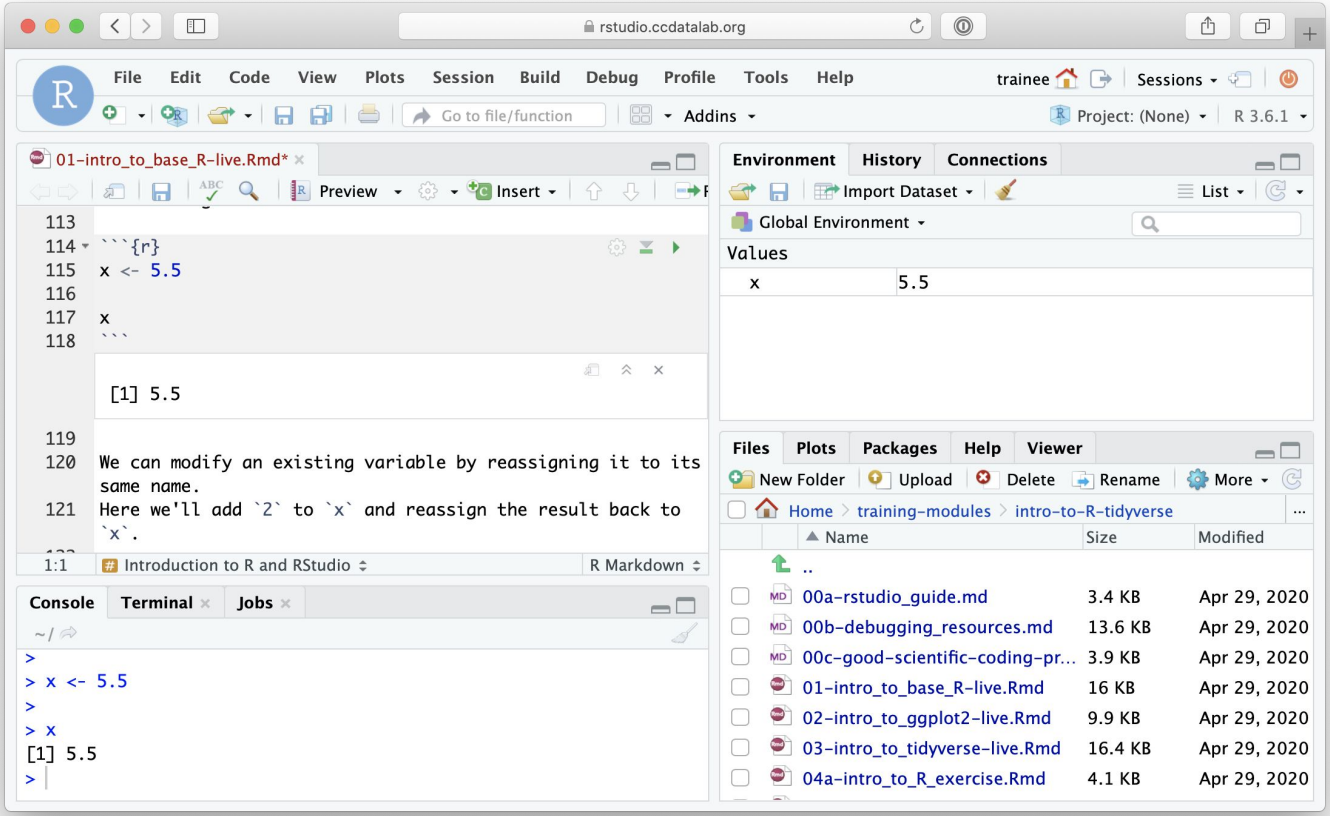
Single-cell RNA-seq Data in R: Import, QC, Normalize, & Visualize

The Data Lab

Before we begin, an RStudio primer/review



New R features that you will see: new pipe `|>`

- In past workshops, and/or if you have worked with `tidyverse` packages, you have probably seen the `magrittr` pipe: `%>%`
  - This allows "chaining" of functions in a readable way:
  - Instead of writing:
    `second_function(first_function(data))`,
    we can write things like:
    `data %>% first_function() %>% second_function()`
- In R version 4.1 and later, there is now a built-in version of this operator, `|>`, so we no longer have to load the `magrittr` package
  - `data |> first_function() |> second_function()`
  - There are some subtle differences between the two, but not much that comes up in normal use

