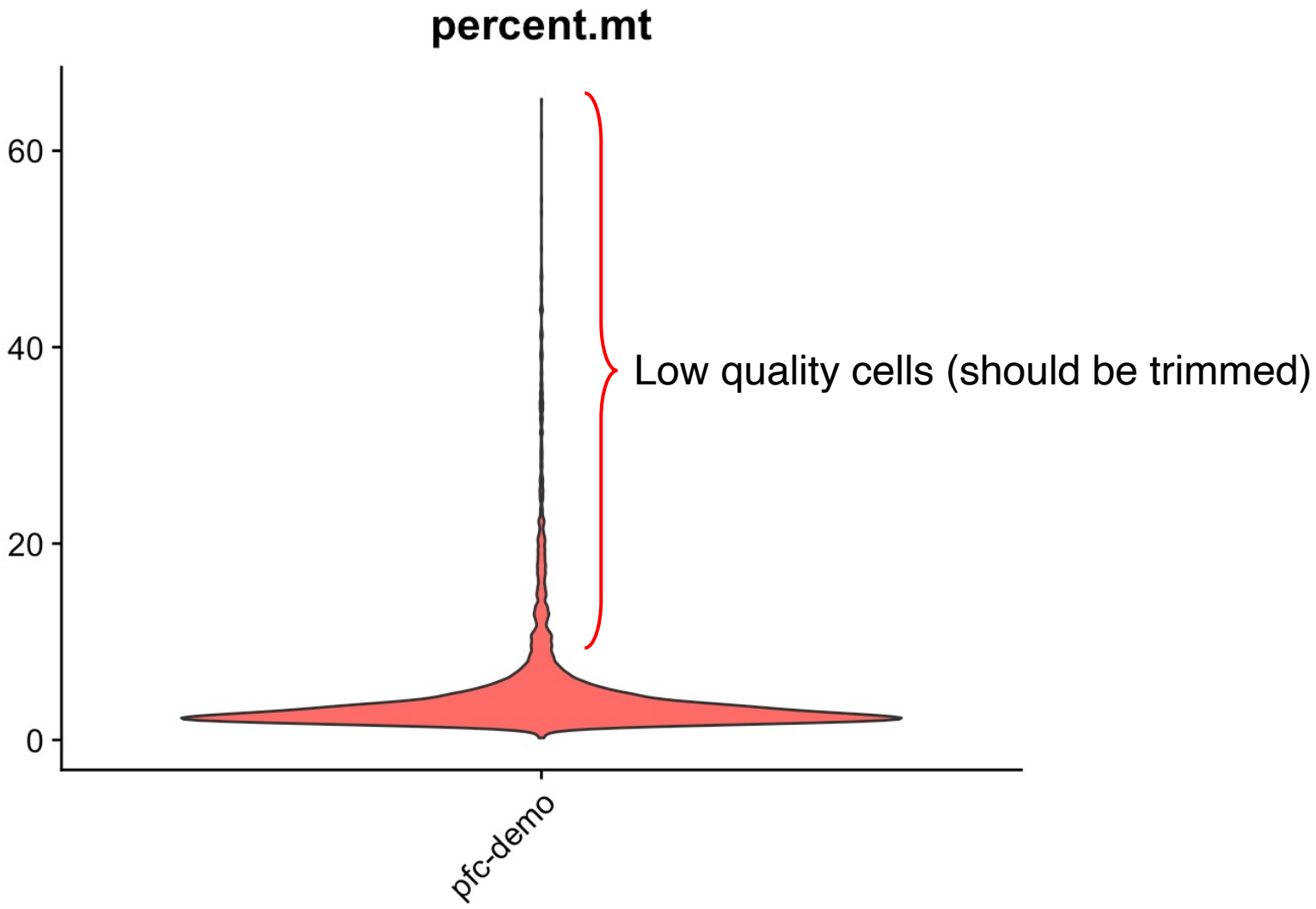
Goal: filter out low-quality cells and retain high-quality cells

Mitochondrial transcript ratio (‘percent.mt’)

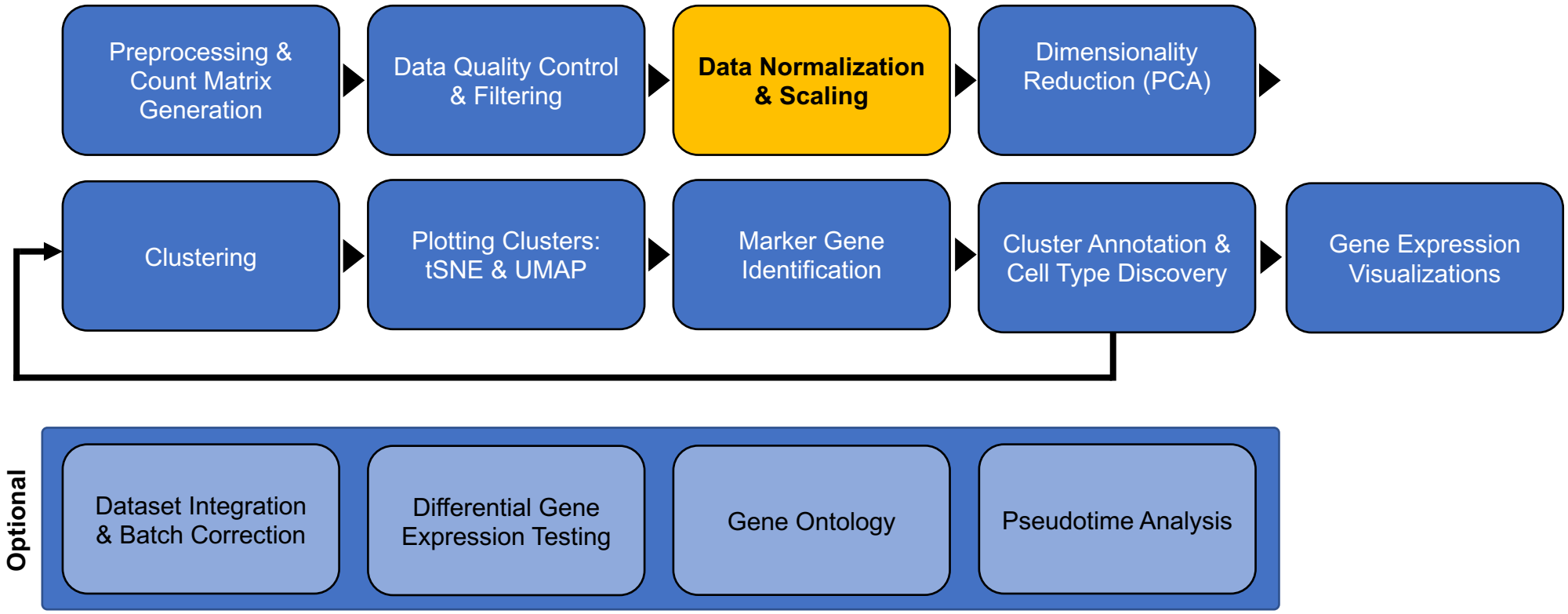
- Low quality and/or dying cells often exhibit a high proportion of mitochondrial transcripts.
- Typical range for ‘healthy’ cells is **< 5-10%**
- Note: If your cell type of interest is known to have a high proportion of mitochondrial transcripts, consider relaxing this threshold.

[Source: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4758103/]

Quality Control Metrics – Visualizing percent.mt



Single Cell Experiment Outline



Data Normalization

Variance of gene expression values should reflect biological variation between cells

Challenge: remove sources of technical variance that may confound our analysis

- Variable mRNA content of cells (often related to cell size)
- Variable rates of mRNA capture between cells
- Variable sequencing depth between cells

Data Normalization

	Cell A	Cell B	Δ
Gene1	1	2	1
Gene2	100	200	100

Data Normalization

	Cell A	Cell B	Δ
Gene1	1	2	1
Gene2	100	200	100
Total UMI	10,000	2,500	

Data Normalization

1. Divide gene's UMI count by the total UMIs in each cell
2. Multiply this value by a scale factor (10,000 by default)

Raw Data → (1) (2) → Normalized Data

	Cell A	Cell B	Δ
Gene1	1	2	1
Gene2	100	200	100
Total UMI	10,000	2,500	

Cell A	Cell B	Δ
2	8	6
100	800	700

[Source: https://www.youtube.com/watch?v=huxkc2GH4lk]

Data Normalization

1. Divide gene's UMI count by the total UMIs in each cell
2. Multiply this value by a scale factor (10,000 by default)
3. Log-transform the result

Raw Data → (1, 2) → Normalized Data → (3) → Log Transformed Data

	Cell A	Cell B	Δ
Gene1	1	2	1
Gene2	100	200	100
Total UMI	10,000	2,500	

Cell A	Cell B	Δ
2	8	6
100	800	700

Cell A	Cell B	Δ
0.69	2.08	1.39
4.61	6.69	2.08

[Source: https://www.youtube.com/watch?v=huxkc2GH4lk]

Data Normalization – Alternative methods

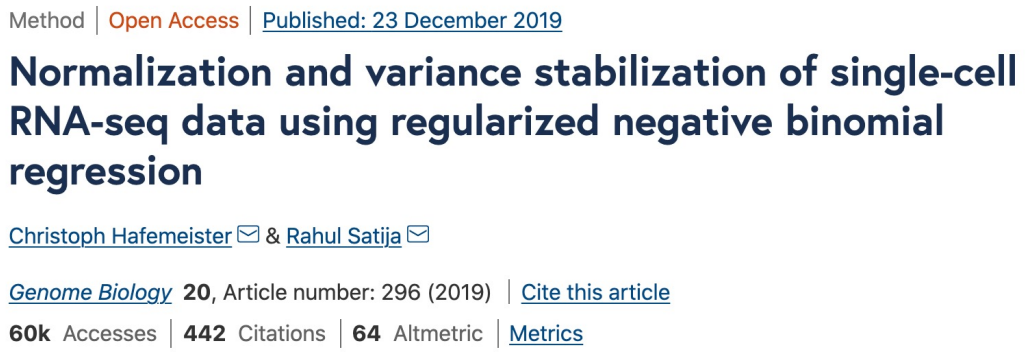
Method	Author	Year	Spike-ins	Model Description
SAMstrt	Katayama et al.	2013	Yes	Poisson resampling and non-parametric statistics
BASiCS	Vallejos et al.	2015	Yes	Use spike-ins for hierarchical Poisson/Gamma model for technical variability. Expand model to incorporate biological genes with new Poisson model
GRM	Ding et al.	2015	Yes	Gamma regression model from spike-ins
Simple Norm.	Satija et al.	2015	No	Divide gene counts for cells, then multiply by scale factor and apply a log(x +1) transformation to the result (included in the Seurat package as NormalizeData)
scran	Lun et al.	2016	No	Deconvolution of size factors from constructed linear system. Form pools of cells and normalize using summed expression values
SCnorm	Bacher et al.	2017	Optional	Quantile based model for log sequencing depth.
Linnorm	Yip et al.	2017	Optional	Linear models defined with a normalization strength coefficient to update means. Focuses on stable genes to perform normalization

[Source: https://www.frontiersin.org/articles/10.3389/fgene.2020.00041/full]

Data Normalization and Scaling with SCTransform

- SCTransform is a powerful tool for normalization and scaling of scRNA-seq data

- "The sctransform method models the UMI counts using a **regularized negative binomial model** to remove the variation due to sequencing depth (total nUMIs per cell), while adjusting the variance based on pooling information across genes with similar abundances"



Method | Open Access | Published: 23 December 2019

Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression

Christoph Hafemeister & Rahul Satija

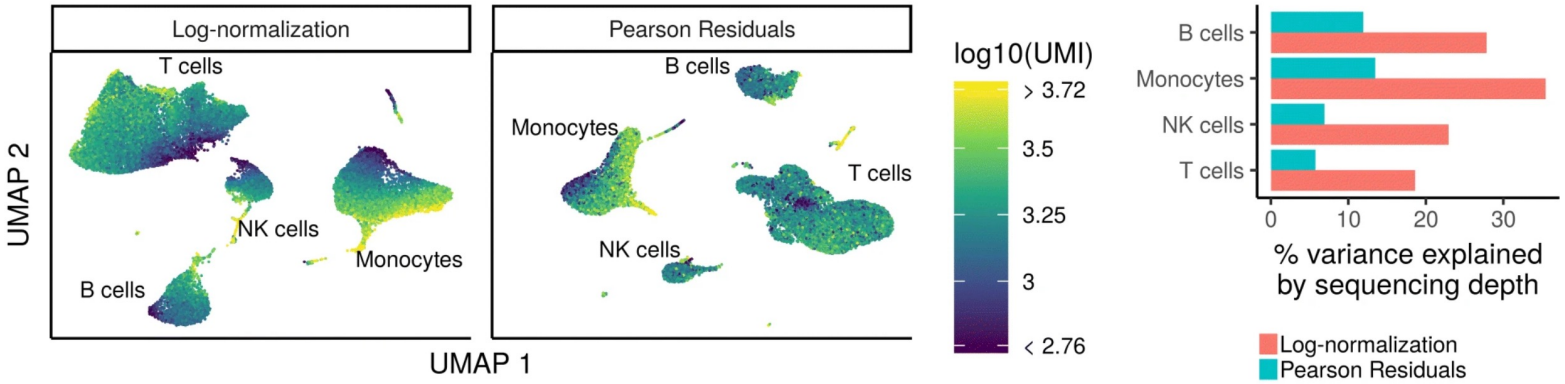
Genome Biology **20**, Article number: 296 (2019) | Cite this article

60k Accesses | **442** Citations | **64** Altmetric | Metrics

Data Normalization and Scaling with SCTransform

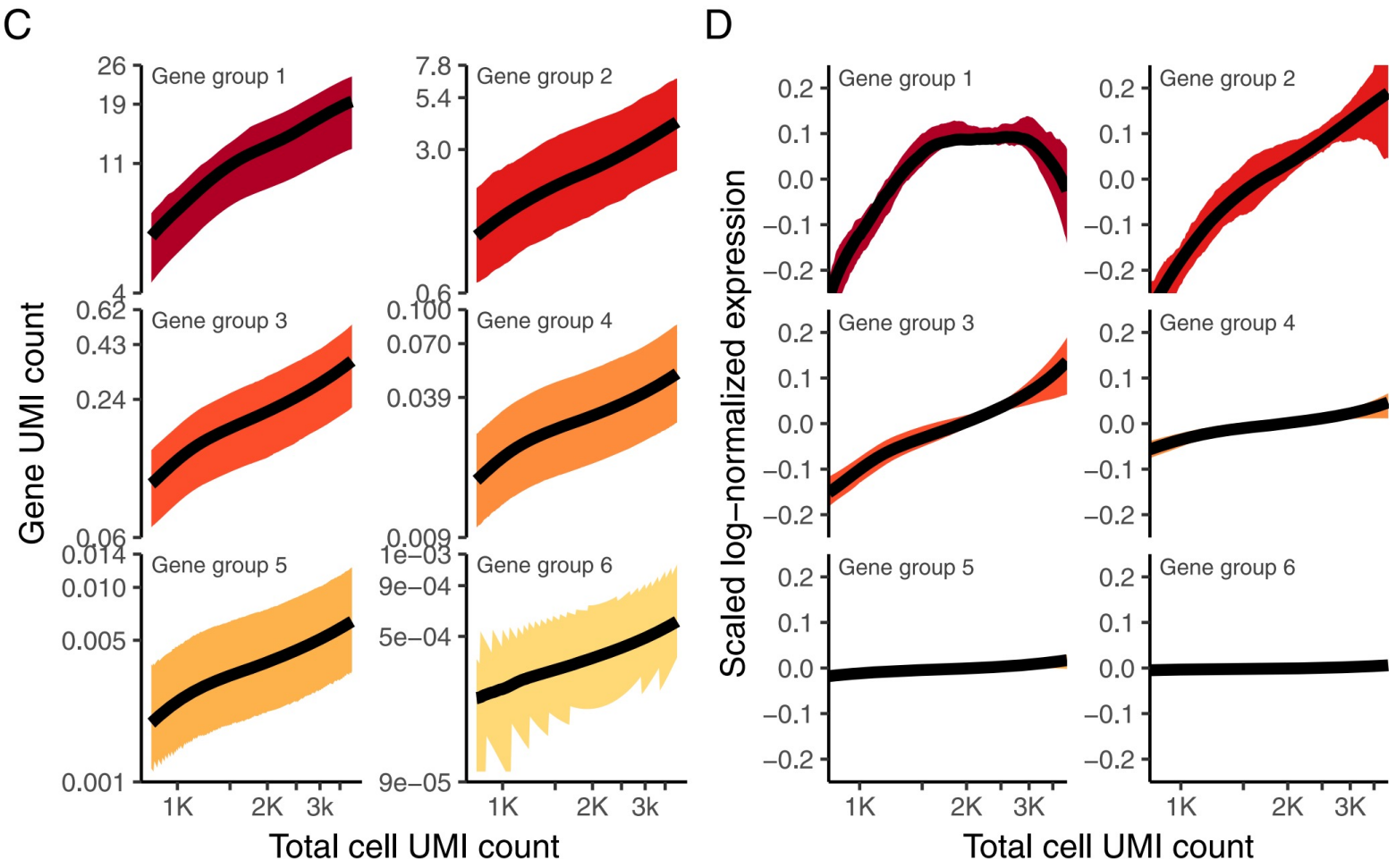
- SCTransform is a powerful tool for normalization and scaling of scRNA-seq data

- "The sctransform method models the UMI counts using a **regularized negative binomial model** to remove the variation due to sequencing depth (total nUMIs per cell), while adjusting the variance based on pooling information across genes with similar abundances"



[Sources: https://hbctraining.github.io/scRNA-seq/lessons/06_SC_SCT_and_integration.html
Hafemeister & Satija, *Genome Biology* 2019]

Data Normalization and Scaling with SCTransform

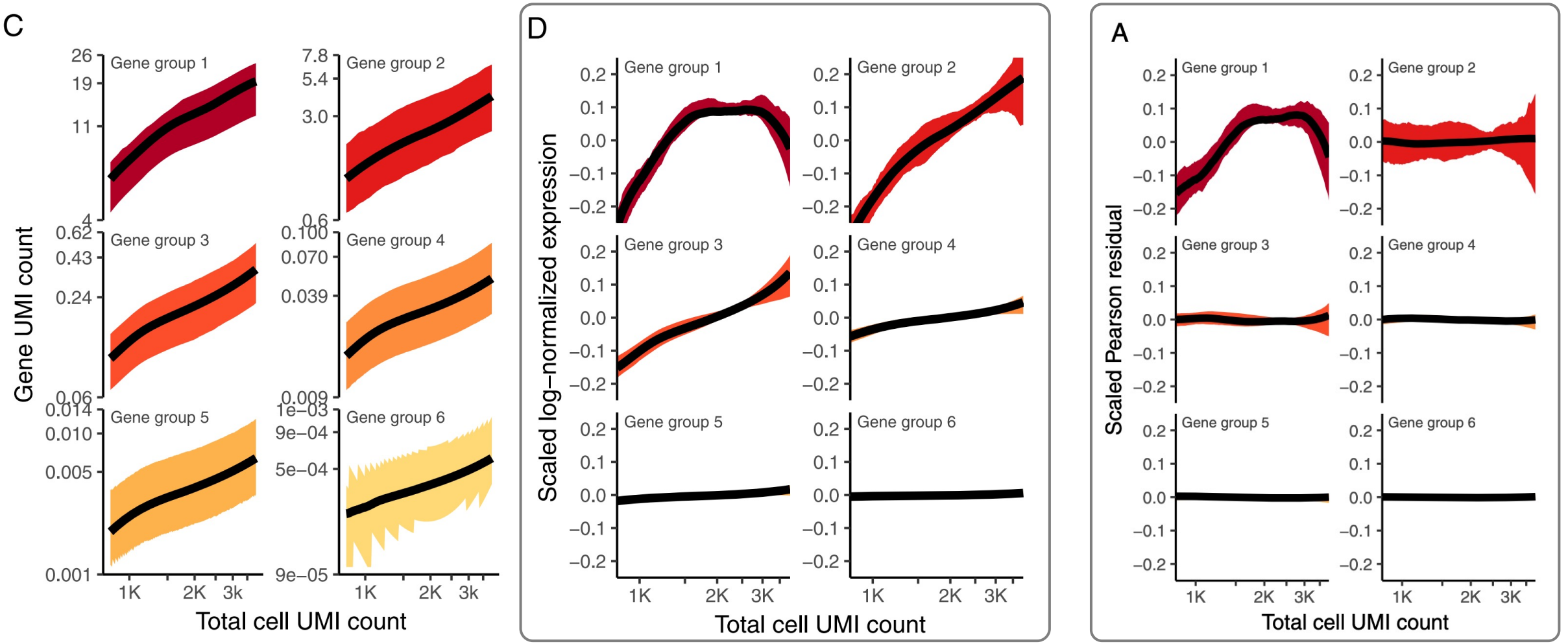


[Source: Hafemeister & Satija, *Genome Biology* 2019]

Data Normalization and Scaling with SCTransform

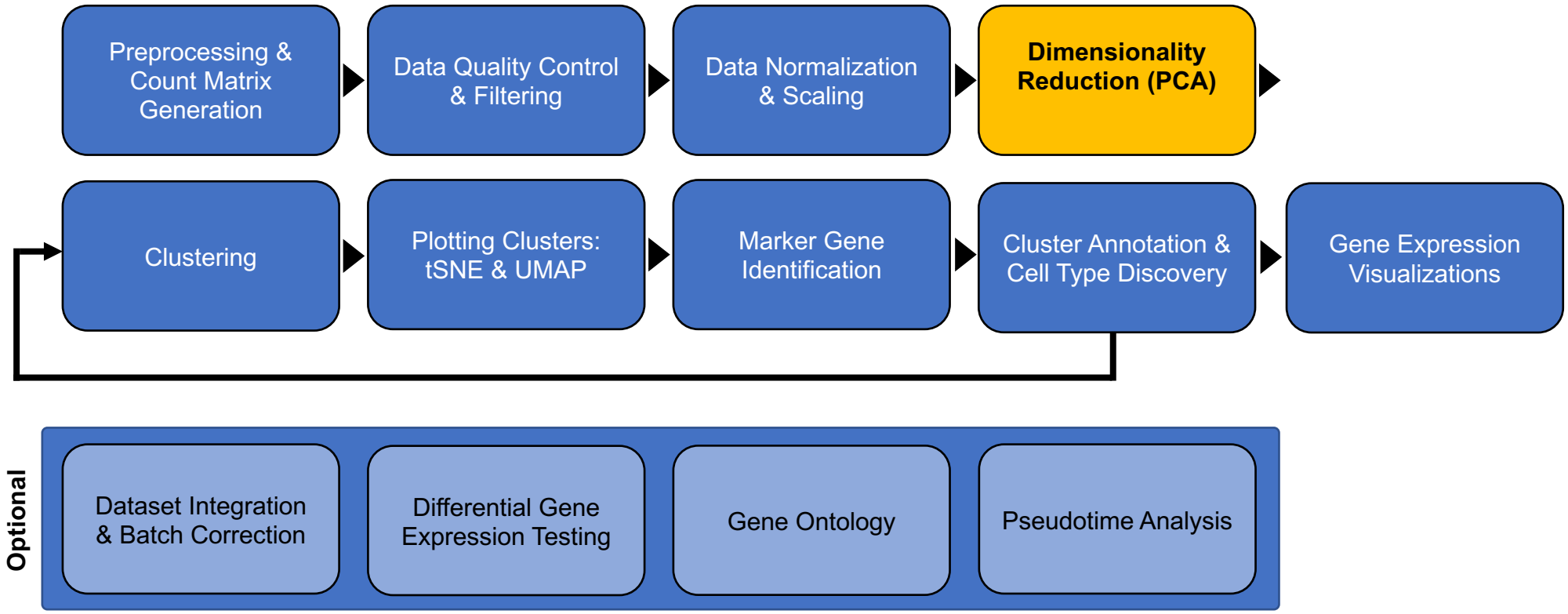
Simple Scaling/Normalization

scTransform Scaling/Normalization



[Source: Hafemeister & Satija, *Genome Biology* 2019]

Single Cell Experiment Outline



Principal Component Analysis

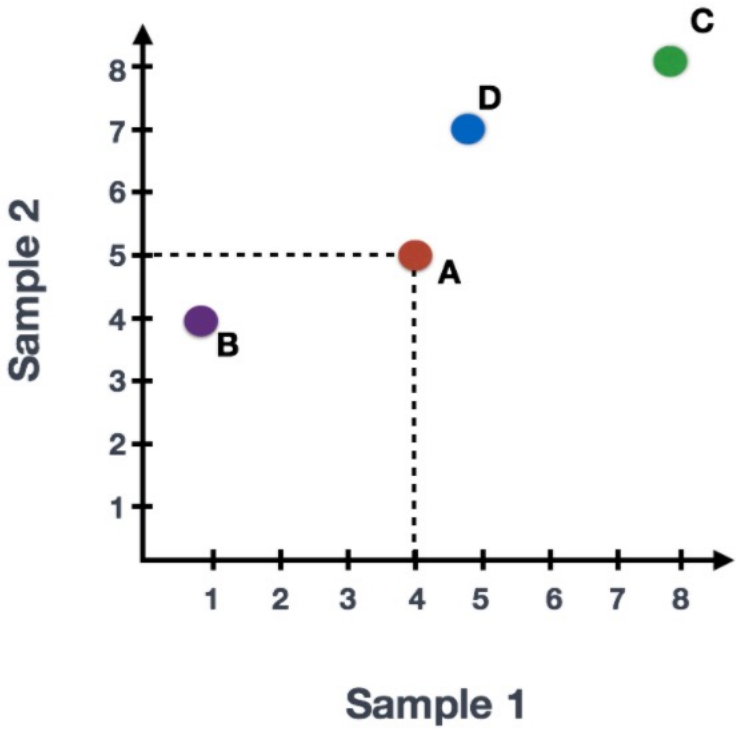
Principal component analysis (PCA) is a technique for **reducing the dimensionality of complex datasets**, increasing interpretability while minimizing information loss.

PCA is used to **represent a multivariate data table as smaller set of variables** in order to observe trends, jumps, clusters and outliers.

This overview may uncover the relationships between observations and variables, and among the variables.

[Sources: https://www.sartorius.com/en/knowledge/science-snippets/what-is-principal-component-analysis-pca-and-how-it-is-used-507186
https://hbctraining.github.io/scRNA-seq/lessons/05_normalization_and_PCA.html
https://doi.org/10.1098/rsta.2015.0202]

Principal Component Analysis

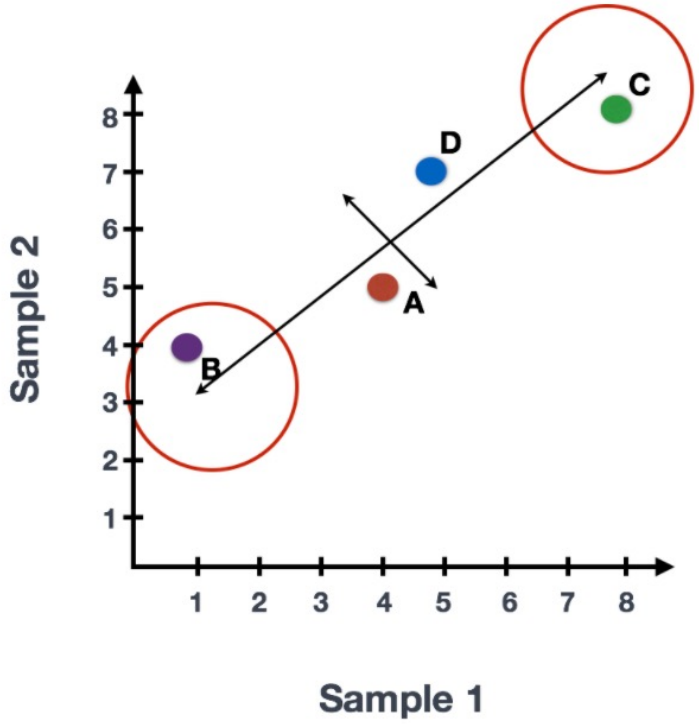


	Sample 1	Sample 2
Gene A	4	5
Gene B	1	4
Gene C	8	8
Gene D	5	7

[Source: https://hbctraining.github.io/scRNA-seq/lessons/05_normalization_and_PCA.html]

Principal Component Analysis

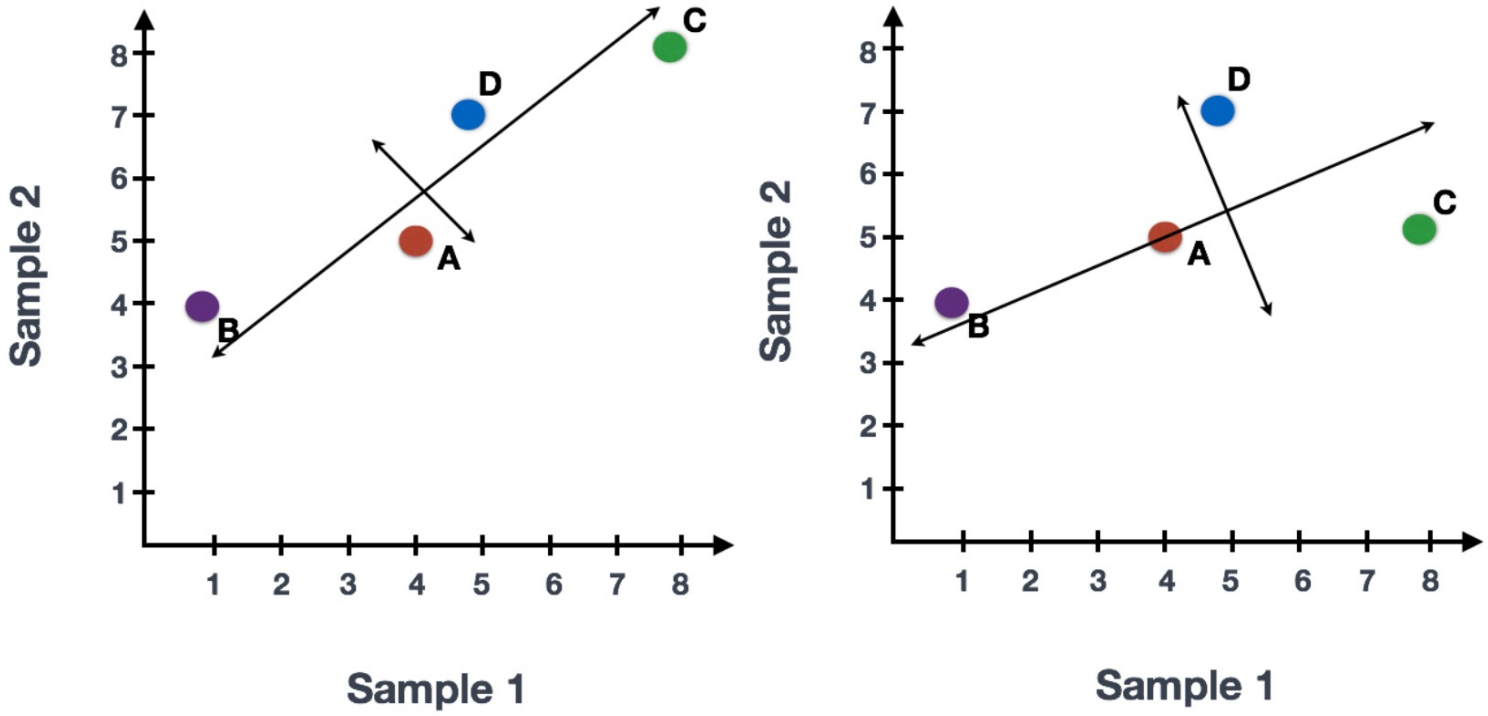
The genes near the ends of each line would be those with the highest variation; these genes have the greatest influence on the direction of the line, mathematically.



[Source: https://hbctraining.github.io/scRNA-seq/lessons/05_normalization_and_PCA.html]

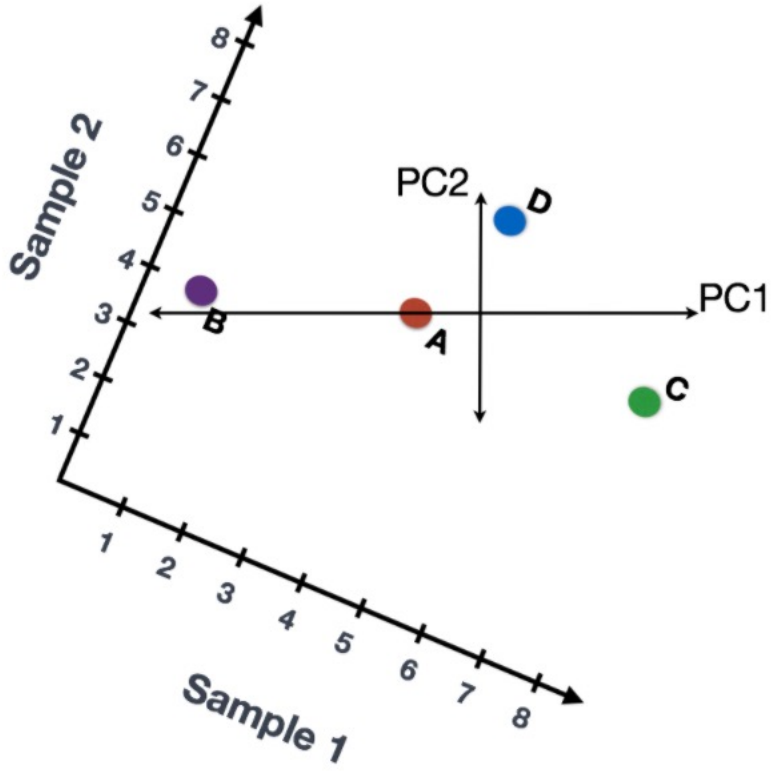
Principal Component Analysis

For example, a small change in the value of *Gene C* would greatly change the direction of the longer line, whereas a small change in *Gene A* or *Gene D* would have little affect on it.



[Source: https://hbctraining.github.io/scRNA-seq/lessons/05_normalization_and_PCA.html]

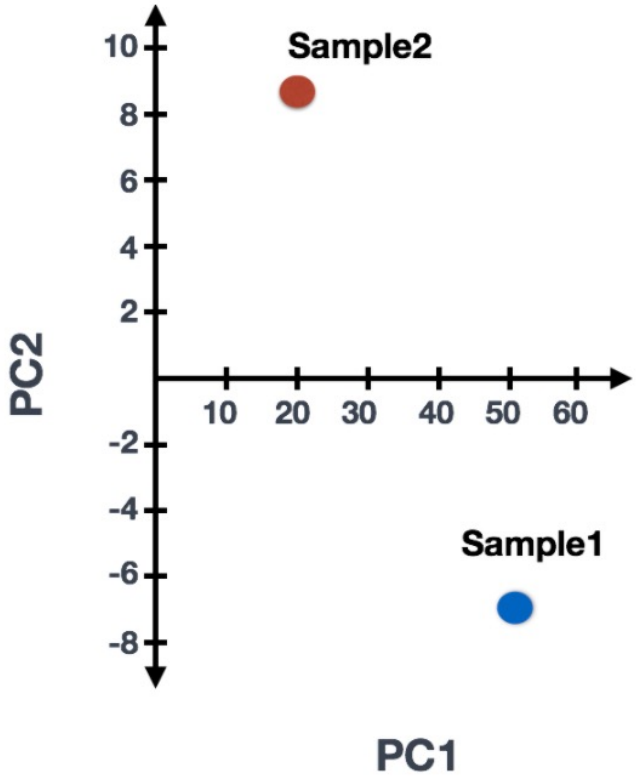
Principal Component Analysis



	Sample 1	Sample 2	Influence on PC1	Influence on PC2
Gene A	4	5	-2	0.5
Gene B	1	4	-10	1
Gene C	8	8	8	-5
Gene D	5	7	1	6

[Source: https://hbctraining.github.io/scRNA-seq/lessons/05_normalization_and_PCA.html]

Principal Component Analysis



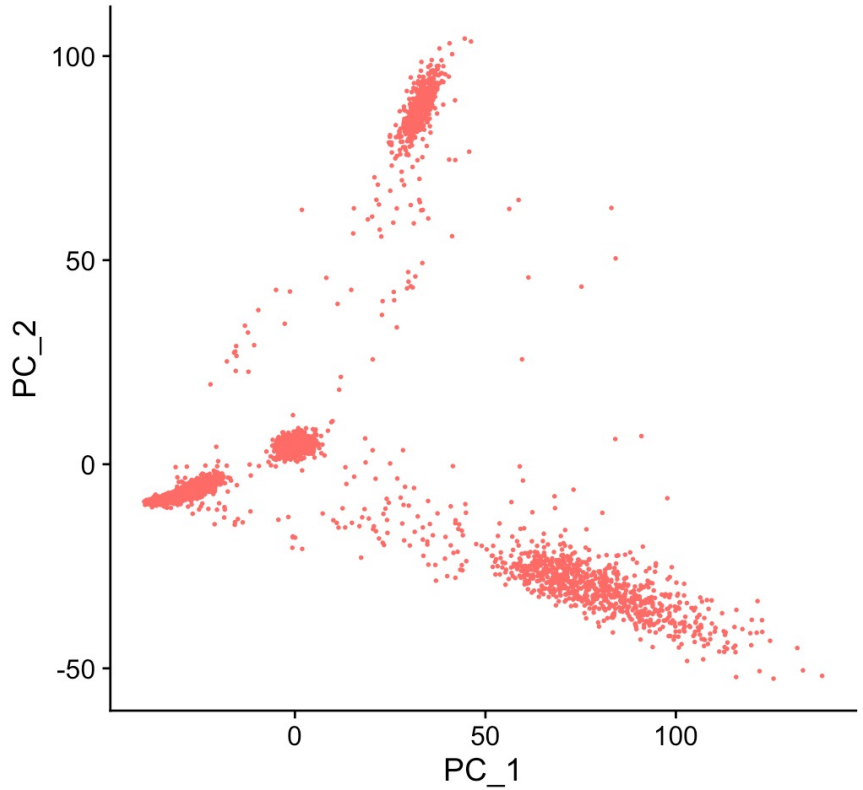
	PC1	PC2
Sample1	51	-7
Sample2	21	8.5

[Source: https://hbctraining.github.io/scRNA-seq/lessons/05_normalization_and_PCA.html]

Principal Component Analysis

Top 2 principal components

5118 prefrontal cortex cells



Dimensionality Reduction Methods

- Following normalization and scaling, we can calculate the top Principal Components (PC) for our data

- Typically 20-70 PCs are sufficient to explain complex single cell data

  (Which is a huge reduction from ~20,000! dimensions)

- These PCs are used to generate ”clusters” of cells that have similar programs of gene expression

Principal Component Analysis

Principal Components have positively and negatively associated genes that inform the variance contained within

Microglia-associated genes

Neuron-associated genes

Oligodendrocyte-associated genes

```
PC_ 1
Positive:  Fos, Sepp1, Ly6c1, Itm2a, Ly6a, Bsg, Sparc, Id1, Flt1, Id3
           Pltp, Ifitm3, Pglyrp1, Slc2a1, Rgs5, B2m, Btg2, Nfkbia, Myl12a, Cldn5
           Slco1a4, Gng11, Ctla2a, Vtn, Atp1a2, Tm4sf1, H2-D1, Esam, Ly6e, Epas1
Negative:  Meg3, Stmn1, Snap25, Calm2, Snrpn, Rtn1, Snhg11, Snca, Atp6v0c, Sncb
           Pcp4, Aldoa, Stmn3, Vsnl1, Zwint, Bex2, Syt1, Mdh1, Atp6v1g2, Hsp90aa1
           Mtap1b, Calm1, Dynll1, Mef2c, Arpp21, Mllt11, Stmn2, Gap43, Chn1, Ttc3
PC_ 2
Positive:  Cst3, Ctss, C1qa, Tyrobp, C1qb, C1qc, Fcer1g, Hexb, Ctsd, Trem2
           Lgmn, P2ry12, Selplg, Laptm5, Csf1r, Rgs10, Aif1, Fcgr3, Fcrls, Cx3cr1
           Gpr34, Ly86, Tmem119, Cd53, Olfml3, Siglech, Ctsz, Trf, Unc93b1, Ccl3
Negative:  Bsg, Ly6c1, Itm2a, Ly6a, Flt1, Id1, Pltp, Id3, Pglyrp1, Rgs5
           Slc2a1, Ifitm3, Slco1a4, Cldn5, Gng11, Ctla2a, Vtn, Esam, Tm4sf1, Epas1
           Cald1, Meg3, Egfl7, Cxcl12, Car4, Ptprb, Ramp2, Pcp4l1, Abcg2, Ifitm2
PC_ 3
Positive:  Meg3, Snap25, Stmn1, Calm2, Snca, Tmsb4x, Snrpn, Rtn1, Sncb, Tmsb10
           Snhg11, B2m, Atp6v0c, Atf3, Vsnl1, Nfkbia, Sparc, Stmn3, Zwint, Calm1
           Mef2c, Pcp4, Zfp36, Syt1, Aldoa, Dynll1, Cyba, Bex2, Bsg, H3f3b
Negative:  Gpr37l1, Dbi, Apoe, Ndrg2, Aldoc, Mt1, Slc1a3, Glul, Clu, Mt2
           Prdx6, Car2, Mfge8, Tubb2b, Atp1a2, Cldn10, S100a1, Slc1a2, Pla2g7, Gstm1
           Ppap2b, Phgdh, S100a16, Acsbg1, Gstm5, Tsc22d4, Ckb, Cst3, Mt3, S100b
PC_ 4
Positive:  Plp1, Cnp, Cldn11, Mbp, Cryab, Mag, Sept4, Mobp, Mog, Pllp
           Mal, Ermn, Gatm, Trf, Ptgds, Apod, Aspa, Gsn, Enpp2, Qdpr
           Fth1, Opalin, Cd9, Tspan2, Pdlim2, Tnfaip6, Sirt2, Tubb4a, Gpr37, Csrp1
Negative:  Cst3, Atp1a2, Clu, Slc1a2, Aldoc, Mt3, Mfge8, Mt2, Slc1a3, Cldn10
           Acsbg1, Prdx6, Pla2g7, Gstm1, Gja1, F3, Gjb6, Ppap2b, Sparcl1, Ndrg2
           Mlc1, Gpr37l1, Tst, Aqp4, Mt1, Mgll, Fam107a, S1pr1, Slc9a3r1, Apoe
PC_ 5
Positive:  Car2, Glul, Csrp1, Fth1, Gstm1, Trf, Mal, Mobp, Ermn, Apod
           Mog, Cst3, Sept4, Aspa, Slc1a2, Acsbg1, Mt1, Sepp1, Mt2, Mag
           Gja1, Cldn11, Tst, Mbp, Il33, Phgdh, Pdlim2, Gjb6, Opalin, Qdpr
Negative:  Serpine2, Scrg1, S100a1, Ptprz1, Pdgfra, Tmem100, Neu4, Qpct, S100a16, Gpr17
           Cd9, Fabp7, Ednrb, Tmem176b, Vcan, Cacng4, S100a13, Fos, Rlbp1, Ccnd1
           Lims2, Cdo1, Gm2a, Dbi, Olig1, Nxph1, Nnat, G0s2, S100b, Tmem176a
```

How many principle components should I include?

From Seurat authors: “Interestingly, we’ve found that when using SCTransform, we often benefit by pushing this parameter even higher. We believe this is because the SCTransform workflow performs more effective normalization, strongly removing technical effects from the data.”