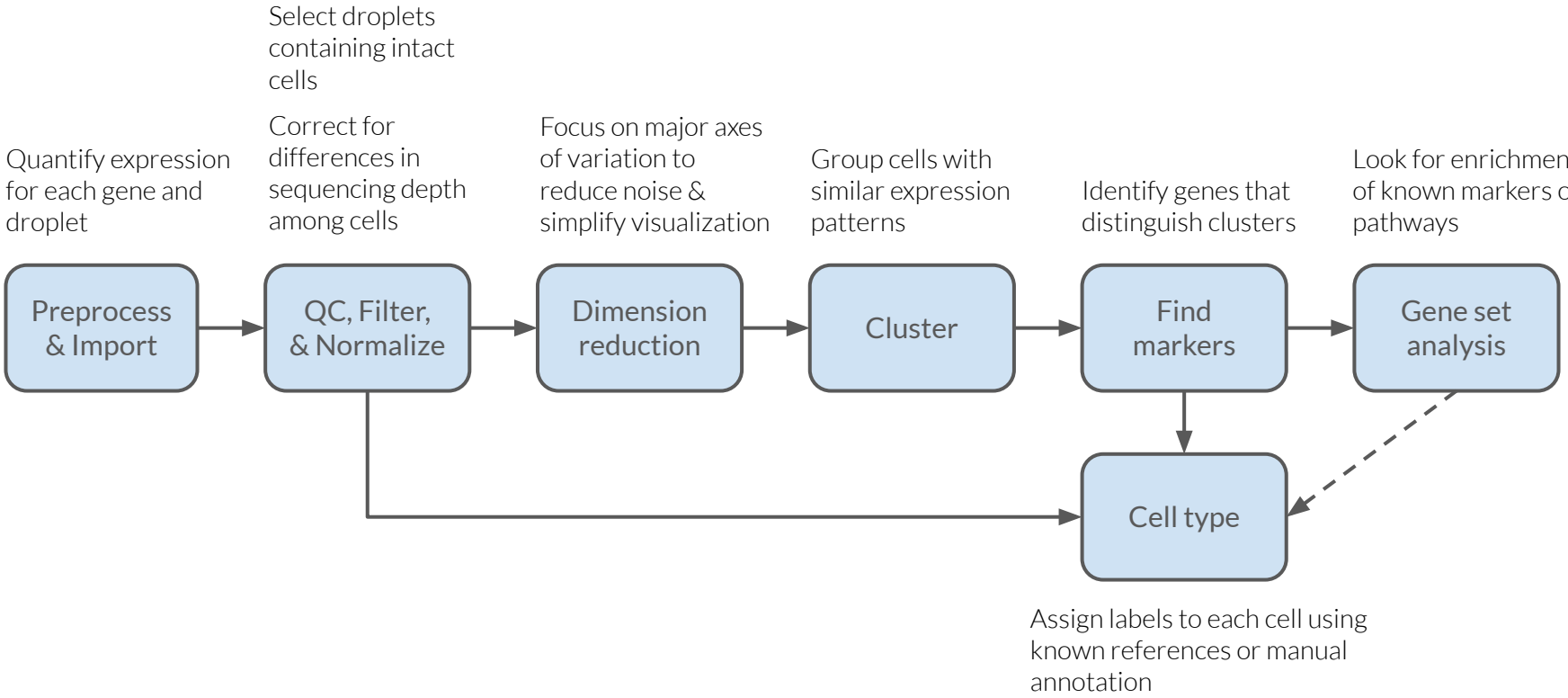
New R features that you will see: function shortcut `\(x)`