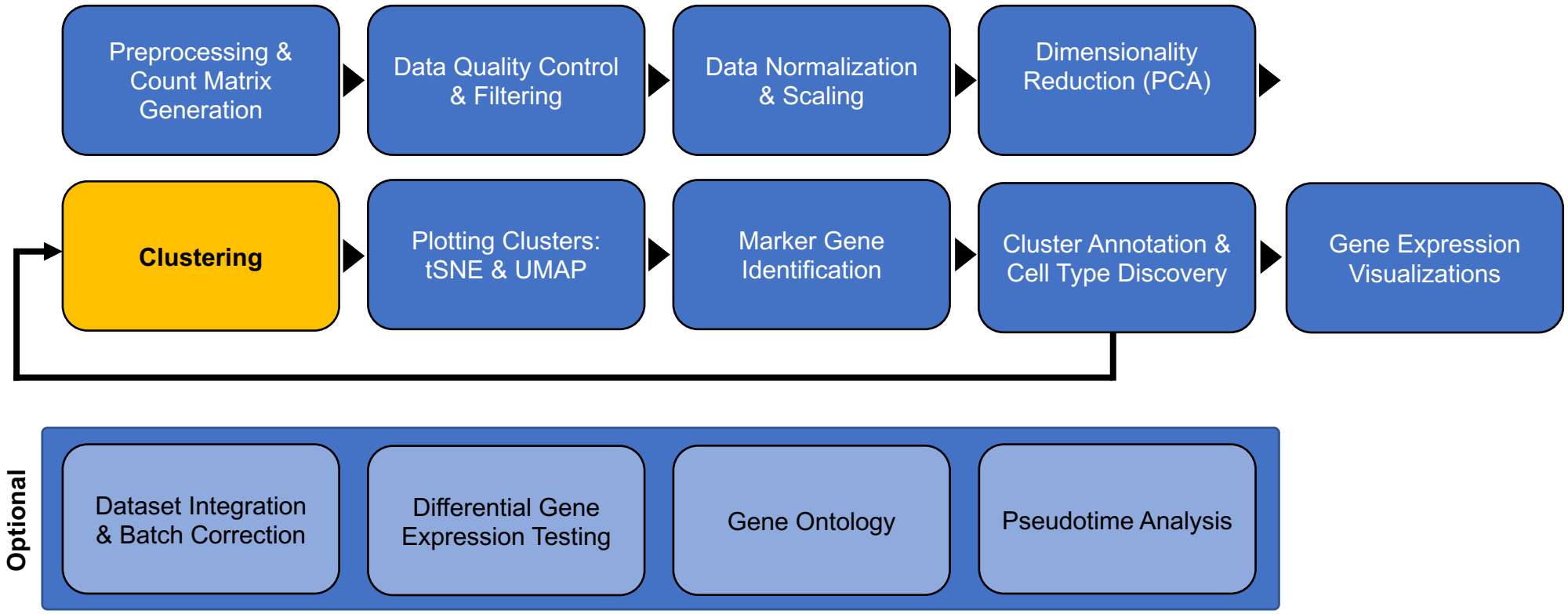
In general, iteratively increasing/decreasing is a good way to determine PC number

By examining each PC’s ”gene loading”, it is possible to see if PCs contain biologically meaningful gene sets, or are perhaps influenced by technical or non-biological signals

- For datasets normalized without SCTransform, see:
  https://satijalab.org/seurat/articles/pbmc3k_tutorial.html#determine-the-dimensionality-of-the-dataset-1

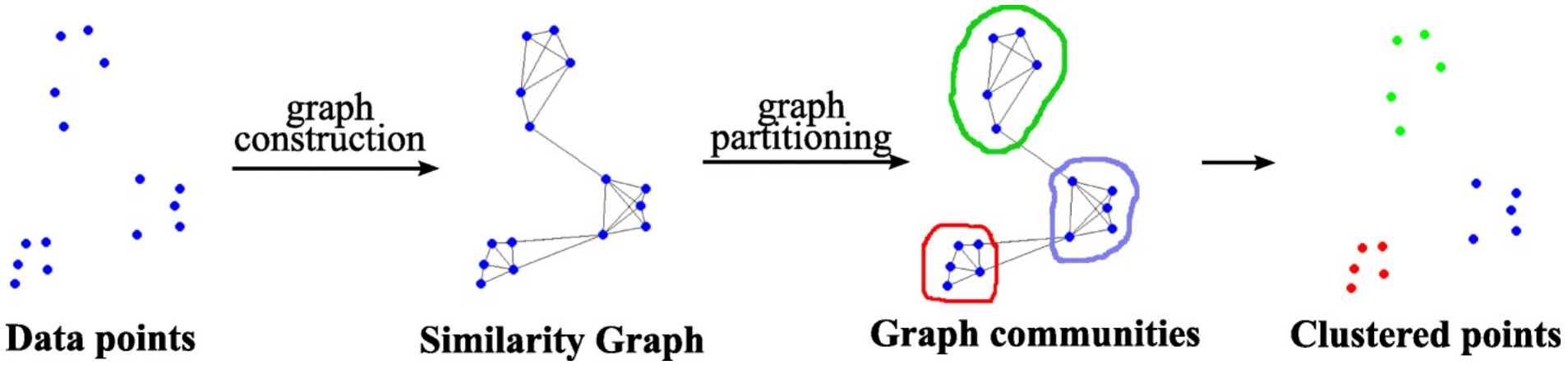
- For datasets normalized with SCTransform:
  https://satijalab.org/seurat/articles/sctransform_vignette.html

Single Cell Experiment Outline



Clustering

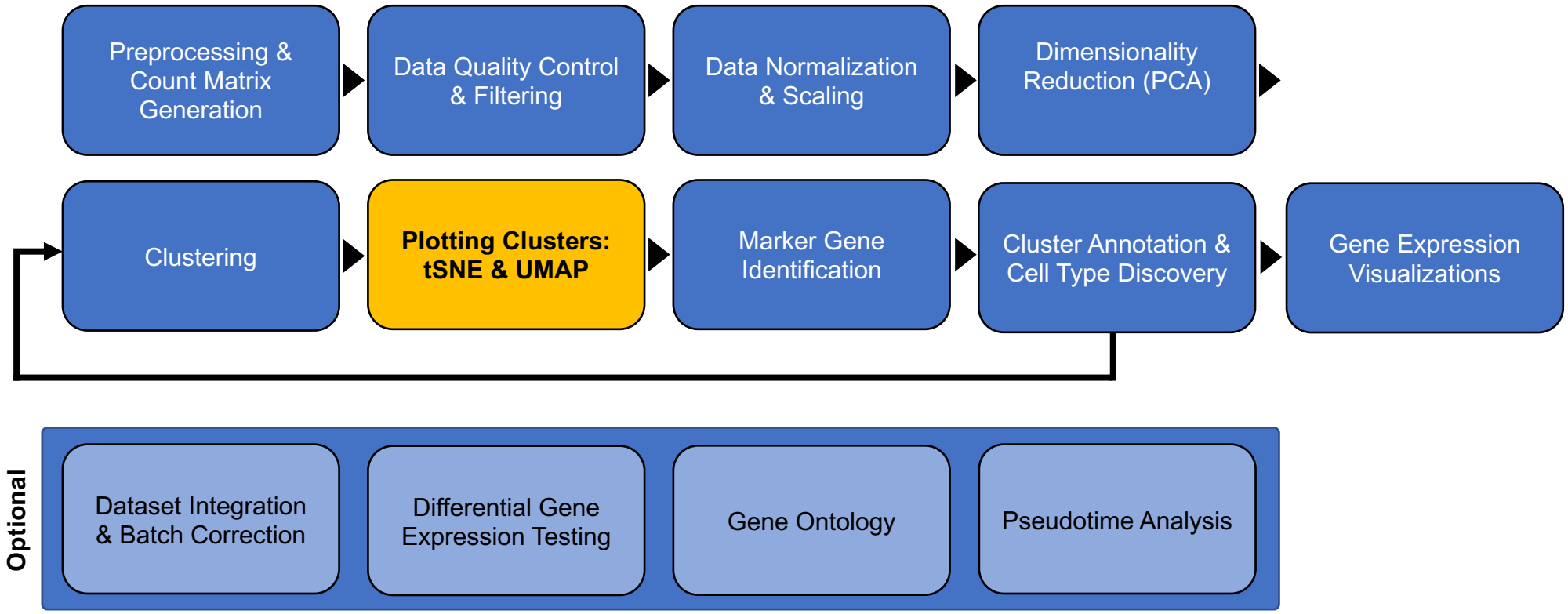
"Briefly, these methods embed cells in a graph structure - for example a K-nearest neighbor (KNN) graph, with edges drawn between cells with similar feature expression patterns, and then attempt to partition this graph into highly interconnected 'quasi-cliques' or 'communities'."



The above example demonstrates 2-dimensional graph-based clustering.
Single cell clustering will be *n*-dimensional, where *n* is the number of input Principal Components

[Source: https://satijalab.org/seurat/archive/v3.1/pbmc3k_tutorial.html https://appliednetsci.springeropen.com/articles/10.1007/s41109-019-0248-7]

Single Cell Experiment Outline



tSNE & UMAP

- Challenge: graph-based clusters are generated from high-dimensional data (impossible to visualize)

- Solution: non-linear dimensionality reduction approaches

> The goal of these algorithms is to learn the underlying manifold of the data in order to place similar cells together in low-dimensional space. Cells within the graph-based clusters determined above should co-localize on these dimension reduction plots. As input to the UMAP and tSNE, we suggest using the same PCs as input to the clustering analysis.
>
> -satijalab.org

tSNE vs. UMAP

tSNE

"**t**-distributed **S**tochastic **N**eighbor **E**mbedding"

Does not scale well for large datasets (bottleneck)

Does not preserve global structure

UMAP

"**U**niform **M**anifold **A**pproximation and **P**rojection"

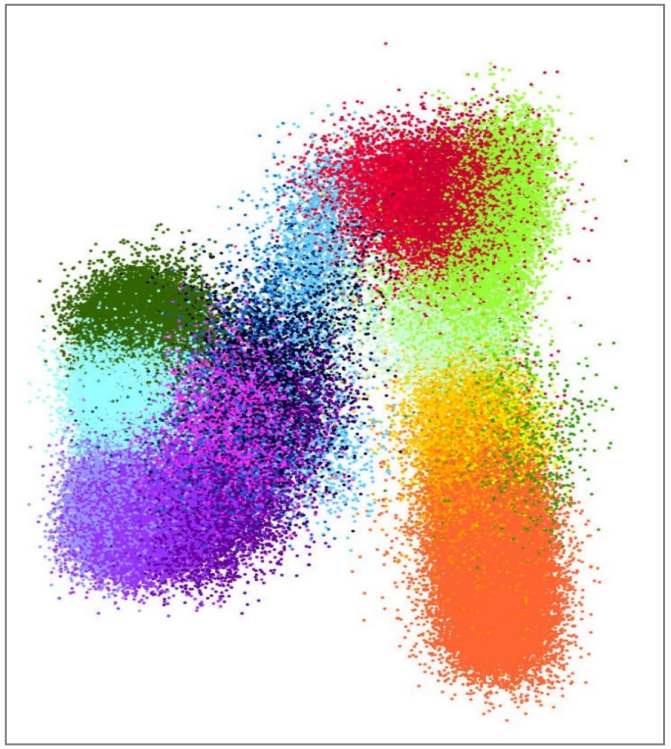
Scales for large datasets

Preserves global structure

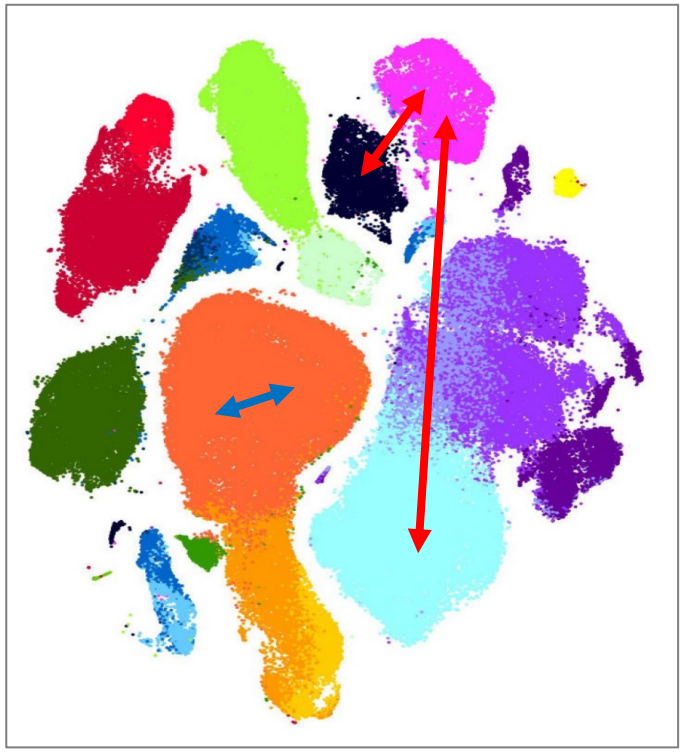
For a detailed explanation of the mathematical differences between tSNE and UMAP, see:
https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668

UMAP preserves global similarity and has improved interpretability

PCA



tSNE



Points within a cluster are similar (local distance)

Distance between clusters does not reflect similarity (~~global distance~~)

UMAP

Points within a cluster are similar (local distance)

Distance between clusters does reflect similarity (global distance)

[Image Source: https://www.frontiersin.org/articles/10.3389/fcell.2020.00234/full]

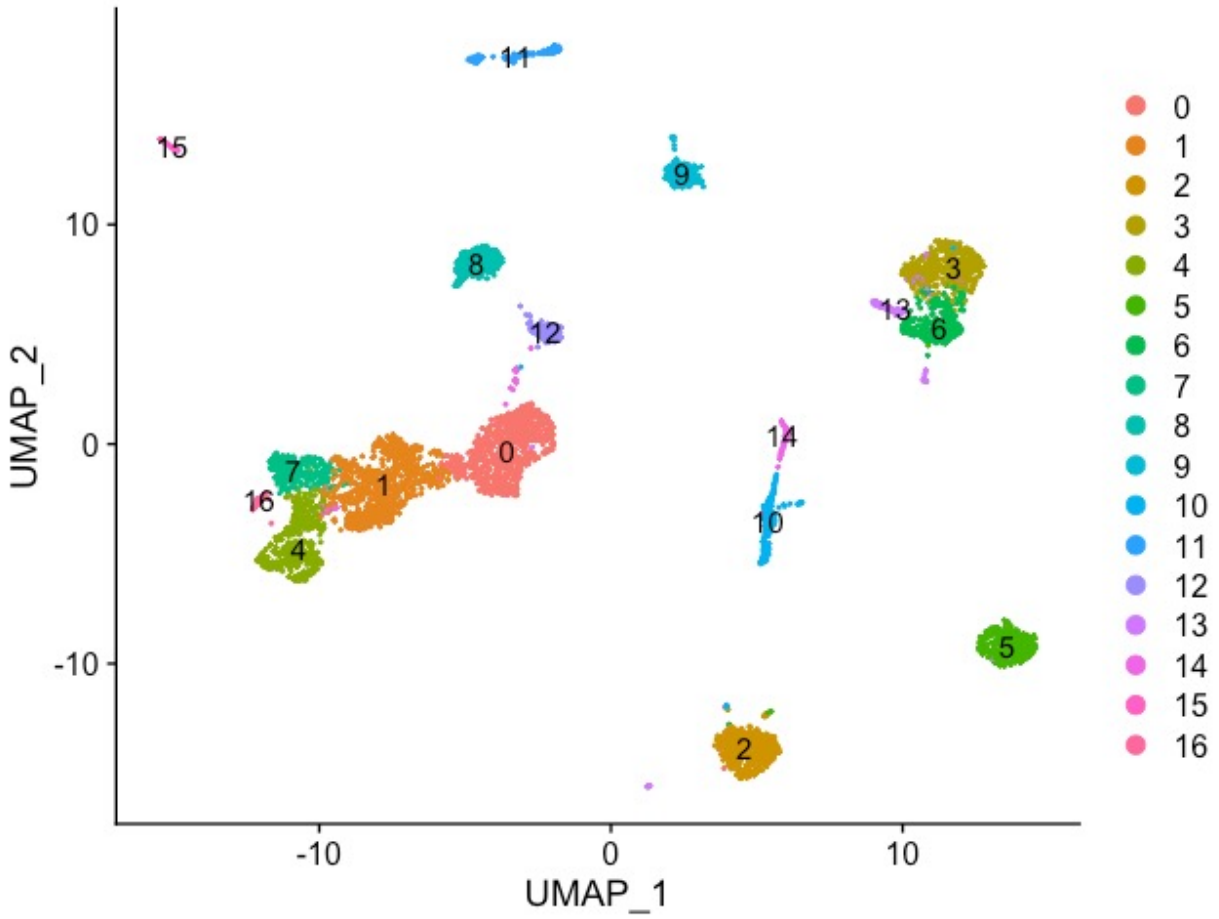
UMAP preserves global similarity and has improved interpretability

5118 prefrontal cortex cells

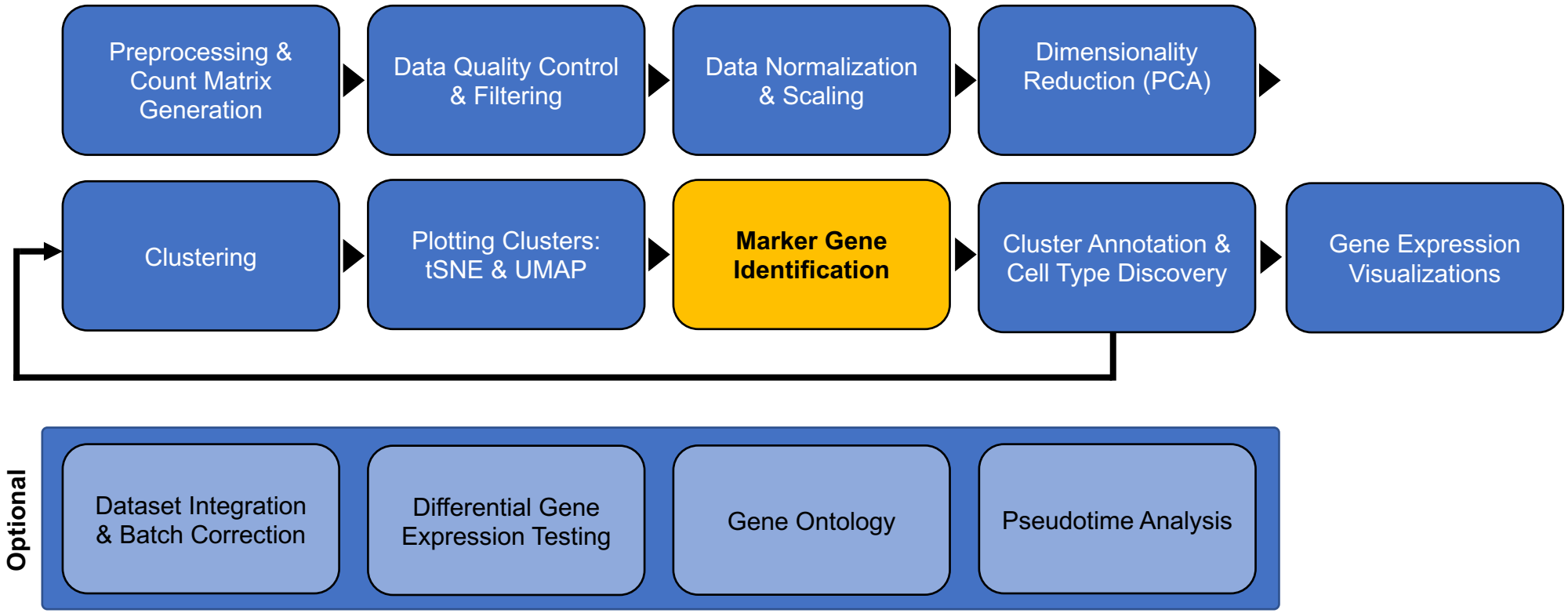
PCA and Clustering performed with 60 PCs

17 Clusters of cells

UMAP generated from 60 PCs



Single Cell Experiment Outline



Marker Gene Identification

Inputs

- Dataset name
- Cluster of interest
- Cluster to compare (optional, default = all other clusters)
- Statistical test to use
- Expression threshold (percentage)
- Enrichment threshold (log fold change)

Outputs

- `avg_logFC`: log fold-chage of the average expression between the two groups. Positive values indicate that the gene is more highly expressed in the first group
- `pct.1`: The percentage of cells where the gene is detected in the first group
- `pct.2`: The percentage of cells where the gene is detected in the second group
- `p_val_adj`: Adjusted p-value, based on bonferroni correction using all genes in the dataset

[Source: https://www.rdocumentation.org/packages/Seurat/versions/3.1.3/topics/FindMarkers]

Marker Genes – Enrichment vs. Specificity

Sample Marker Gene Identification Output

	p_val	avg_log2FC	pct.1	pct.2	p_val_adj
Deptor	1.637081e-188	1.1815459	0.693	0.098	2.653872e-184
Sstr2	6.401468e-166	0.7499726	0.568	0.067	1.037742e-161
Tmem91	1.190652e-147	1.2051243	0.739	0.146	1.930167e-143
Pcp4	5.833237e-133	2.4150679	1.000	0.733	9.456260e-129
Ighg2c	2.887307e-131	0.5429715	0.377	0.034	4.680613e-127
Cadps2	1.642305e-130	0.6543565	0.572	0.086	2.662340e-126
Slc24a2	8.354747e-129	1.2690044	0.895	0.258	1.354388e-124
Il1rapl2	5.060726e-119	0.4106940	0.300	0.022	8.203943e-115
Dkkl1	1.242784e-106	1.1583628	0.798	0.219	2.014678e-102
Stmn2	1.346602e-106	1.1874354	1.000	0.966	2.182977e-102
Elavl4	2.781282e-106	1.2752329	0.918	0.385	4.508736e-102
BC048546	4.427534e-106	0.9453233	0.486	0.078	7.177476e-102
Hs3st2	8.660502e-102	0.7205145	0.619	0.133	1.403954e-97
Galnt9	2.468550e-101	0.5980814	0.568	0.108	4.001766e-97
Snap25	4.505057e-100	1.3393901	1.000	0.967	7.303147e-96
Scube1	1.863118e-99	0.5195003	0.436	0.064	3.020301e-95
Syt1	4.526528e-97	1.1448332	1.000	0.959	7.337954e-93

Tips for Enrichment vs Specificity

- Enrichment: High ‘avg_log2FC’
- Specificity: Low ‘pct.2’

Differential Gene Expression (DGE) Algorithms available in Seurat

- "wilcox" : Identifies differentially expressed genes between two groups of cells using a Wilcoxon Rank Sum test (default)
- "bimod" : Likelihood-ratio test for single cell gene expression, (McDavid et al., Bioinformatics, 2013)
- "roc" : Identifies 'markers' of gene expression using ROC analysis. For each gene, evaluates (using AUC) a classifier built on that gene alone, to classify between two groups of cells. An AUC value of 1 means that expression values for this gene alone can perfectly classify the two groupings (i.e. Each of the cells in cells.1 exhibit a higher level than each of the cells in cells.2). An AUC value of 0 also means there is perfect classification, but in the other direction. A value of 0.5 implies that the gene has no predictive power to classify the two groups. Returns a 'predictive power' (abs(AUC-0.5) * 2) ranked matrix of putative differentially expressed genes.
- "t" : Identify differentially expressed genes between two groups of cells using the Student's t-test.
- "negbinom" : Identifies differentially expressed genes between two groups of cells using a negative binomial generalized linear model. Use only for UMI-based datasets
- "poisson" : Identifies differentially expressed genes between two groups of cells using a poisson generalized linear model. Use only for UMI-based datasets
- "LR" : Uses a logistic regression framework to determine differentially expressed genes. Constructs a logistic regression model predicting group membership based on each feature individually and compares this to a null model with a likelihood ratio test.
- "MAST" : Identifies differentially expressed genes between two groups of cells using a hurdle model tailored to scRNA-seq data. Utilizes the MAST package to run the DE testing.
- "DESeq2" : Identifies differentially expressed genes between two groups of cells based on a model using DESeq2 which uses a negative binomial distribution (Love et al, Genome Biology, 2014).This test does not support pre-filtering of genes based on average difference (or percent detection rate) between cell groups. However, genes may be pre-filtered based on their minimum detection rate (min.pct) across both cell groups. To use this method, please install DESeq2, using the instructions at https://bioconductor.org/packages/release/bioc/html/DESeq2.html

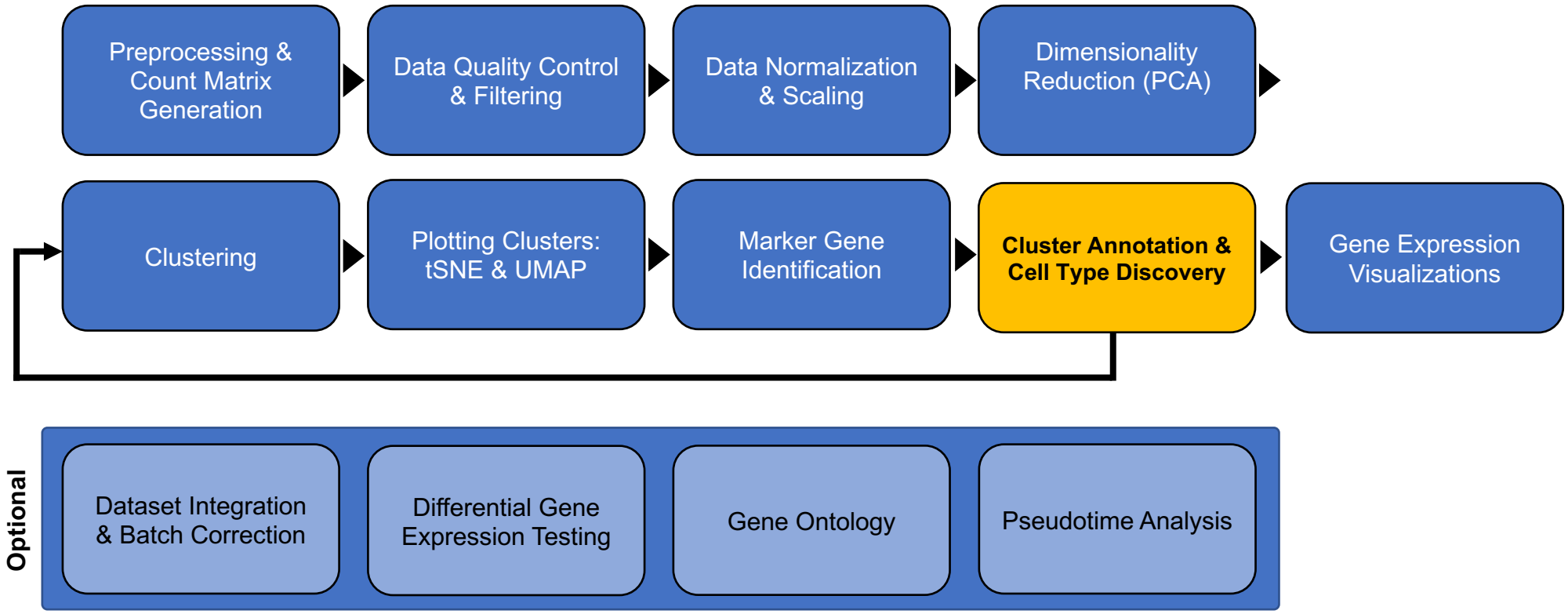
[Source: https://www.rdocumentation.org/packages/Seurat/versions/3.1.3/topics/FindMarkers]

Additional reading on DGE algorithms for single cell RNA-seq

- Reproducibility of Methods to Detect Differentially Expressed Genes from Single-Cell RNA Sequencing
  - https://www.frontiersin.org/articles/10.3389/fgene.2019.01331/full

- Confronting false discoveries in single-cell differential expression
  - “https://www.nature.com/articles/s41467-021-25960-2#Bib1

Single Cell Experiment Outline



Cluster Annotation and Cell Type Discovery

Goal: identify cell type(s) present in each cluster

Strategies

- Online gene expression databases
  - Allen Brain Atlas – https://portal.brain-map.org/
  - gTEX Portal – https://gtexportal.org/
- Published single cell resources/atlases
  - Broad single cell portal – https://singlecell.broadinstitute.org/single_cell
  - MouseBrain – https://mousebrain.org
  - DropViz – https://dropviz.org
- Pubmed!
- Gene Ontology analysis
  - http://geneontology.org/

A note on doublets and low-quality cells

During cluster identification and iterative clustering, it is common to encounter clusters that have signatures of either doublets or low-quality cells.

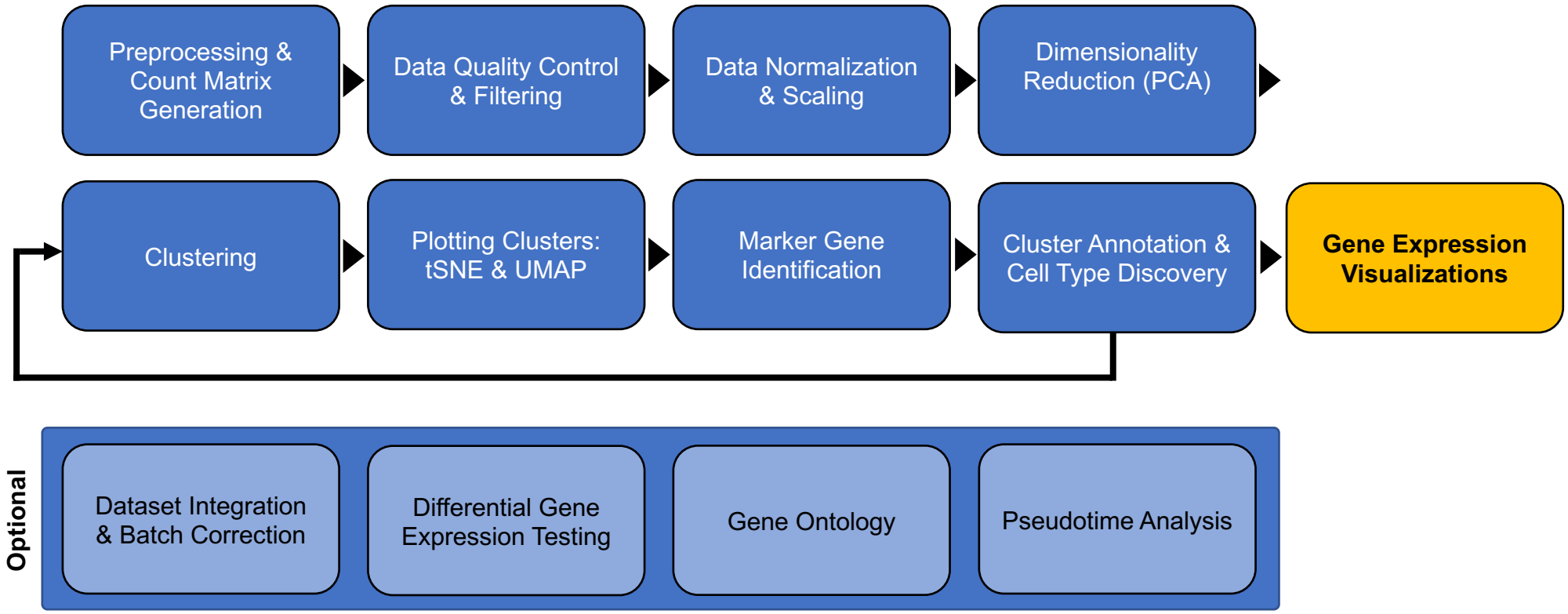
Doublets

- Overlapping signatures of multiple cell types
- In UMAP plots, doublet clusters may be positioned between the 2+ cell types contained in the doublets
- Higher nFeature_RNA/nCount_RNA

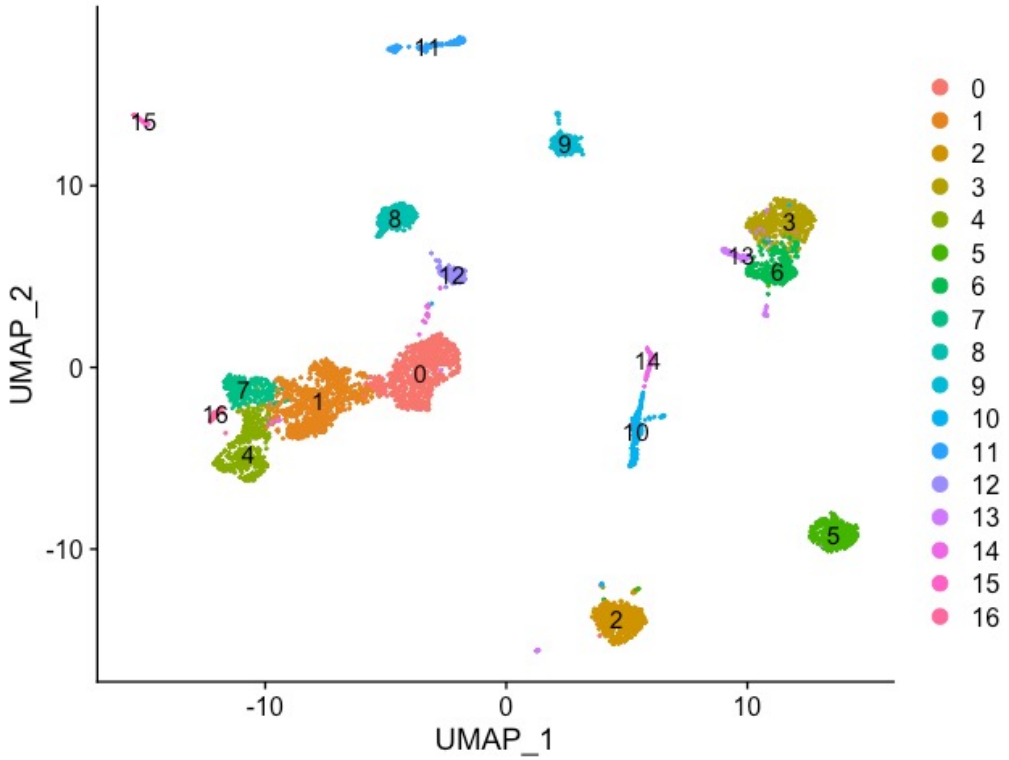
Low Quality

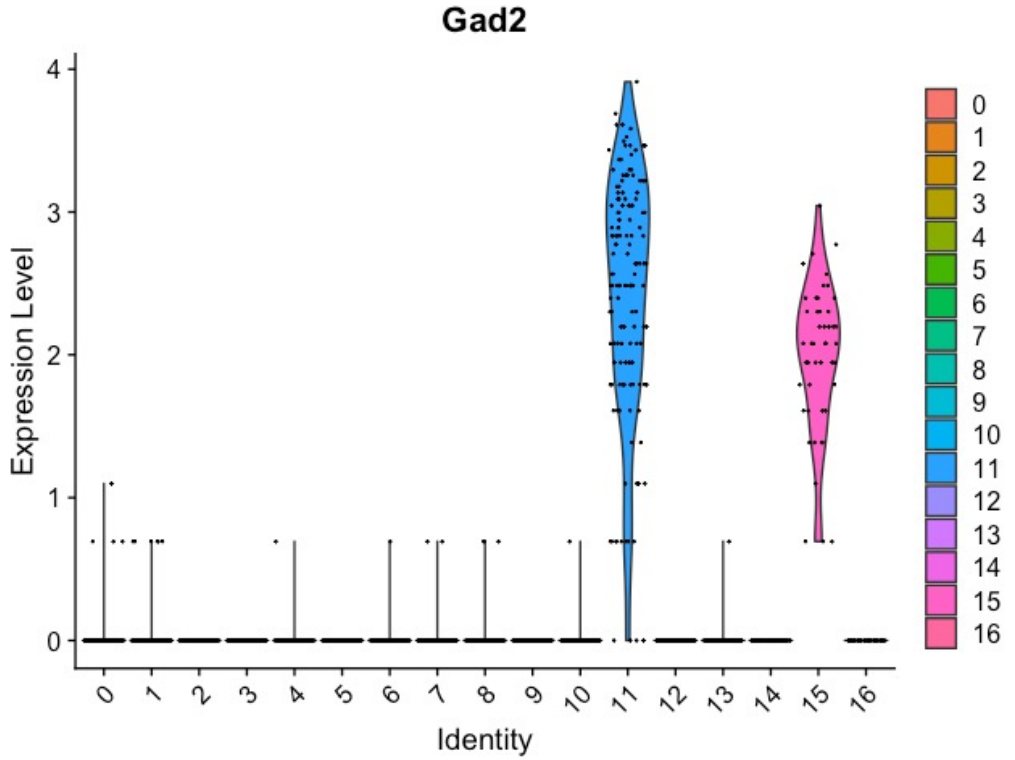
- Overlapping signatures of multiple cell types
- Very low nFeature_RNA/nCount_RNA
- Enrichment of ”mt-” mitochondrial genes
- Enrichment of ribosomal proteins
- Few specific/significant markers of identity
- Often one of the largest clusters, containing low quality cells of all cell types

Single Cell Experiment Outline



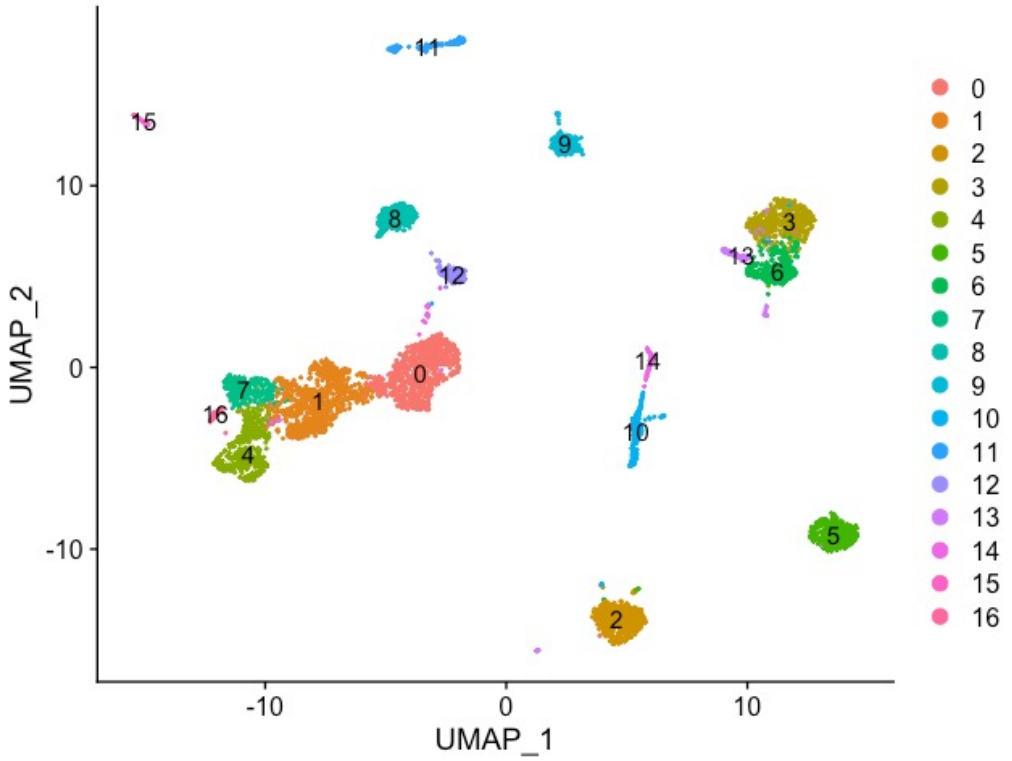
Gene Expression Visualizations – Violin Plots

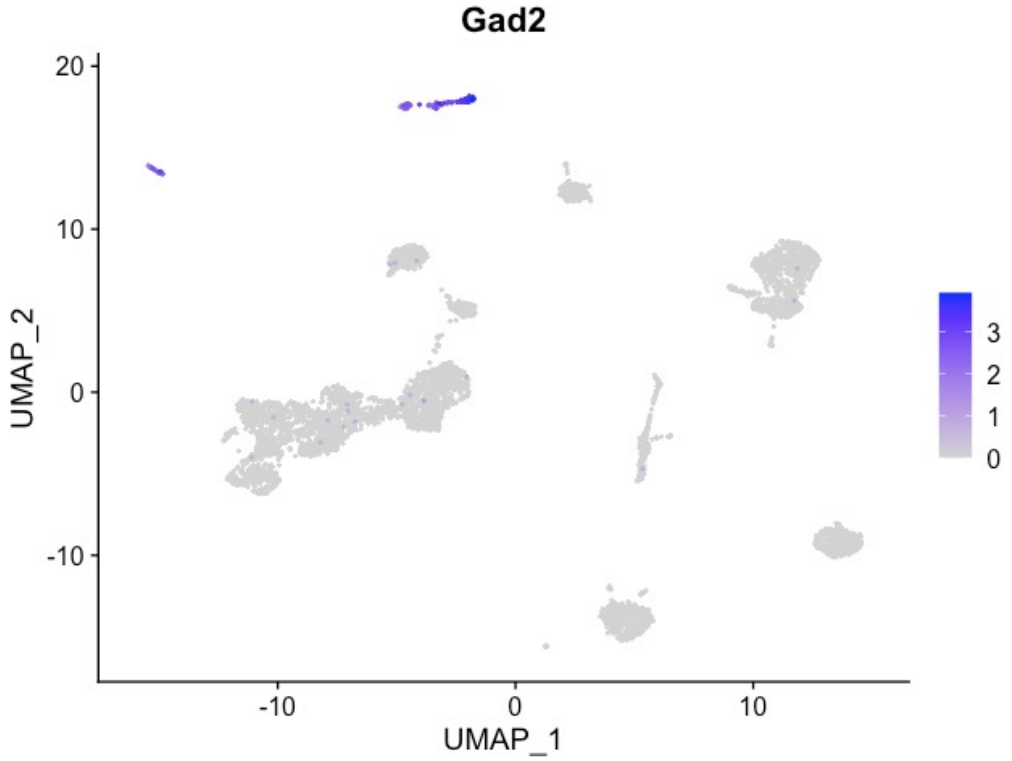




Gad2 = marker of GABAergic inhibitory neurons

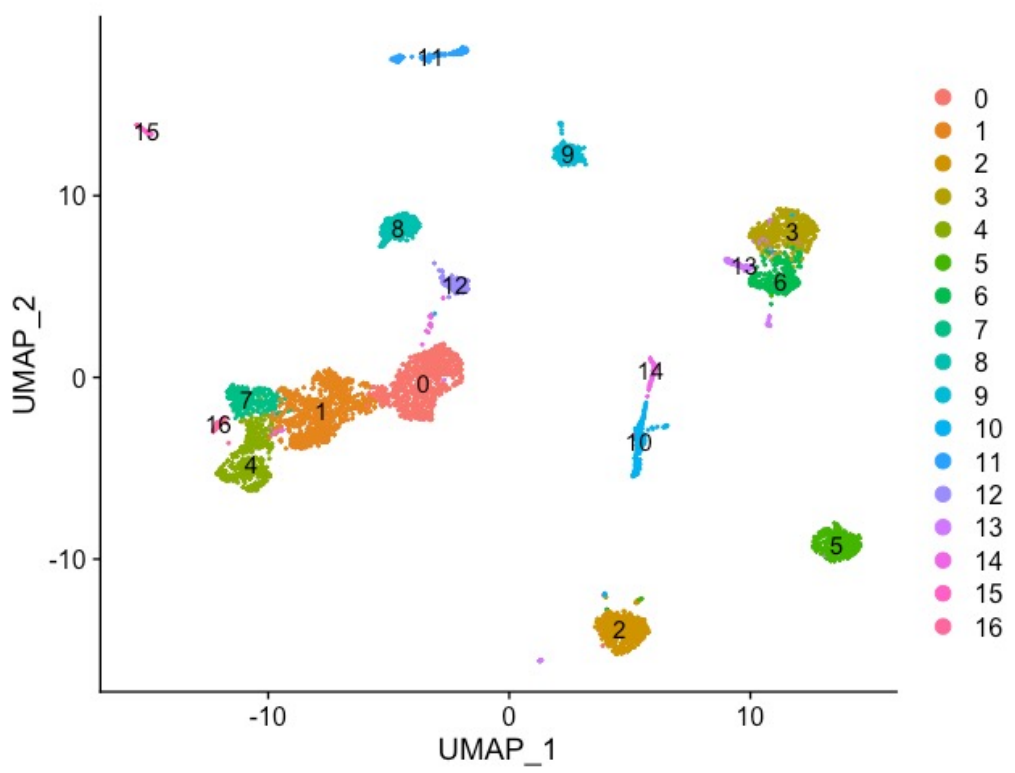
Gene Expression Visualizations – Feature Plots

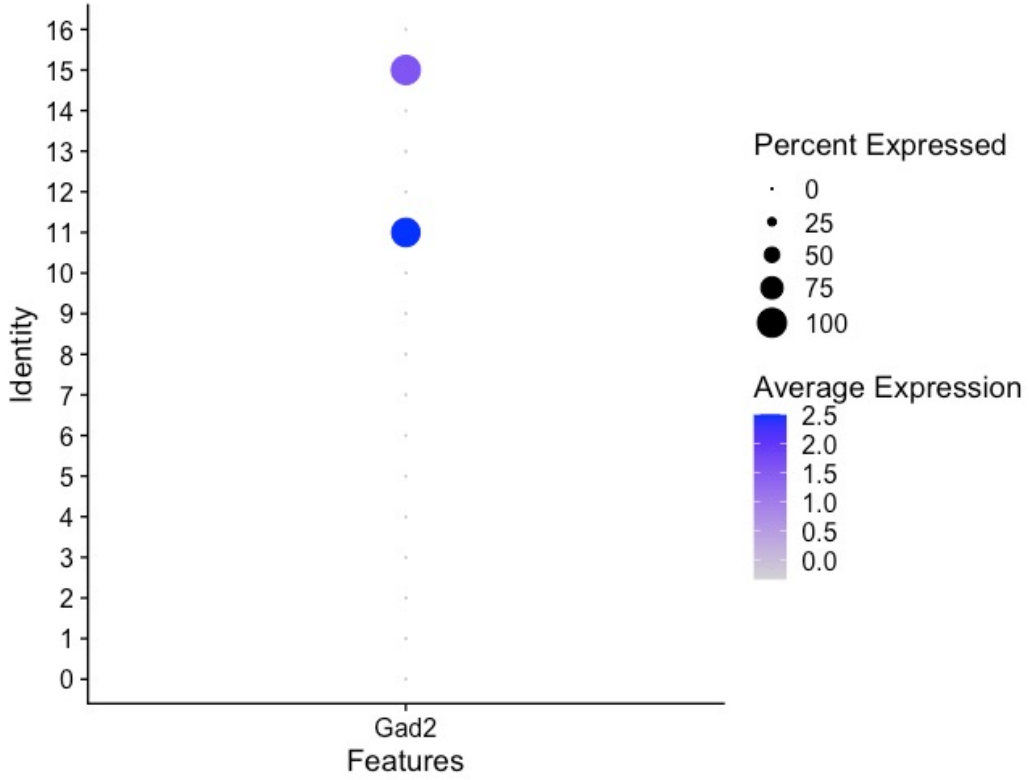




Gad2 = marker of GABAergic inhibitory neurons

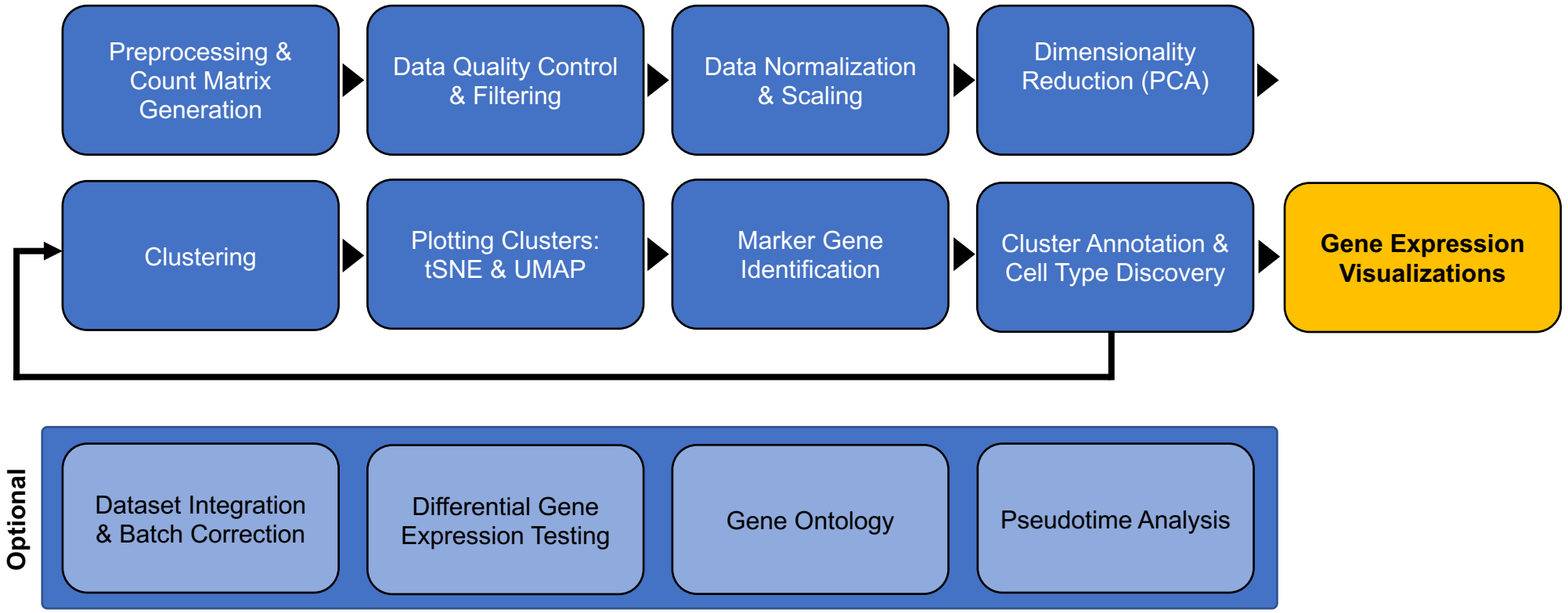
Gene Expression Visualizations – Dot Plots