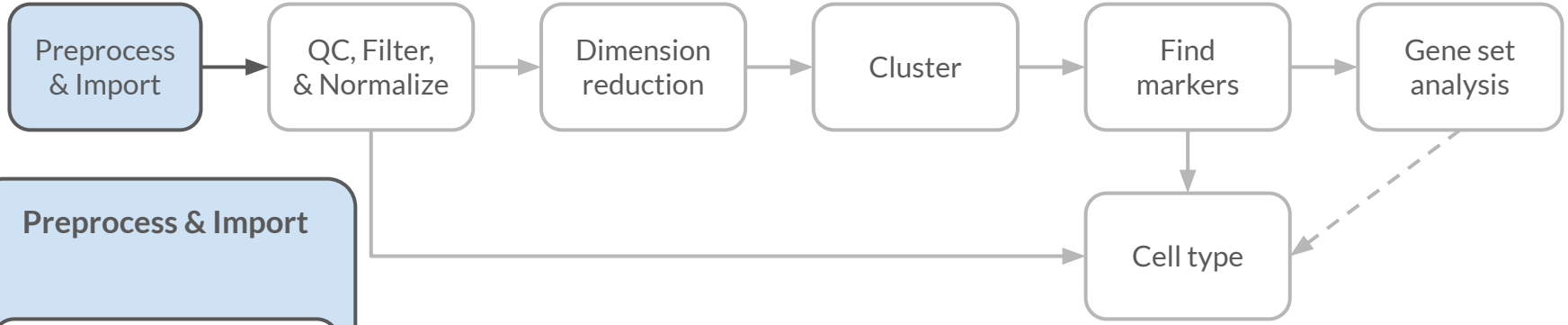
- R 4.1 also added a shortcut for making custom (little) functions
- A "regular" function is defined with the `function()` function:

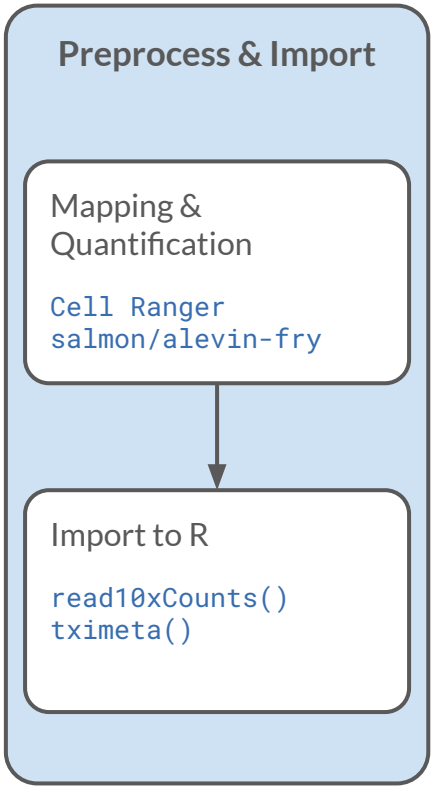
```
my_func <- function(x){
  (x + 1)^2
}
```

- Sometimes, we don't want to save our function, just use it quickly in another function (like `apply()` or a `purrr` package function)
  - In `purrr` functions, we could use a shortcut:
    `~(.x + 1)^2`
  - Now we can use a slightly more verbose but more flexible shortcut anywhere:
    `\(x) (x + 1)^2` or `\(n) {(n + 1)^2}`