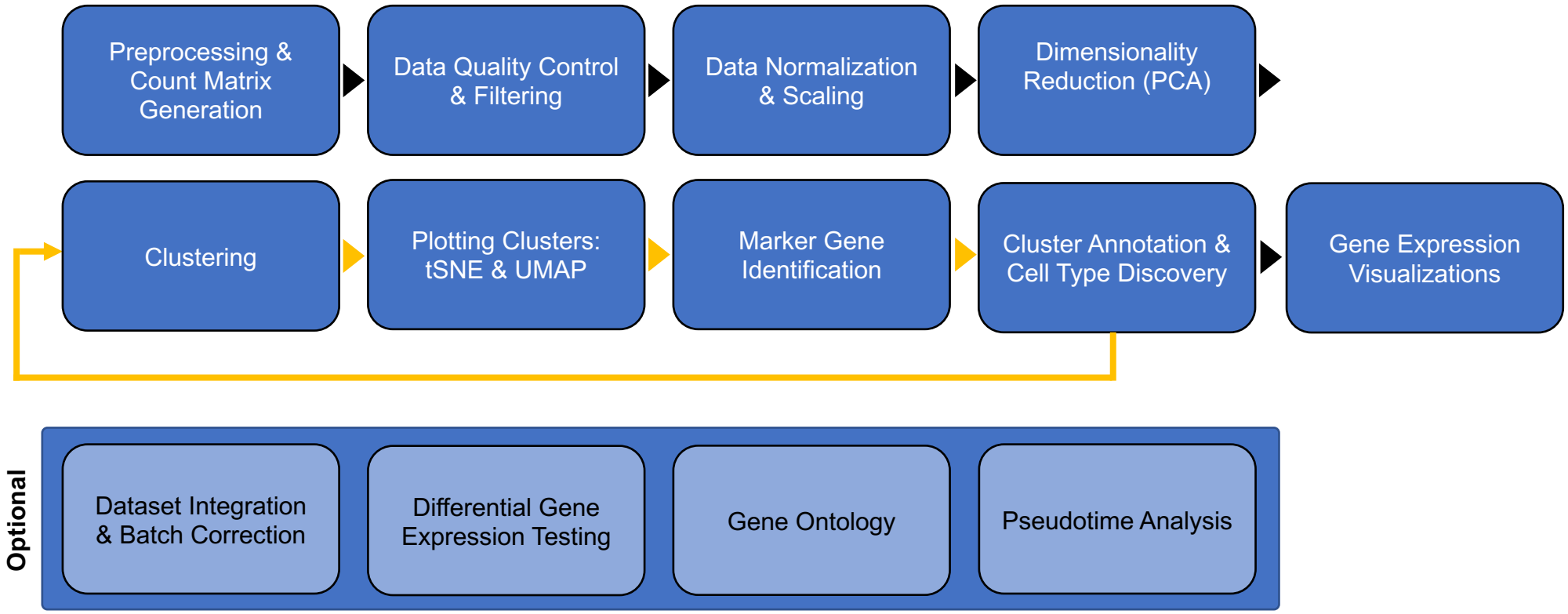


Gad2 = marker of GABAergic inhibitory neurons

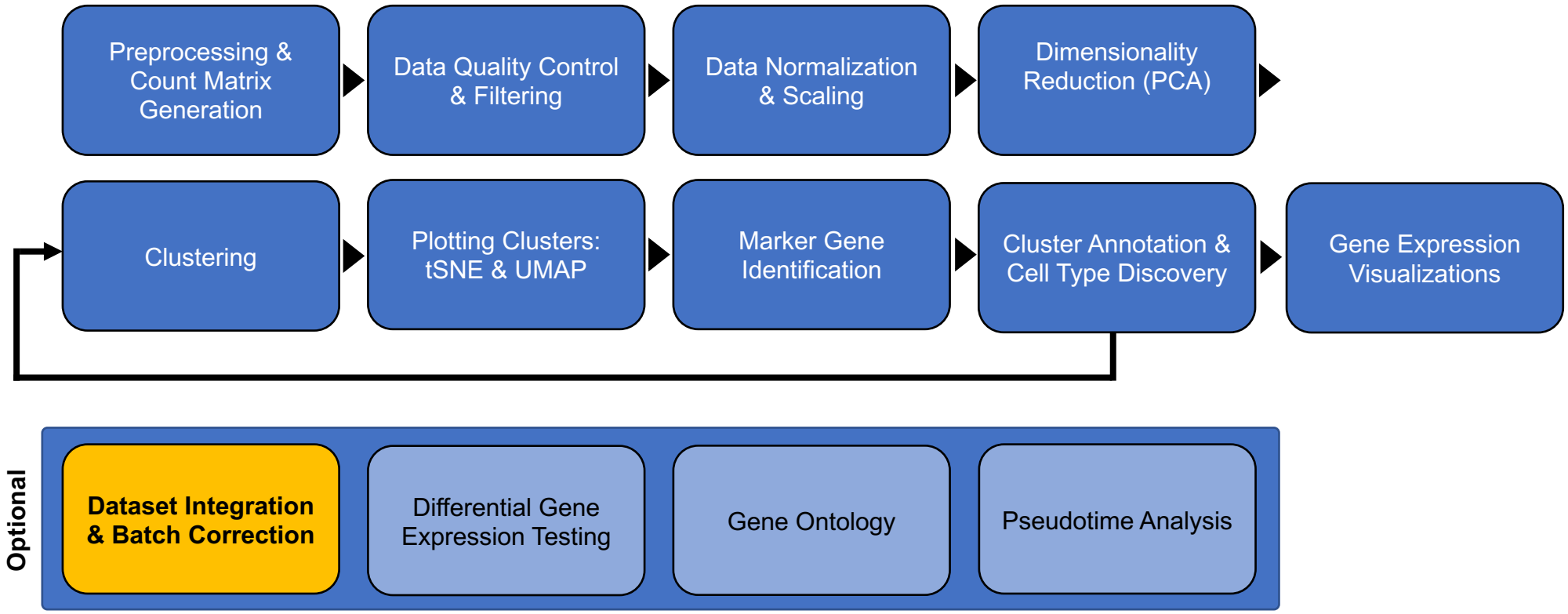
Single Cell Experiment Outline



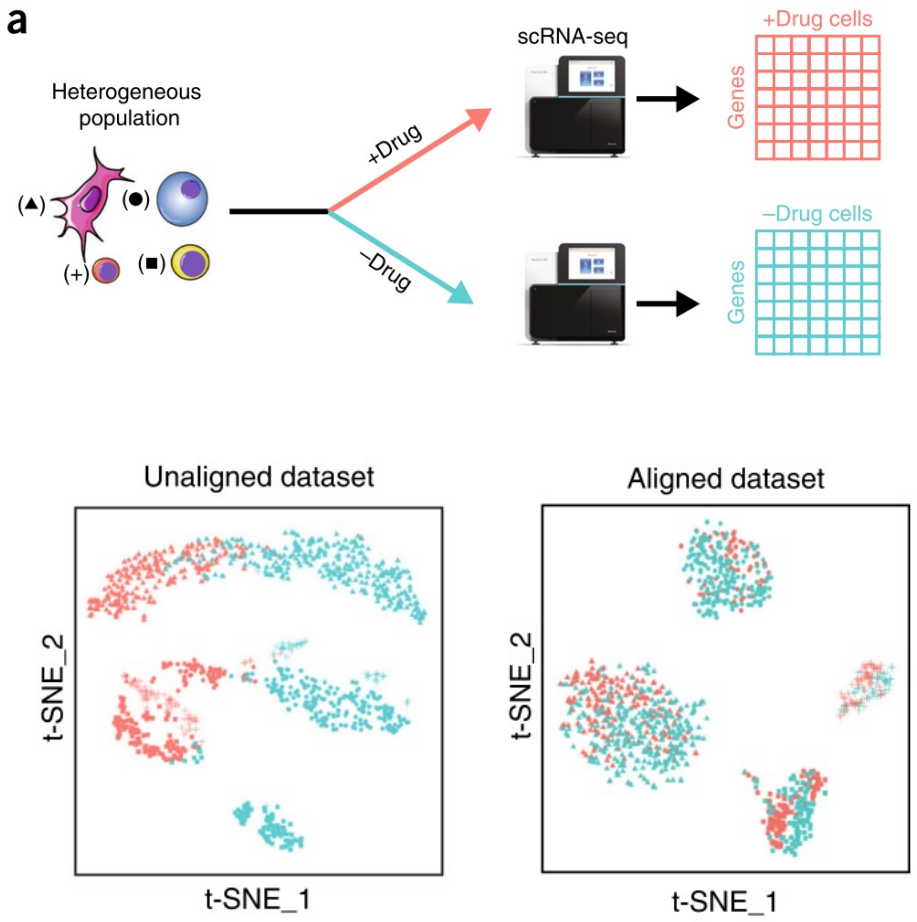
Single Cell Experiment Outline

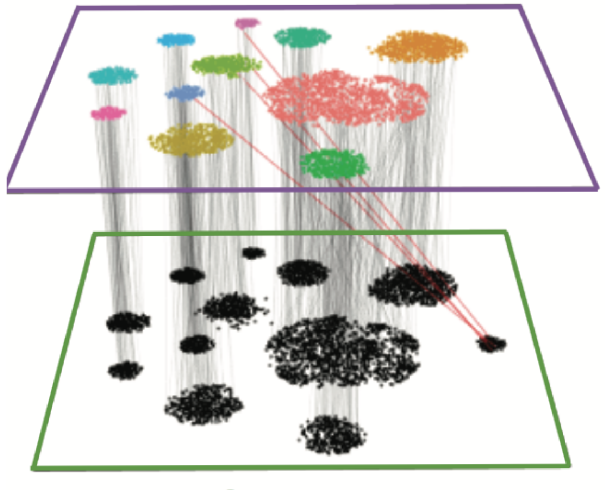


Single Cell Experiment Outline

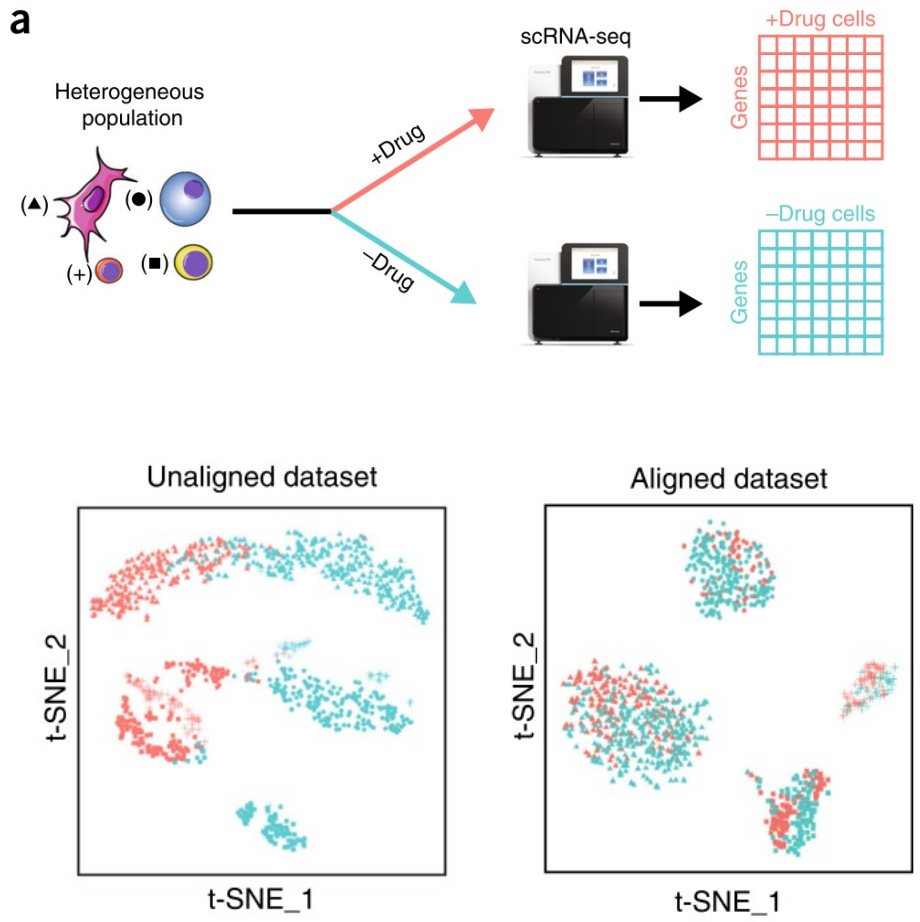


Single Cell Integration to correct for technical variation





Single Cell integration can be used to correct for technical variation

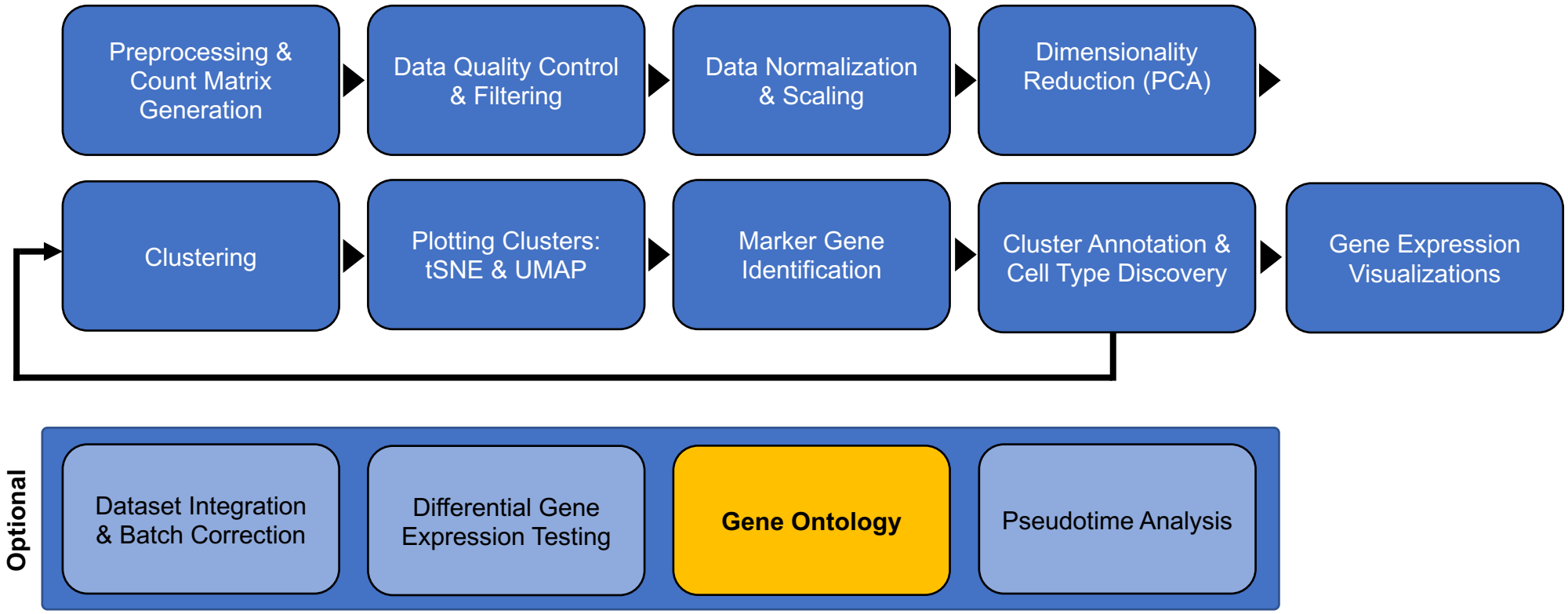


Useful Applications of Integration

- Batch effects within similar samples
- Treatment effects on samples
- Technology differences between samples

Additional reading on single cell integration:
https://www.youtube.com/watch?v=omU_-ExMYIE
https://satijalab.org/seurat/articles/integration_introduction.html
https://www.cell.com/cell/fulltext/S0092-8674(19)30559-8
https://www.nature.com/articles/nbt.4096

Single Cell Experiment Outline



Gene Ontology Analysis

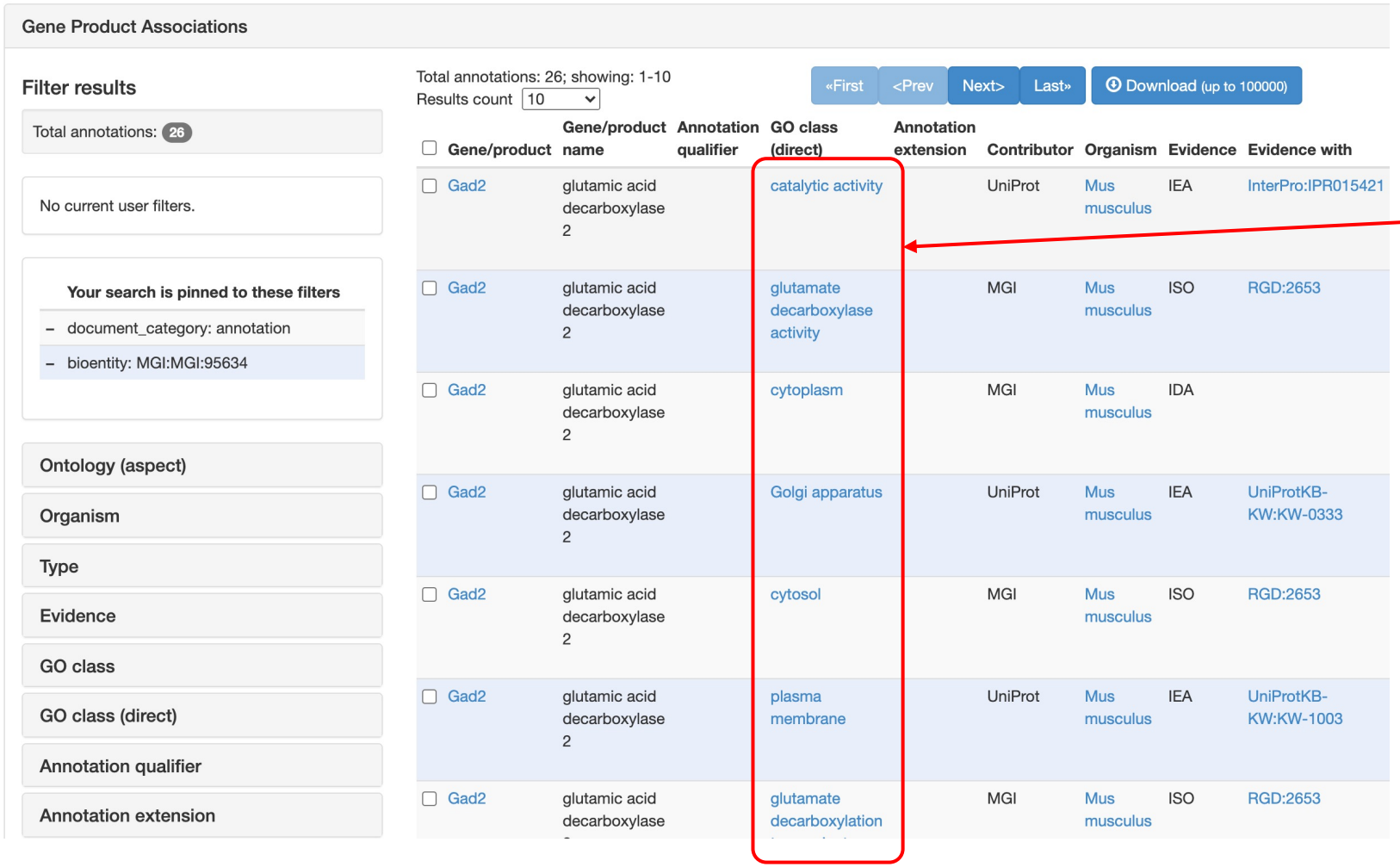
Gene Ontology

From Wikipedia, the free encyclopedia

The **Gene Ontology** (**GO**) is a major bioinformatics initiative to unify the representation of gene and gene product attributes across all species.[1] More specifically, the project aims to: 1) maintain and develop its controlled vocabulary of gene and gene product attributes; 2) annotate genes and gene products, and assimilate and disseminate annotation data; and 3) provide tools for easy access to all aspects of the data provided by the project, and to enable functional interpretation of experimental data using the GO, for example via enrichment analysis.[2][3] GO is part of a larger classification effort, the Open Biomedical Ontologies, being one of the Initial Candidate Members of the OBO Foundry.[4]

Genes have associated descriptive GO “terms”

geneontology.org



Gene Product Associations

Filter results

Total annotations: 26

No current user filters.

Your search is pinned to these filters

- document_category: annotation
- bioentity: MGI:MGI:95634

Ontology (aspect)

Organism

Type

Evidence

GO class

GO class (direct)

Annotation qualifier

Annotation extension

Total annotations: 26; showing: 1-10

Results count 10

«First <Prev Next> Last» Download (up to 100000)

Gene/product	Gene/product name	Annotation qualifier	GO class (direct)	Annotation extension	Contributor	Organism	Evidence	Evidence with
Gad2	glutamic acid decarboxylase 2		catalytic activity		UniProt	Mus musculus	IEA	InterPro:IPR015421
Gad2	glutamic acid decarboxylase 2		glutamate decarboxylase activity		MGI	Mus musculus	ISO	RGD:2653
Gad2	glutamic acid decarboxylase 2		cytoplasm		MGI	Mus musculus	IDA	
Gad2	glutamic acid decarboxylase 2		Golgi apparatus		UniProt	Mus musculus	IEA	UniProtKB-KW:KW-0333
Gad2	glutamic acid decarboxylase 2		cytosol		MGI	Mus musculus	ISO	RGD:2653
Gad2	glutamic acid decarboxylase 2		plasma membrane		UniProt	Mus musculus	IEA	UniProtKB-KW:KW-1003
Gad2	glutamic acid decarboxylase		glutamate decarboxylation		MGI	Mus musculus	ISO	RGD:2653

GO Terms

3 major classes:

- Biological Process
- Molecular Function
- Cellular Compartment

Gad2 = marker of GABAergic inhibitory neurons

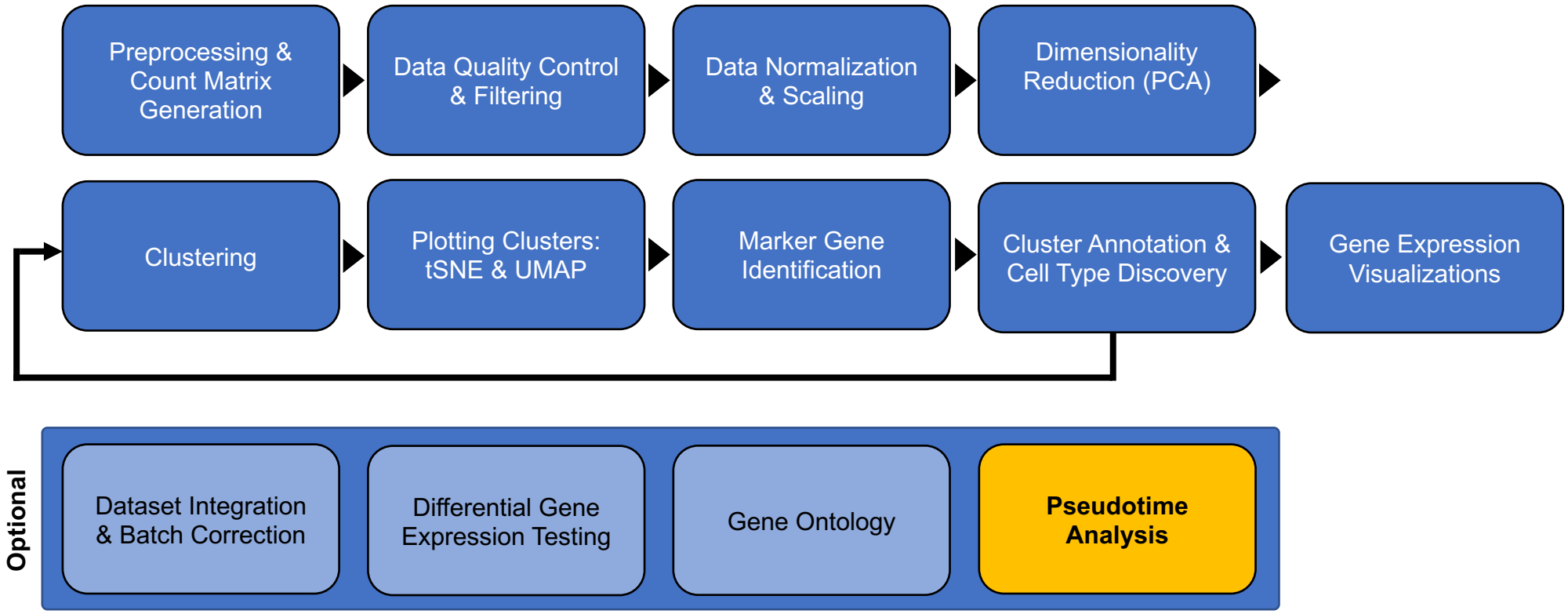
Using GO ‘enrichment’ to learn about gene lists

- Large lists of marker genes can be daunting to interpret for multiple cell types.
- Gene Ontology enrichment can identify any GO terms that are significantly enriched (over baseline) in a gene list of interest
- GO Enrichment is a tool to learn about gene lists and inform analyses/experiments, but use caution when drawing conclusion from GO results

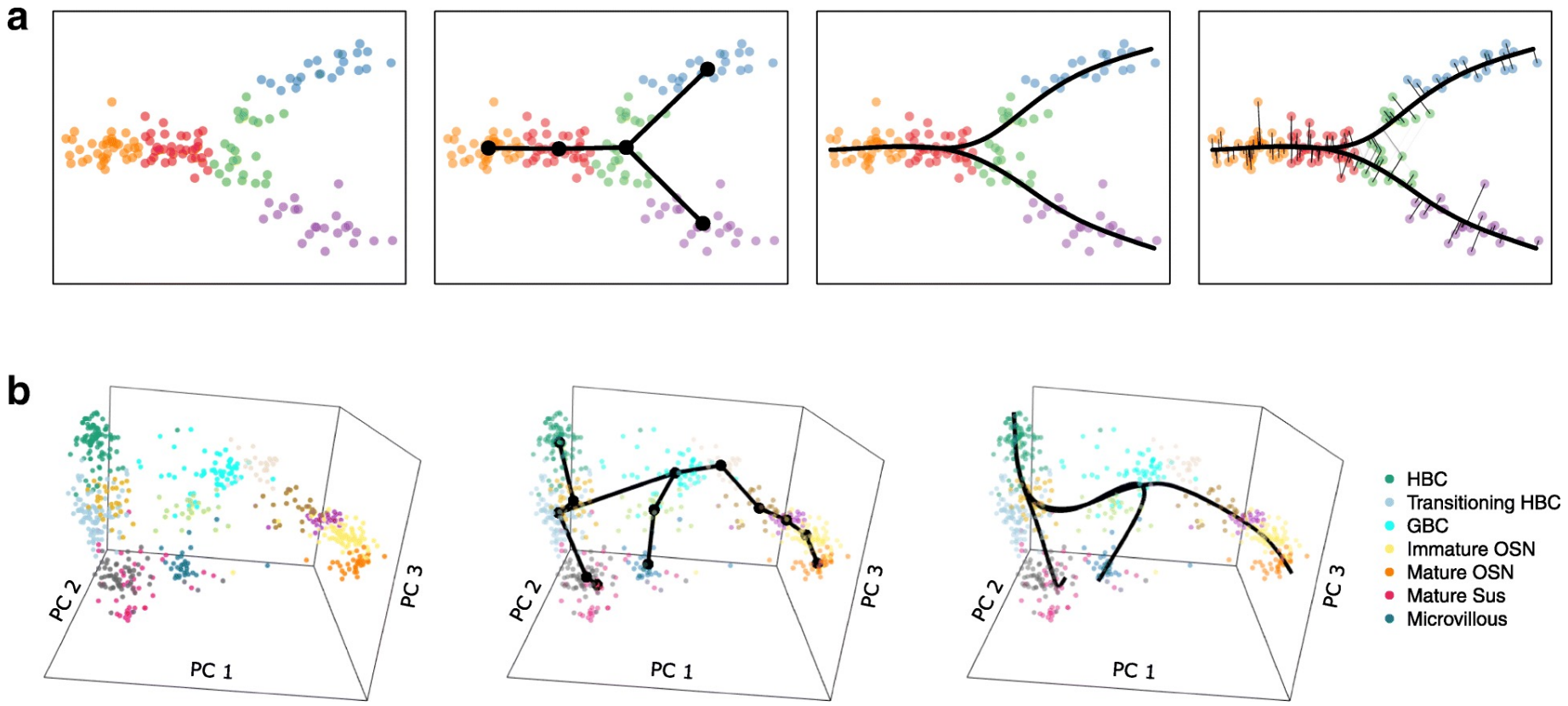
GO Enrichment Resources for single cell

- http://geneontology.org/ - online GO enrichment tool
- https://david.ncifcrf.gov/ - online GO enrichment tool
- https://bioconductor.org/packages/devel/bioc/vignettes/topGO/inst/doc/topGO.pdf - R package for Gene Set/Ontology in single cell data
- https://www.nature.com/articles/s41467-020-15298-6 - Gene Set Enrichment Analysis (GSEA) tool for single cell data

Single Cell Experiment Outline

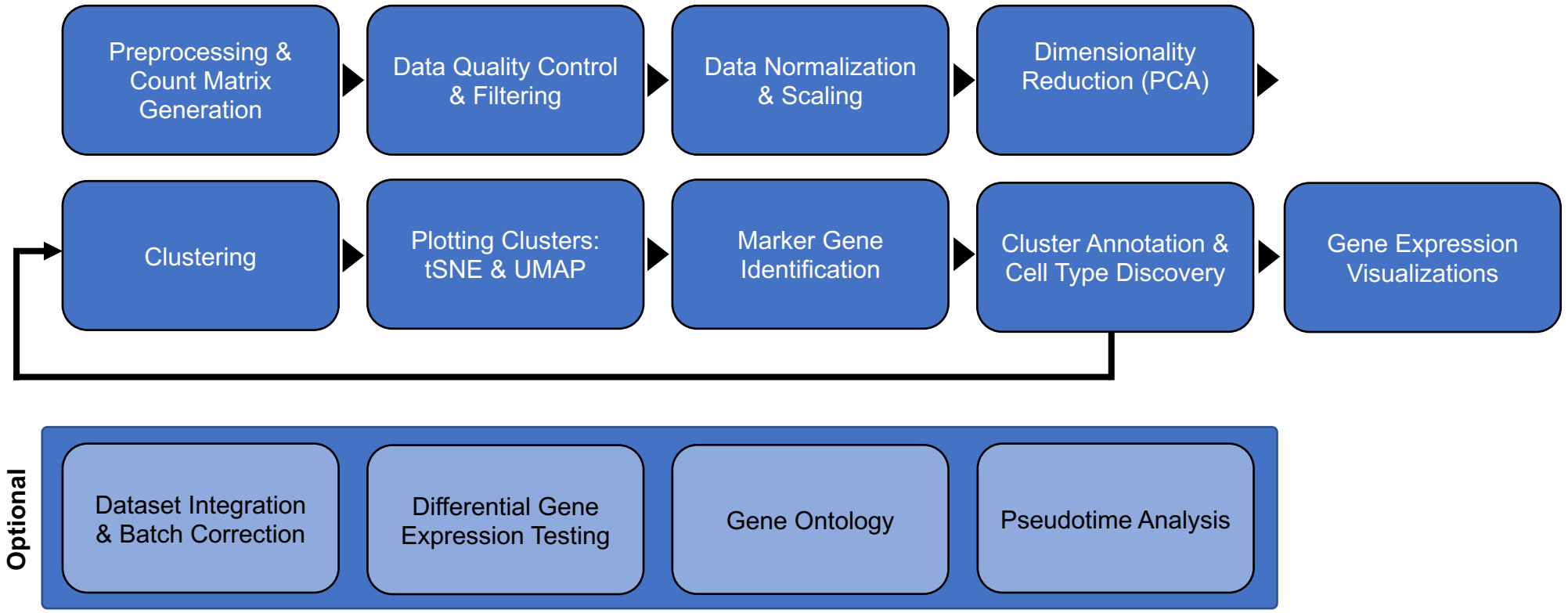


Trajectory Inference AKA “Pseudotime” Analysis



[Source: https://en.wikipedia.org/wiki/Trajectory_inference]

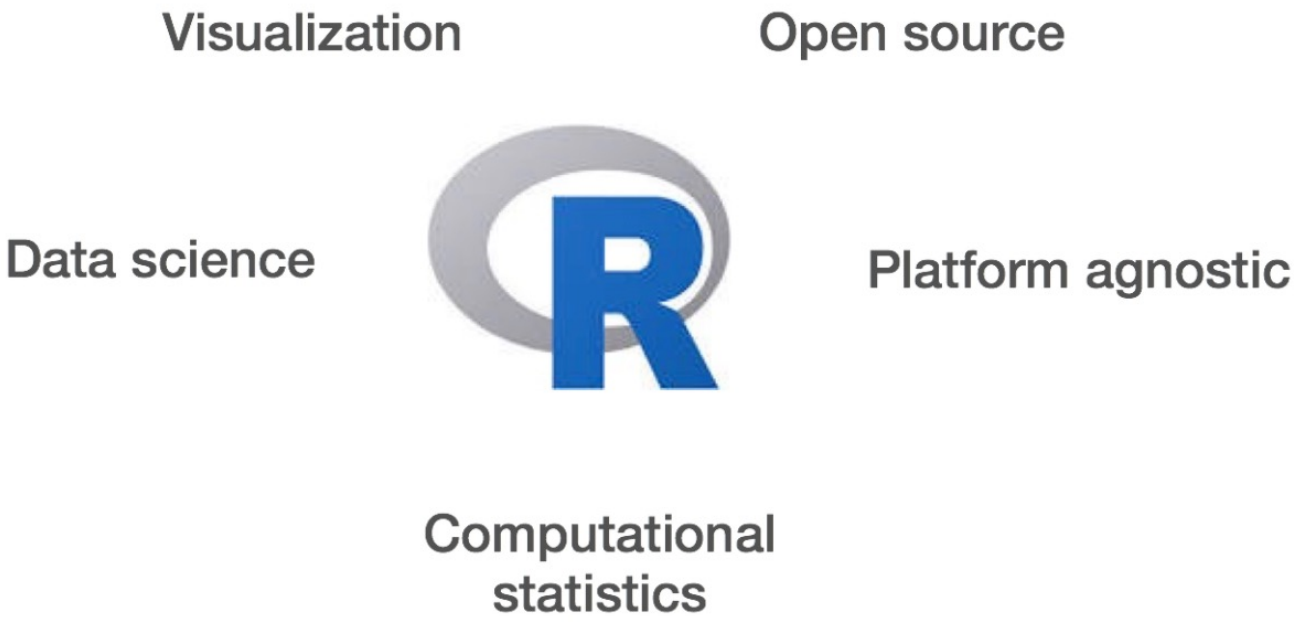
Single Cell Experiment Outline



Part II: Introduction to R

What is R?

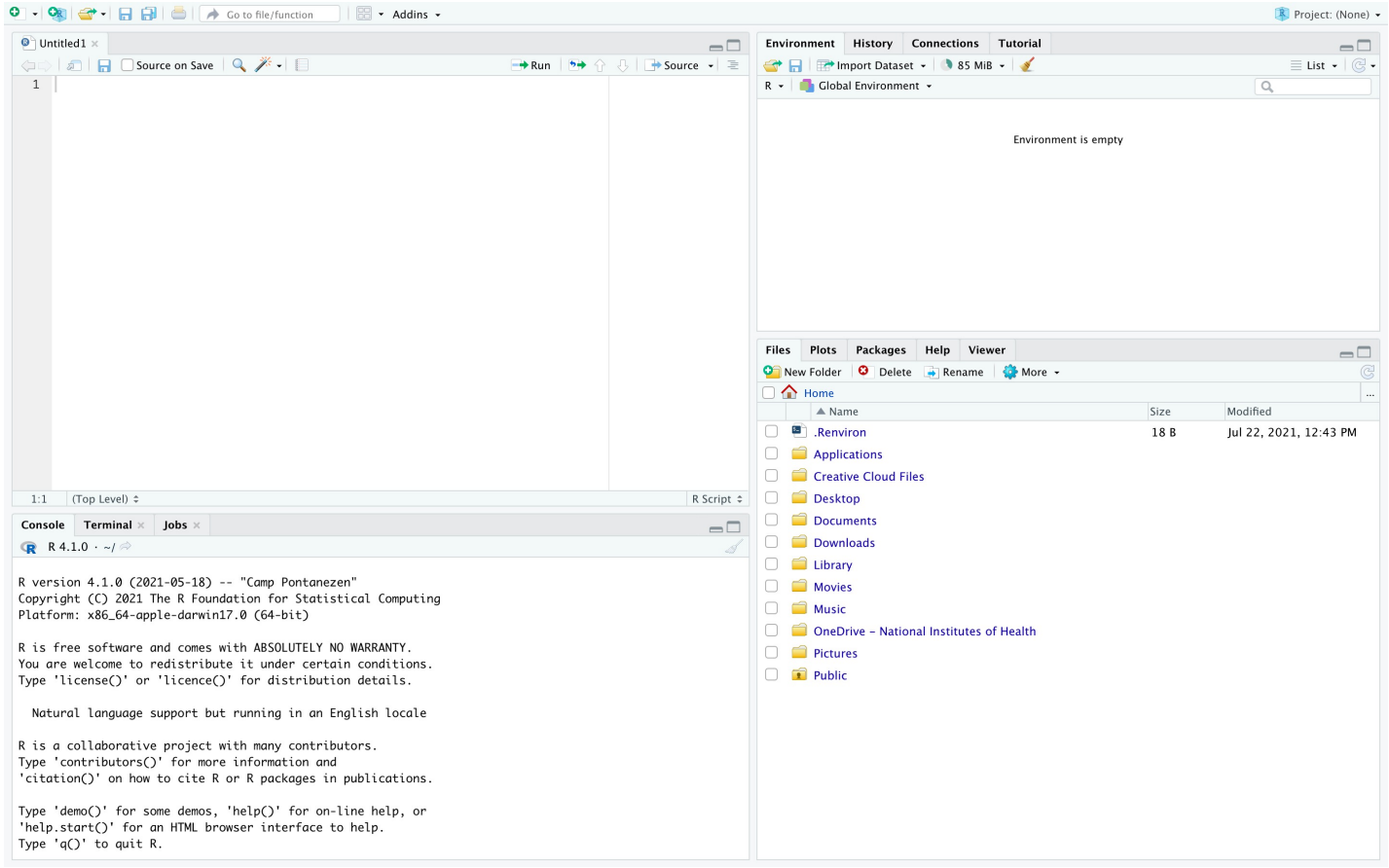
"R is a free software environment for statistical computing and graphics."



[Sources: https://hbctraining.github.io/Training-modules/IntroR/lessons/01_Intro-to-R.html https://www.r-project.org/]

R Studio

RStudio is an Integrated Development Environment for R, a programming language for statistical computing and graphics.



R Studio

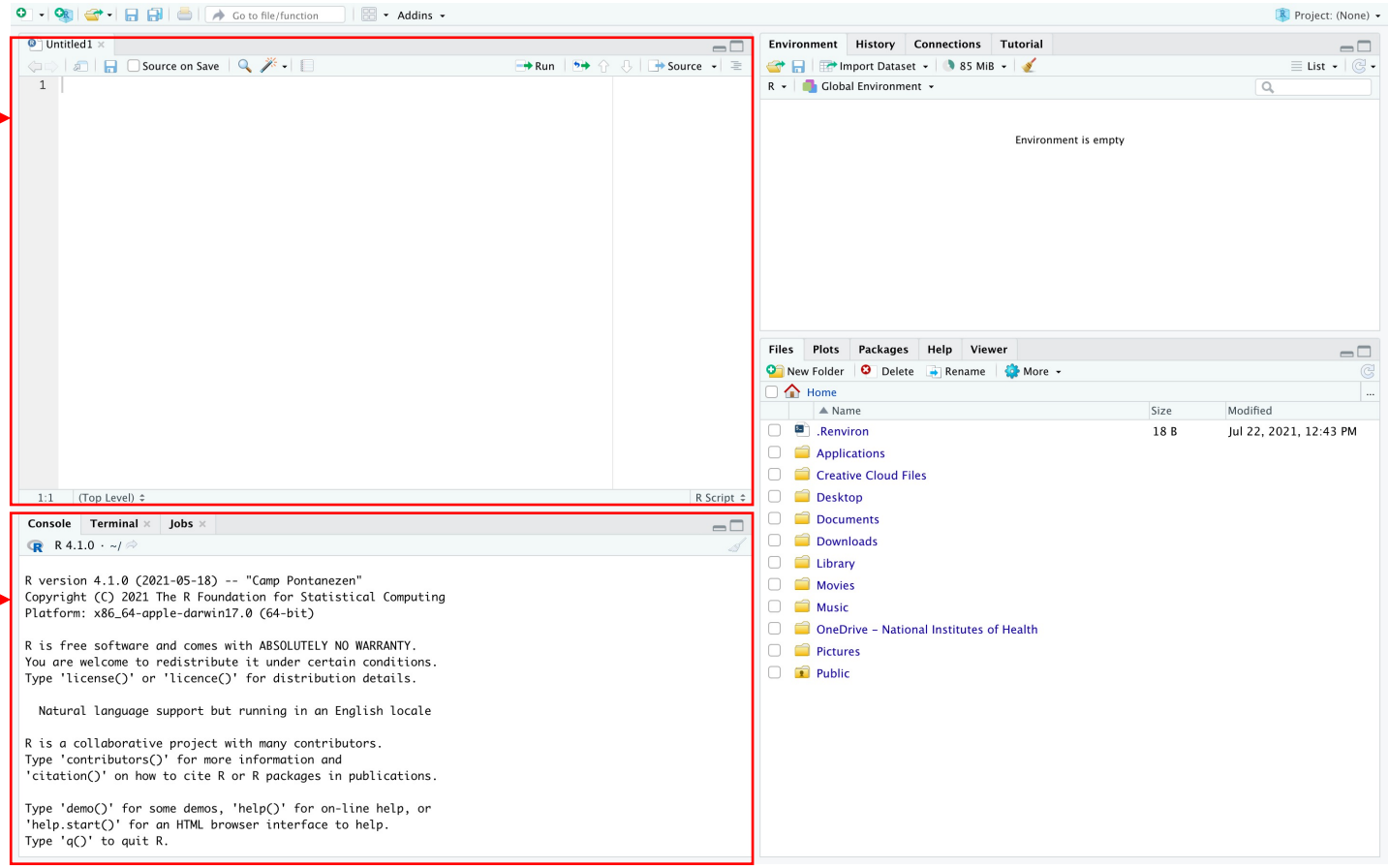
RStudio is an Integrated Development Environment for R, a programming language for statistical computing and graphics.

Script editor

Here you can type out commands and save them to a file. You can also submit the commands to run in the console.