Single sample scRNA-seq overview



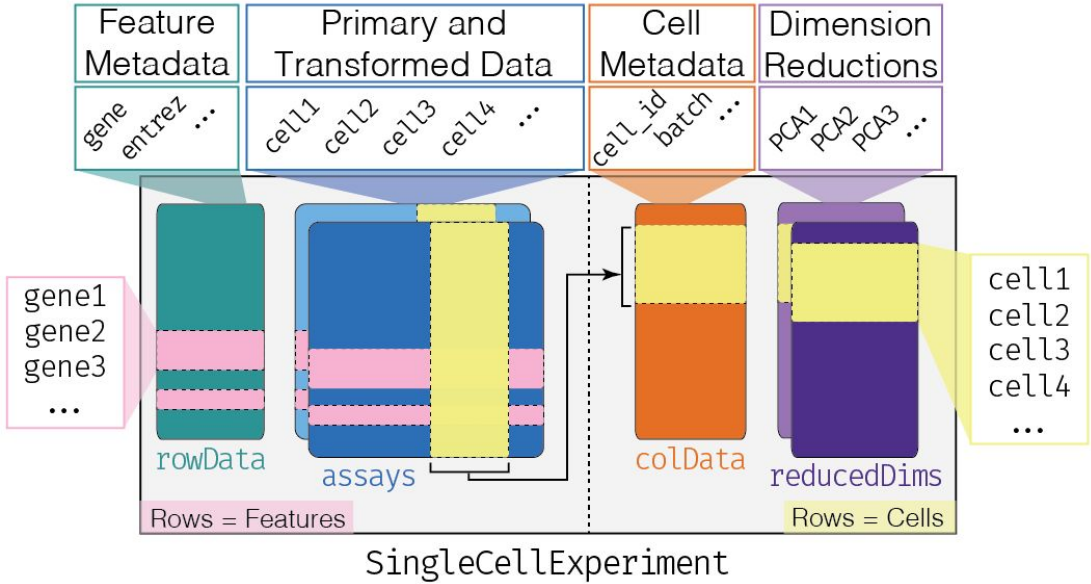




- Mapping & Quantification
  - Start with raw FASTQ files & a reference genome/transcriptome
  - End with a **gene-by-cell count matrix** (format varies by tool)
  - **Cell Ranger** is the most common tool
  - CITE-seq & cell hashing may require specialized processing
- Import to R
  - We will be using R/Bioconductor for processing and analysis
  - Commands for import vary based on the quantification tool used and file formats
  - R object we want to end up with is a `SingleCellExperiment`

The SingleCellExperiment class

- During this workshop, we will be working mostly with the Bioconductor suite of R packages
- Its main data class for storing single-cell data is the SingleCellExperiment (SCE)

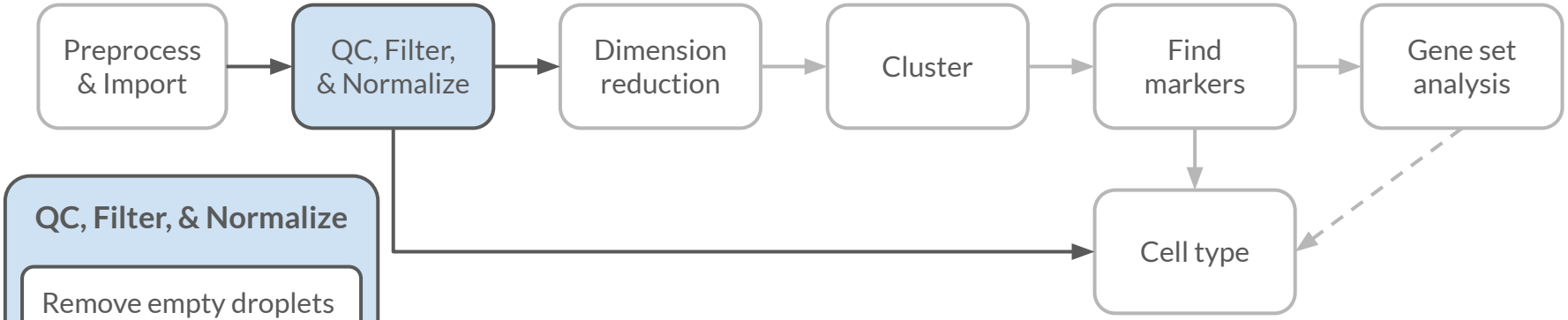


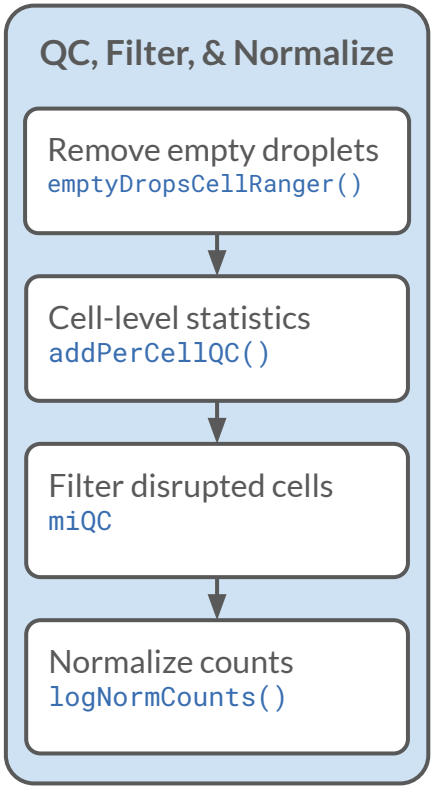
https://bioconductor.org/books/release/OSCA.intro/the-singlecellexperiment-class.html

Importing Data

- Single-cell data, after preprocessing/quantification* (or whenever you get it), may be in a variety of formats:
  - “Sparse” matrix files (mtx)
  - HDF5 files (from CellRanger, often)
  - LOOM (a special kind of HDF5)
  - AnnData (another special kind of HDF5 used by many Python tools)
  - SCE objects (in .rds files)
  - Seurat objects (in .rds files)
  - Excel tables
- Each type may require a different function for importing to an SCE object…
  - `DropletUtils::read10xCounts()`
  - `seurat::as.SingleCellExperiment()`
  - `zellkonverter::readH5AD()`

* we are not covering preprocessing here, but ask us about it!