Console

Here you can type commands and see output. *The console is all you would see if you ran R in the command line without RStudio.*

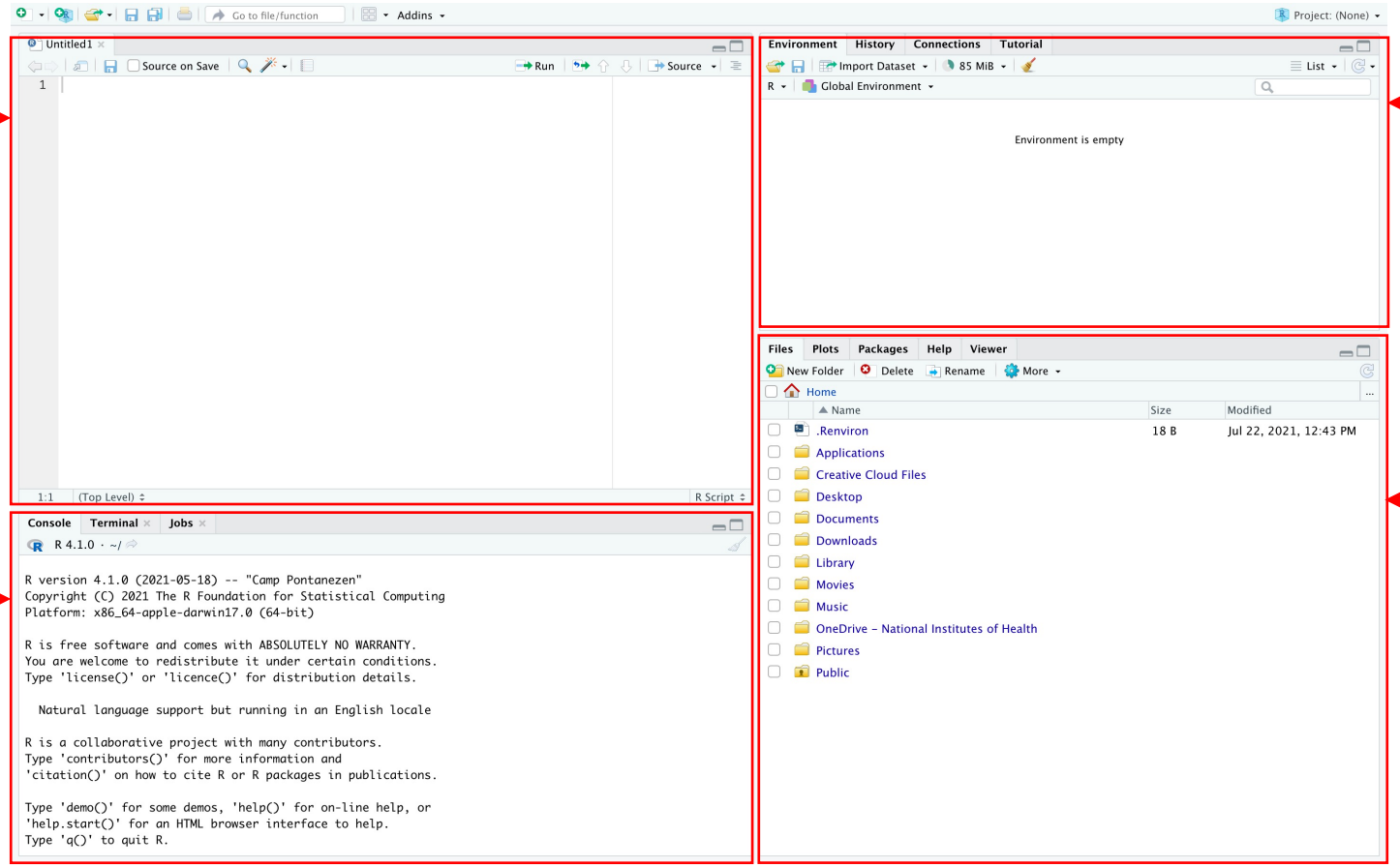


R Studio

RStudio is an Integrated Development Environment for R, a programming language for statistical computing and graphics.

Script editor

Here you can type out commands and save them to a file. You can also submit the commands to run in the console.



Console

Here you can type commands and see output. *The console is all you would see if you ran R in the command line without RStudio.*

Environment/History

This window shows all active objects and history keeps track of all commands run in console

Files/Plots/ Packages/Help

This window shows the active filesystem, any active plots, active packages, and system help.

R Syntax – Variables

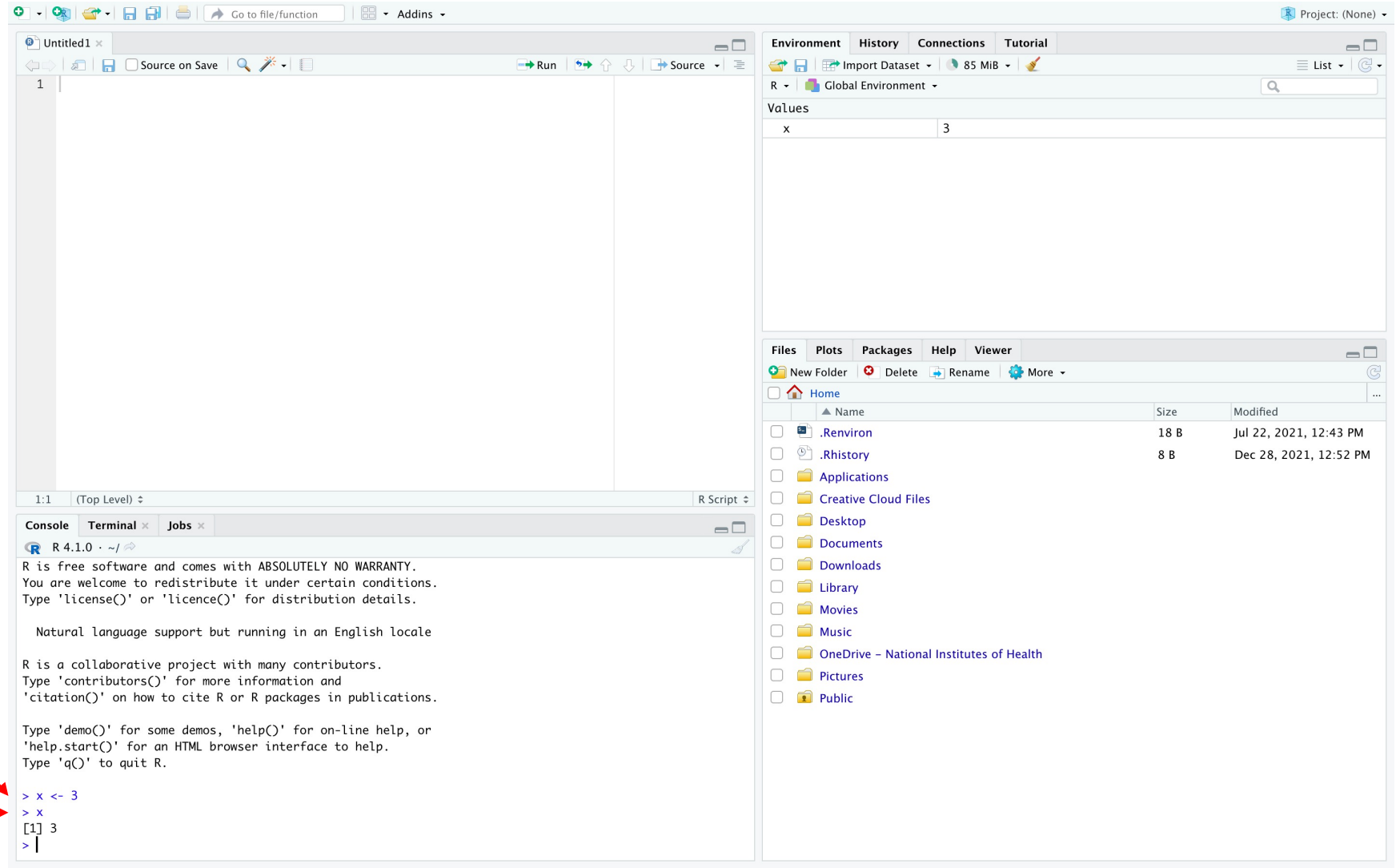
Variables

- **Variables** are used to store information to be referenced and manipulated in a computer program.

Assignment operator

- The assignment operator (<-) assigns **values on the right** to **variables on the left**.

R Syntax – Variables



Here, the value 3 is assigned to variable ‘x’

We can now access the value of ‘x’ by typing it into the console

Notice that the variable ‘x’ is now in our working environment, with a value of 3

R Syntax – Variables

Variables

- **Variables** are used to store information to be referenced and manipulated in a computer program

Variable naming conventions

- Avoid starting variable names with numbers (this can cause errors)
- Variable names should be short but informative
- Remember that R is case sensitive

What can variables store?

- In R, variables can store single values *or* complex data structures

R Syntax – Comments

Comments are used to explain R code and make it more readable.

They can also be used to prevent execution when testing alternative code.

Comments start with a # and end at the end of the line

When executing the code, R will ignore anything that starts with #

Comment your code liberally!

```
> print("Hello World") #this is a comment
[1] "Hello World"
```

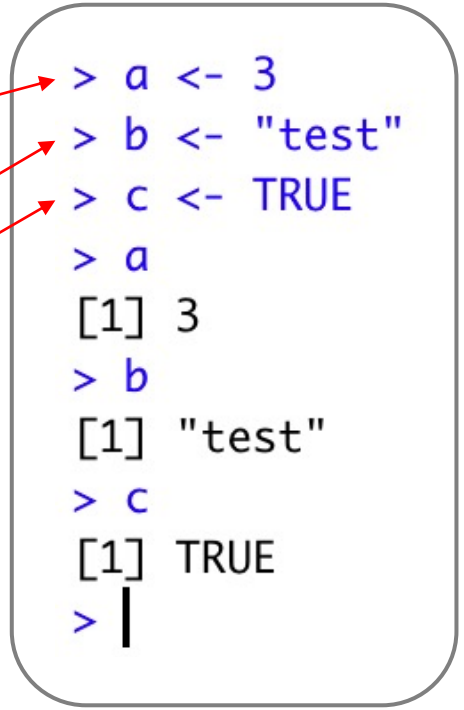
R Data Types and Structures – Simple Data Types

Simple Data Types

"numeric" – stores numerical values

"character" – stores text value, requires quotes ("")

"logical" – stores TRUE and FALSE (boolean)



R Data Types and Structures – Vectors

"vector" – stores a collection of values *of the same type*

A vector of numbers

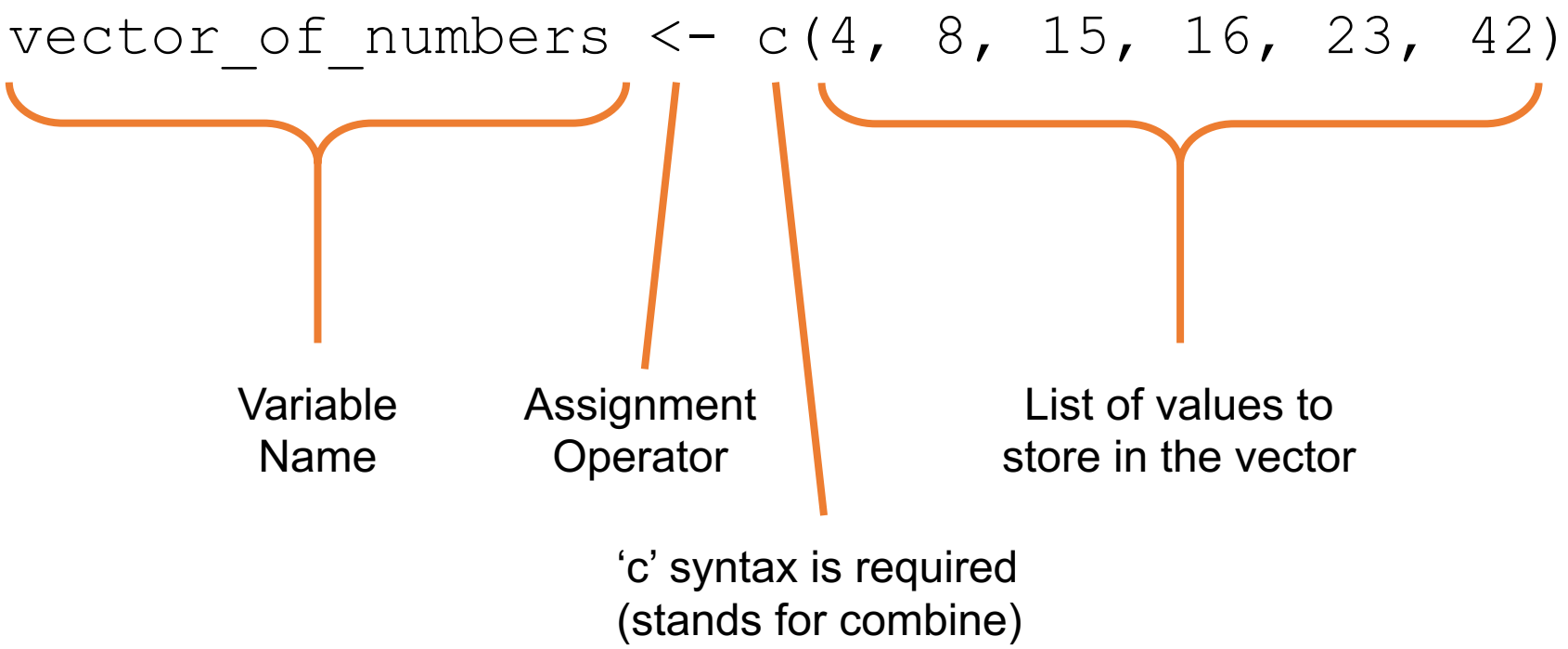
A vector of characters

A vector of logical values

[Source: https://hbctraining.github.io/Training-modules/IntroR/lessons/02_syntax_and_data_structures.html]

R Data Types and Structures – Vector Syntax

"vector" – stores a collection of values *of the same type*

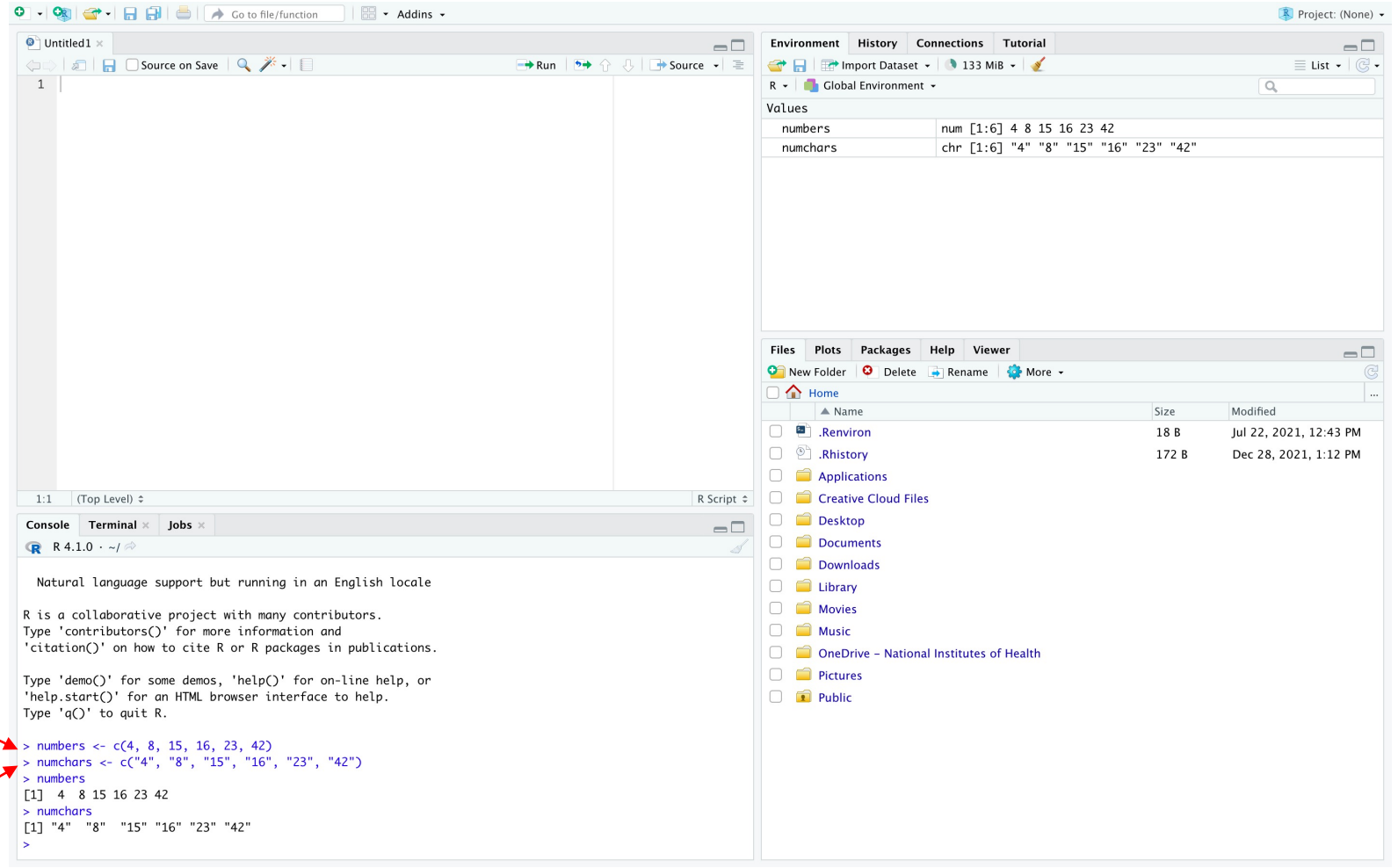


R Data Types and Structures - Vectors

"vector" – stores a collection of values *of the same type*

A vector of numbers

A vector of characters



R Data Types and Structures - Matrix

"matrix" – a collection of vectors of **same length and identical datatype**.

Vectors can be combined as columns in the matrix or by row, to create a 2-dimensional structure.

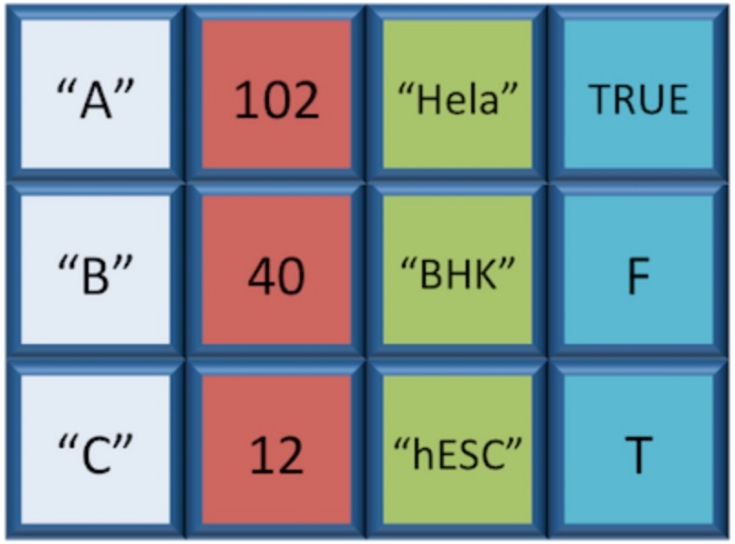
A matrix of numbers

[Source: https://hbctraining.github.io/Training-modules/IntroR/lessons/02_syntax_and_data_structures.html]

R Data Types and Structures – Data Frame (data.frame)

"data.frame" – the *de facto* data structure for most tabular data.

A data.frame is similar to a matrix in that it's a collection of vectors of of the **same length** and each vector represents a column. However, in a dataframe **each vector can be of a different data type** (e.g., characters, integers, factors).



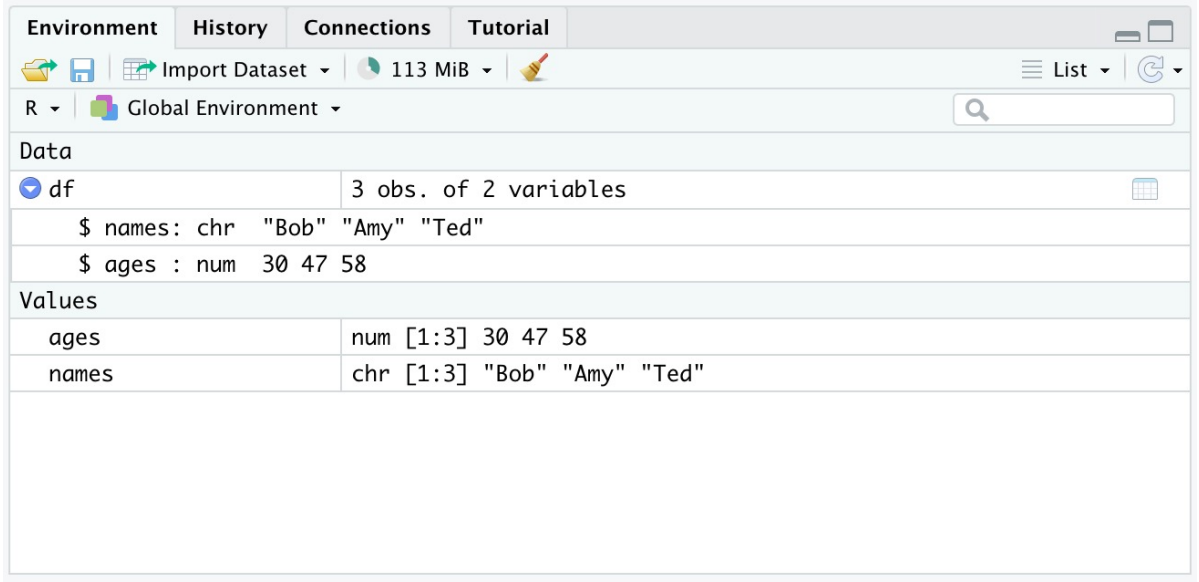
[Source: https://hbctraining.github.io/Training-modules/IntroR/lessons/02_syntax_and_data_structures.html]

R Data Types and Structures – Creating a Data Frame (data.frame)

Console

```
> names <- c("Bob", "Amy", "Ted")
> ages <- c(30, 47, 58)
> df <- data.frame(names, ages)
> df
  names ages
1   Bob   30
2   Amy   47
3   Ted   58
```

Environment



R Data Types and Structures – Data Frame Interaction

Functions for data inspection

- `str()`: compact display of data contents (env.)
- `class()`: data type (e.g. character, numeric, etc.) of vectors and data structure of dataframes, matrices, and lists.
- `summary()`: detailed display, including descriptive statistics, frequencies
- `head()`: will print the beginning entries for the variable
- `tail()`: will print the end entries for the variable
- `dim()`: returns dimensions of the dataset
- `nrow()`: returns the number of rows in the dataset
- `ncol()`: returns the number of columns in the dataset
- `rownames()`: returns the row names in the dataset
- `colnames()`: returns the column names in the dataset

[Source: https://hbctraining.github.io/Training-modules/IntroR/lessons/04_data-wrangling.htl]

R Data Types and Structures – Data Frame Interaction

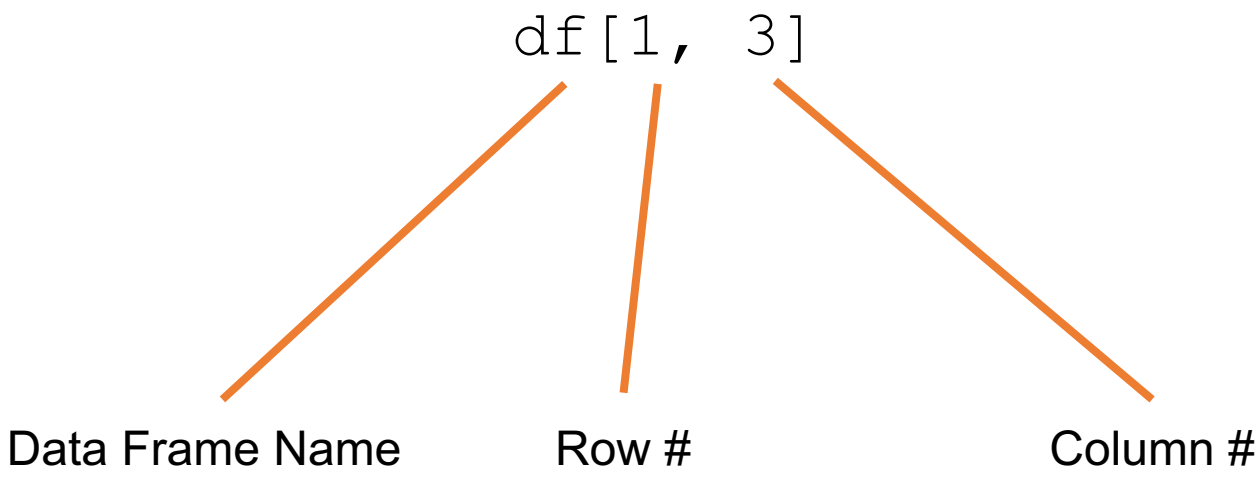
Data Frame ‘df’

	names	ages
1	Bob	30
2	Amy	47
3	Ted	58

```
> str(df)
'data.frame':   3 obs. of  2 variables:
 $ names: chr  "Bob" "Amy" "Ted"
 $ ages : num  30 47 58
> head(df, 2)
  names ages
1   Bob   30
2   Amy   47
> dim(df)
[1] 3 2
> nrow(df)
[1] 3
> ncol(df)
[1] 2
> rownames(df)
[1] "1" "2" "3"
> colnames(df)
[1] "names" "ages"
```

R Data Types and Structures – Data Frame Interaction

Accessing slots within a Data Frame



[Source: https://hbctraining.github.io/Training-modules/IntroR/lessons/04_data-wrangling.htl]

R Data Types and Structures – Data Frame Interaction

Data Frame 'df'

	names	ages
1	Bob	30
2	Amy	47
3	Ted	58

```
> df[1, 1]          # element from the first row in the first column of the data frame
[1] "Bob"
> df[3, 2]          # element from the third row in the 2nd column
[1] 58
> df[3, ]           # vector containing all elements in the 3rd row
  names ages
3   Ted   58
> df[, 1]           # vector containing all elements in the 1st column
[1] "Bob" "Amy" "Ted"
> df[ , 1:2]        # dataframe containing first two columns
  names ages
1   Bob   30
2   Amy   47
3   Ted   58
> df[c(1,3), ]      # dataframe containing first and third rows
  names ages
1   Bob   30
3   Ted   58
> df[4, 17]         # element that does not exist
NULL
> df[1:2, ]         # dataframe containing the first two rows
  names ages
1   Bob   30
2   Amy   47
> df[1:2 , "ages"] # elements of the ages column corresponding to the first two names
[1] 30 47
```

R Data Types and Structures – List

"list" – a data structure that can hold any number of any types of other data structures.