- Quality Control and Filtering
  - Identify and remove low-quality cells
    - Empty droplets
    - Damaged/disrupted cells
    - Poorly sequenced cells (low counts)
- Normalization
  - Goal: Mitigate the effects of variation in sequencing depth across cells
  - Produce a log-scale counts matrix we can use for comparisons among cells

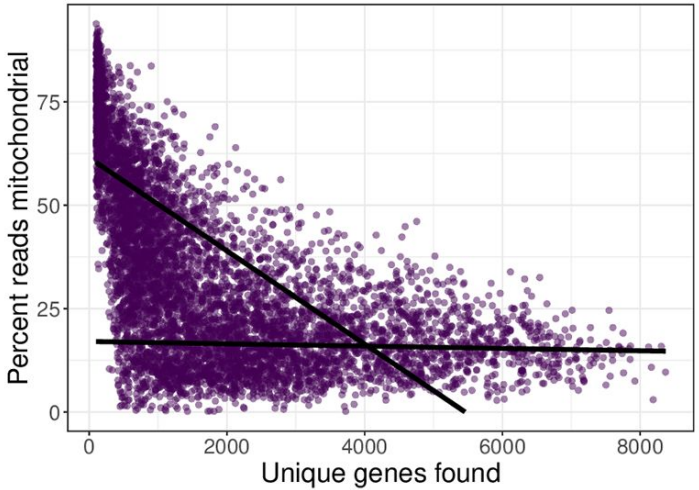
Initial Quality Control

- After preprocessing, you may have a raw and/or filtered matrix of count data
  - **Gene × Droplet** (cell) matrix with separate counts for each gene in each droplet
- Primary filtering is to remove “empty” droplets that did not contain a cell
  - Methods have changed over time, so different versions of Cell Ranger may have different contents of the filtered matrix
  - If you start with the raw matrix and filter yourself, you will know what was done!
    - and *maybe* can compare across versions, but other caveats for Cell Ranger version changes exist too!
    - the raw matrix is not usually too much larger, because the filtered droplets have mostly zero counts

Filtering damaged/disrupted/dying cells

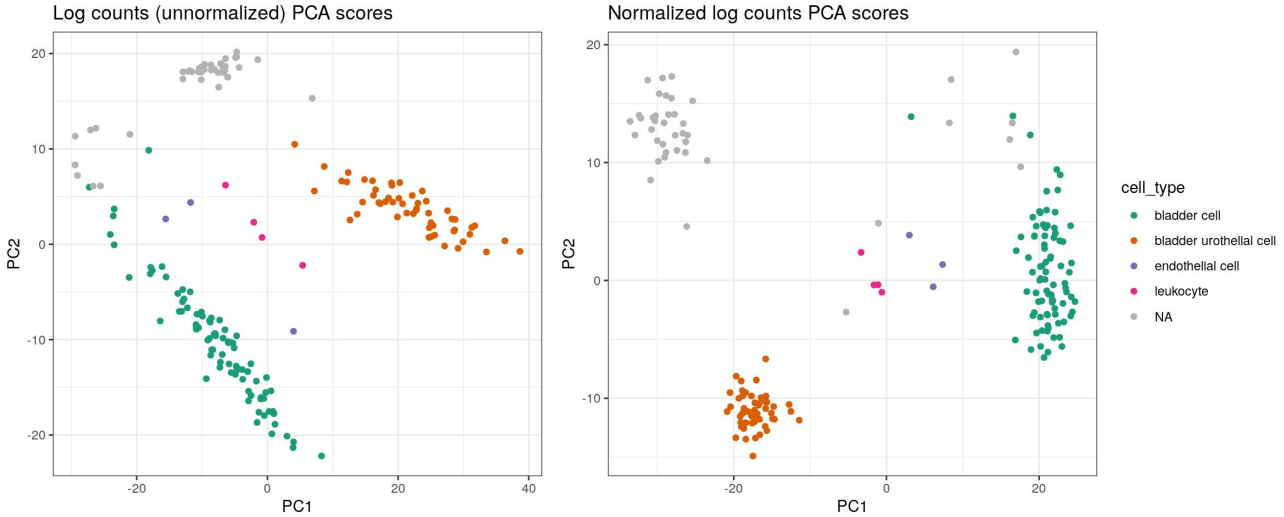
- During library preparation, cells may be broken prematurely
  - mRNA in the cytoplasm leaks out, giving unreliable (and usually lower) counts
  - mRNA in the mitochondria has an extra layer of protection (or 2) and will not leak out as readily
  - We can use the percentage of mitochondrial mRNA as an extra QC measure
  - But what cutoff should we use?