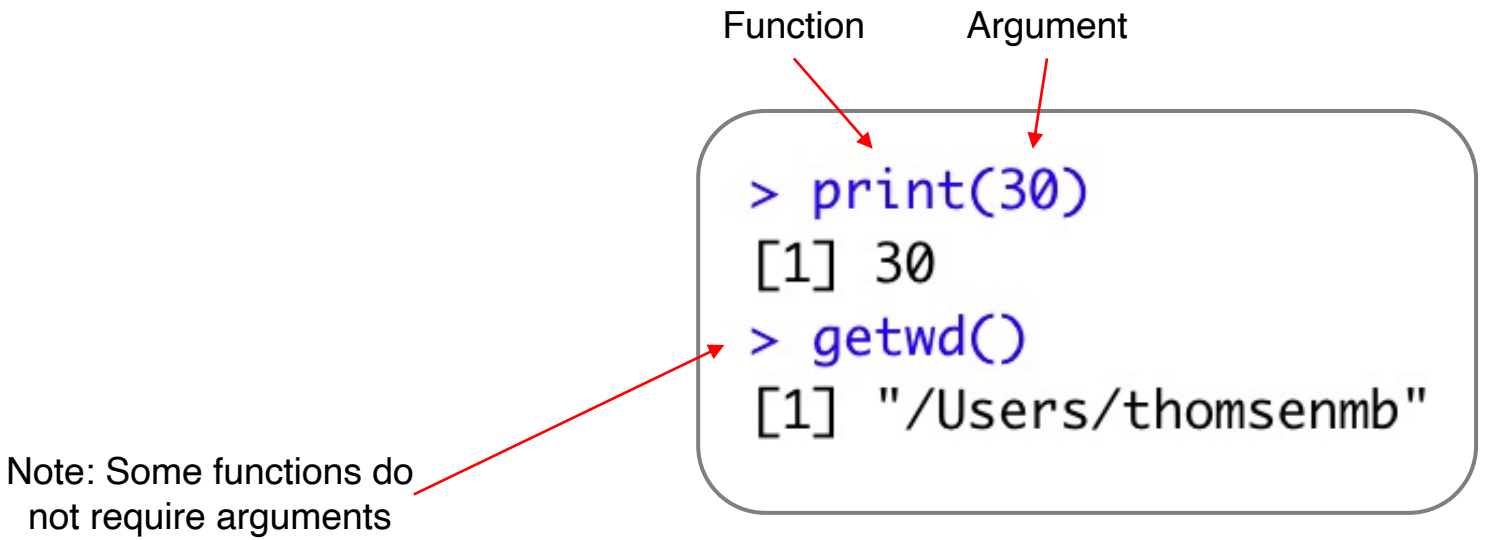
A list of diverse data structures

[Source: https://hbctraining.github.io/Training-modules/IntroR/lessons/02_syntax_and_data_structures.html]

R Functions and Arguments

Functions are **"self contained" modules of code that accomplish a specific task**.

Functions usually take in some sort of data structure (value, vector, dataframe etc.), process it, and return a result.

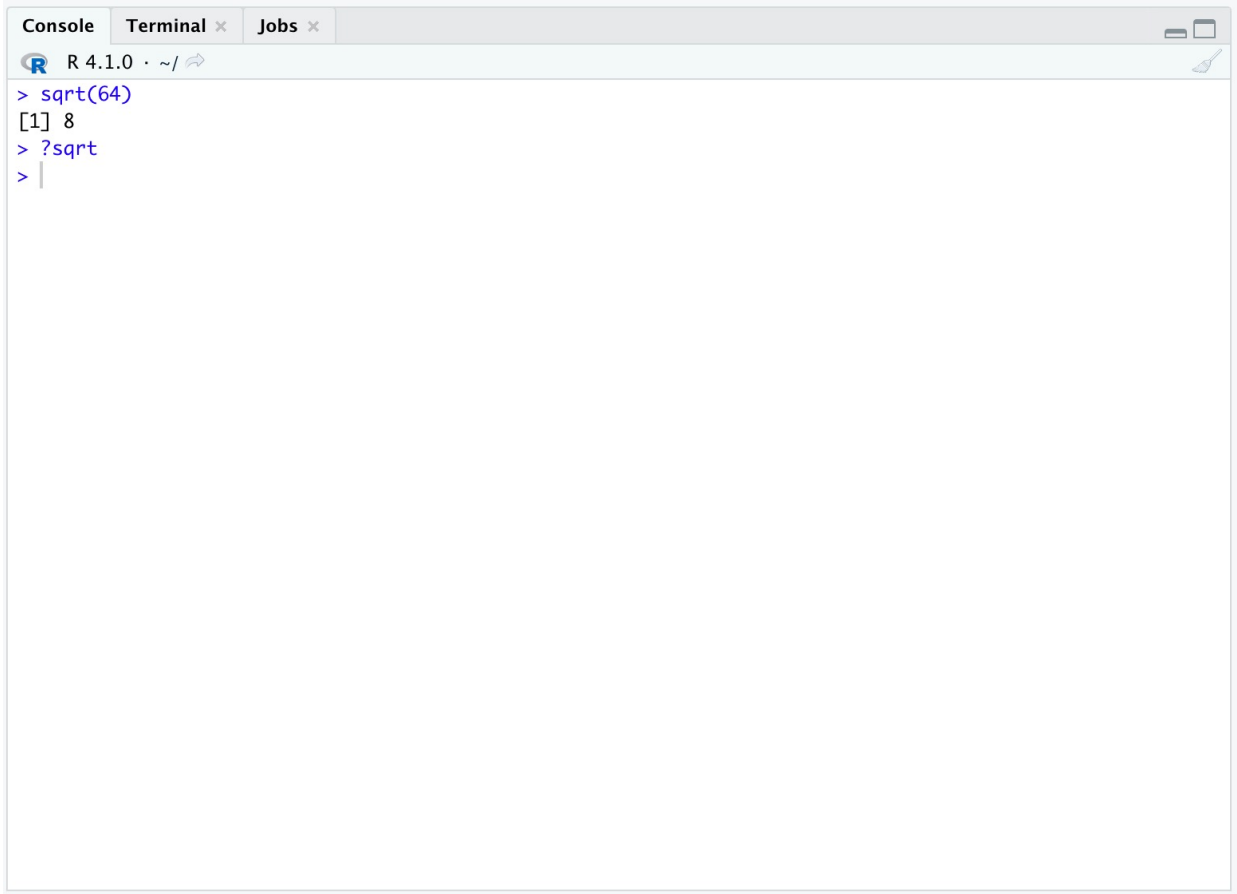


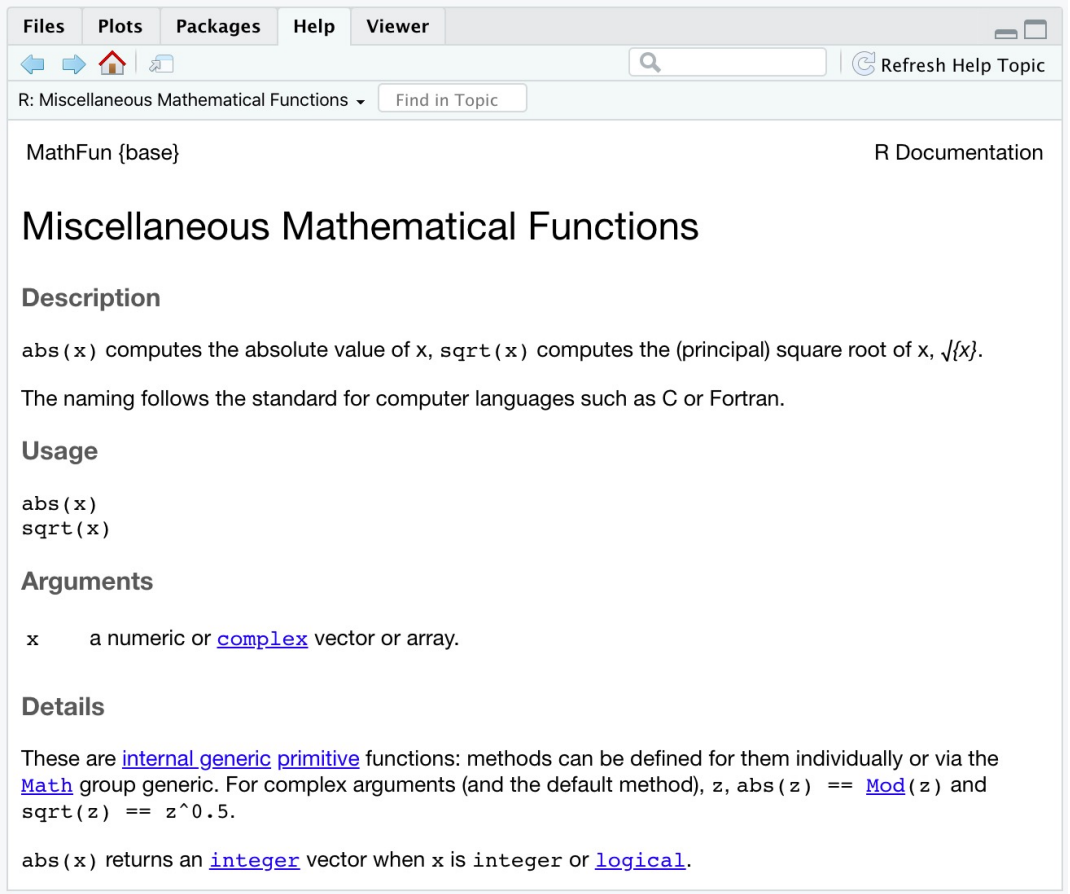
Arguments can include:

- Variables
- Options
- Values

[Source: https://hbctraining.github.io/Training-modules/IntroR/lessons/03_functions-and-arguments.html]

R Functions and Arguments – Getting Help





R Packages and Libraries

Packages are collections of R functions, data, and compiled code in a well-defined format, created to add specific functionality. There are 10,000+ user contributed packages and growing.

There are a set of **standard (or base) packages** which are considered part of the R source code and automatically available as part of your R installation. Base packages contain the **basic functions** that allow R to work, and enable standard statistical and graphical functions on datasets; for example, all of the functions that we have been using so far in our examples.

The directories in R where the packages are stored are called the **libraries**. The terms *package* and *library* are sometimes used synonomously and there has been discussion amongst the community to resolve this. It is somewhat counter-intuitive to *load a package* using the `library()` function and so you can see how confusion can arise.

You can check what packages are loaded in your R session by typing into the console:

```
sessionInfo()
```

[Source: https://hbctraining.github.io/Training-modules/IntroR/lessons/03_functions-and-arguments.html]

R Packages and Libraries

sessionInfo() run in Console

Console | Terminal | Jobs

R 4.1.0 · ~/

```
> sessionInfo()
R version 4.1.0 (2021-05-18)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Catalina 10.15.7

Matrix products: default
BLAS:   /System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/vecLib.framework/Versions/
A/libBLAS.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
[1] compiler_4.1.0 tools_4.1.0
>
```

'Packages' tab

Files | Plots | Packages | Help | Viewer

Install | Update

	Name	Description	Version
System Library			
☑	base	The R Base Package	4.1.0
☐	boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-28
☐	class	Functions for Classification	7.3-19
☐	cluster	"Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.	2.1.2
☐	codetools	Code Analysis Tools for R	0.2-18
☐	compiler	The R Compiler Package	4.1.0
☑	datasets	The R Datasets Package	4.1.0
☐	foreign	Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ...	0.8-81
☑	graphics	The R Graphics Package	4.1.0
☑	grDevices	The R Graphics Devices and Support for Colours and Fonts	4.1.0
☐	grid	The Grid Graphics Package	4.1.0
☐	KernSmooth	Functions for Kernel Smoothing Supporting Wand & Jones (1995)	2.23-20
☐	lattice	Trellis Graphics for R	0.20-44
☐	MASS	Support Functions and Datasets for Venables and Ripley's MASS	7.3-54
☐	Matrix	Sparse and Dense Matrix Classes and Methods	1.3-3
☑	methods	Formal Methods and Classes	4.1.0
☐	mgcv	Mixed GAM Computation Vehicle with Automatic Smoothness Estimation	1.8-35
☐	nlme	Linear and Nonlinear Mixed Effects Models	3.1-152
☐	nnet	Feed-Forward Neural Networks and Multinomial Log-	7.3-16

R Packages and Libraries – Installation and Loading

The standard package installation function: `install.packages()`

```
> install.packages('ggplot2')
Installing package into ‘/Users/thomsenmb/Library/R/x86_64/4.1/library’
(as ‘lib’ is unspecified)
also installing the dependencies ‘colorspace’, ‘cli’, ‘crayon’, ‘utf8’, ‘farver’, ‘labeling’, ‘lifecycl
e’, ‘munsell’, ‘R6’, ‘RColorBrewer’, ‘viridisLite’, ‘ellipsis’, ‘fansi’, ‘magrittr’, ‘pillar’, ‘pkgconfi
g’, ‘vctrs’, ‘digest’, ‘glue’, ‘gtable’, ‘isoband’, ‘rlang’, ‘scales’, ‘tibble’, ‘withr’

trying URL 'https://cran.rstudio.com/bin/macosx/contrib/4.1/colorspace_2.0-2.tgz'
Content type 'application/x-gzip' length 2618801 bytes (2.5 MB)
==================================================
downloaded 2.5 MB

trying URL 'https://cran.rstudio.com/bin/macosx/contrib/4.1/cli_3.1.0.tgz'
Content type 'application/x-gzip' length 1123462 bytes (1.1 MB)
==================================================
downloaded 1.1 MB
```

(cropped console output)

```
The downloaded binary packages are in
        /var/folders/56/hrfpsyqx6571fwbnxr3f77b0mt31tv/T//RtmpUAEoGn/downloaded_packages
>
```

In general, before installing new packages check the package documentation for best practices and any potential prerequisites to install. Not all packages and versions can be installed using the “install.packages()” function.

R Packages and Libraries – Installation and Loading

Once packages are installed, they must be loaded into the session in order to access.

The standard package loading function: `library()`

ggplot2 installation completed

Trying to use installed packages before loading will result in errors

Load the ggplot2 package

Now ggplot2 commands can be used

```
The downloaded binary packages are in
        /var/folders/56/hrfpsyqx6571fwbnxr3f77b0mt31tv/T//RtmpUAEoGn/downloaded_packages

> ggplot()
Error in ggplot() : could not find function "ggplot"
> library(ggplot2)
> ggplot()
```

Troubleshooting Resources

https://github.com/satijalab/seurat/issues - Seurat github development page, place to ask questions to the authors directly, and has a complete archive of existing questions/updates

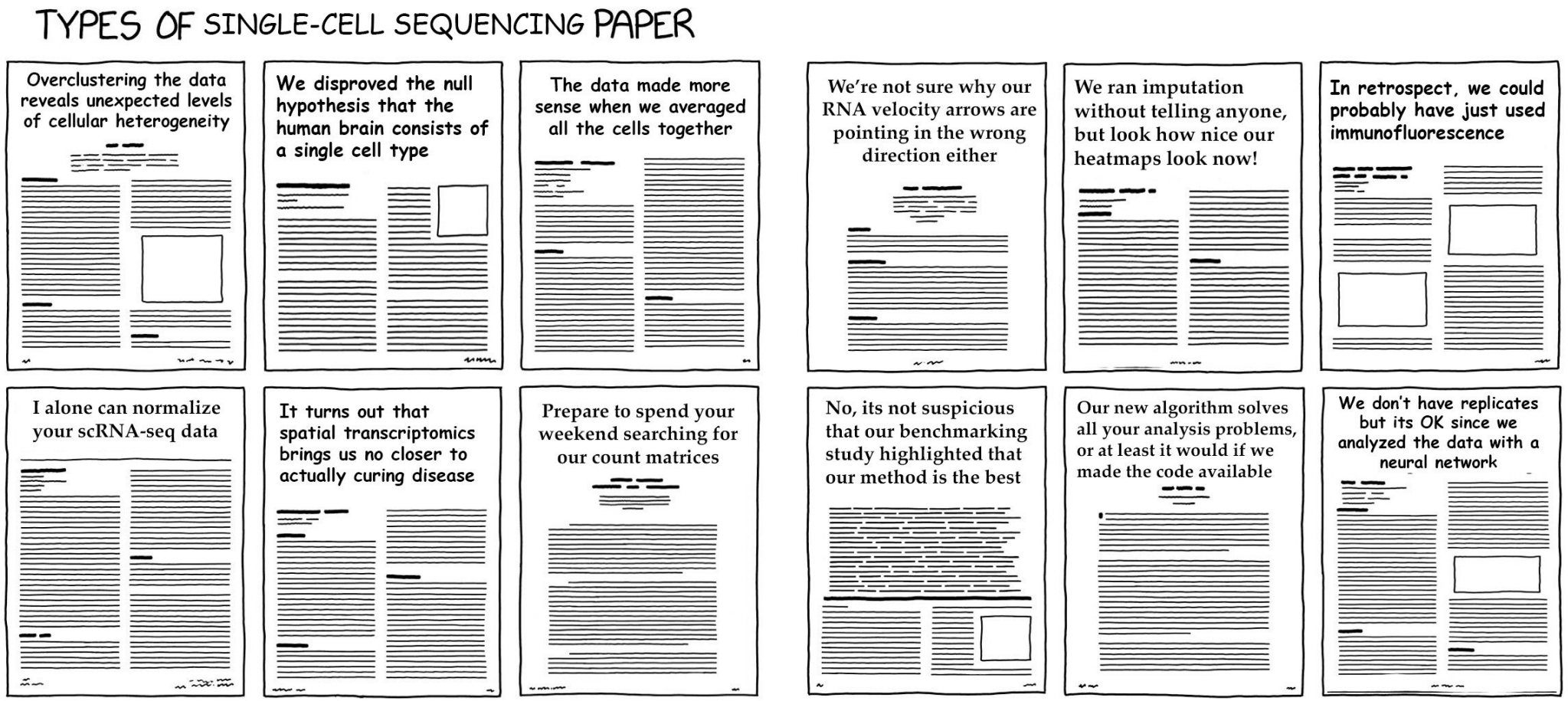
https://www.biostars.org/

http://seqanswers.com/

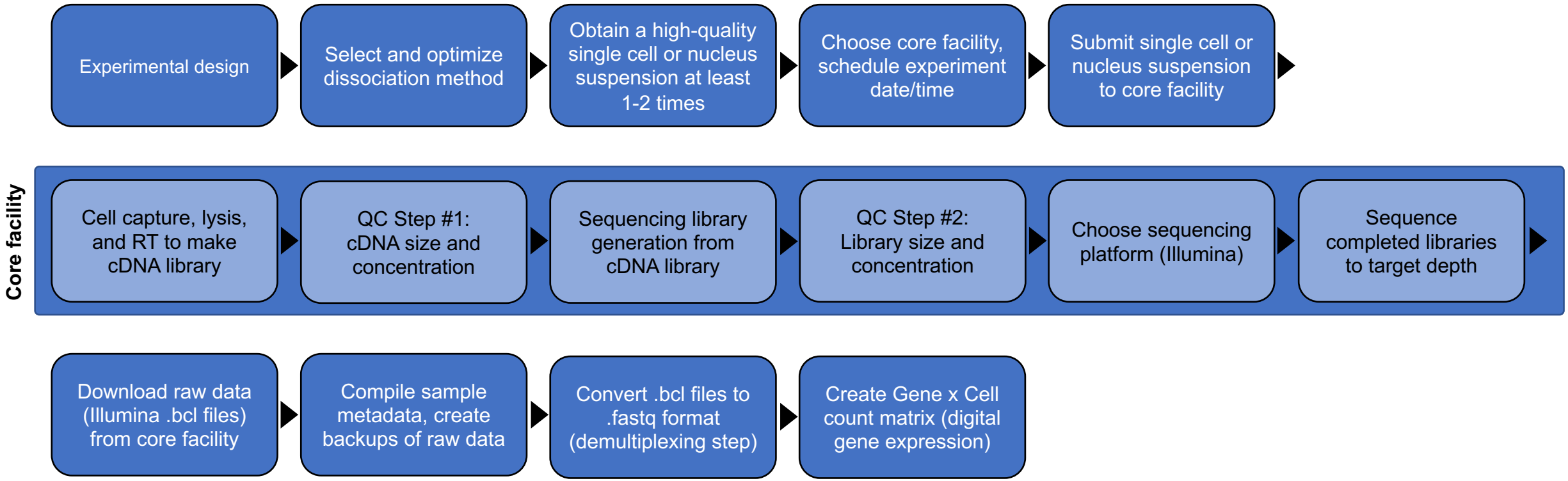
https://stackexchange.com/

Part III: Data Analysis and Visualization Workshop!

Don’t be these people



Single Cell Experiment Outline



Single Cell Experiment Outline