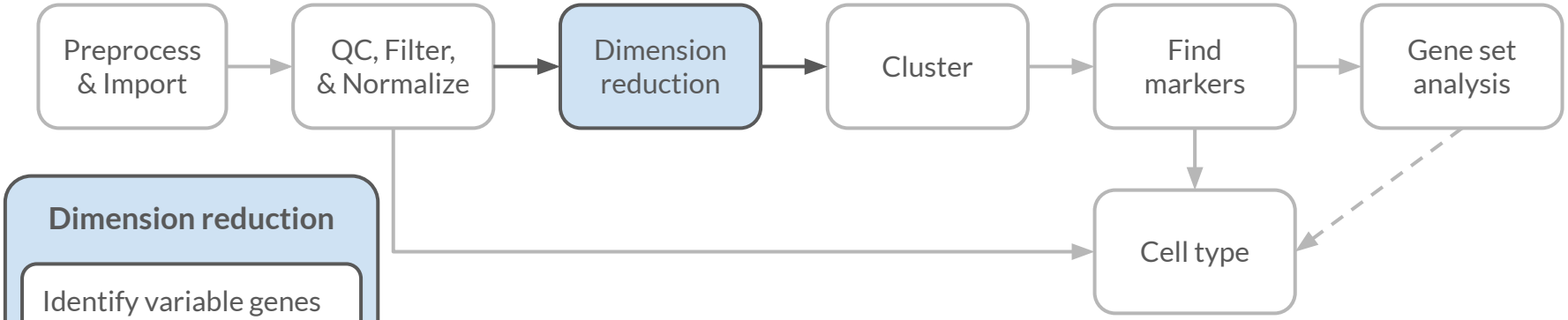
- miQC (Hippen *et al*. 2021) is a method that combines the total counts and the percentage of mitochondrial genes to identify likely-disrupted cells
  - https://doi.org/10.1371/journal.pcbi.1009290



Normalization

- The number of reads per cell often varies
  - This technical variation may mask biological variation
  - Normalization corrects per-cell counts for read depth





Dimension reduction

Identify variable genes

`modelGeneVar()`

PCA

`runPCA()`

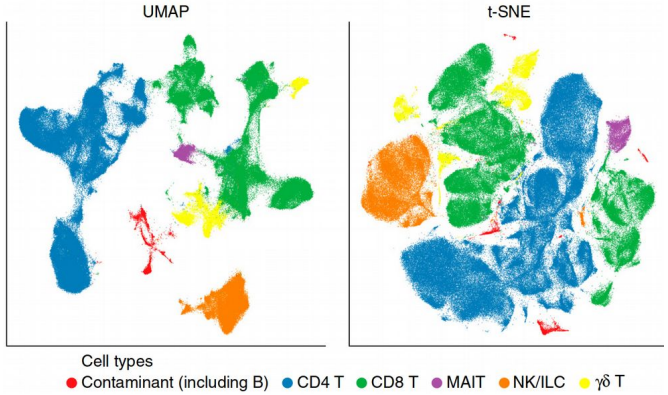
UMAP

`runUMAP()`

- Transcriptome data is highly multidimensional
  - Each gene's expression measurement is a separate dimension
  - Expression is often correlated among genes
- Goal: a more compact representation of the expression data with fewer dimensions
  - Reduce uninformative and redundant information
  - Increase "signal-to-noise" ratio
  - Speed up downstream calculations
  - Allow us to make visualizations that capture the important variation in the data

Dimensionality Reduction Methods

- Feature selection
  - Select the most (biologically) variable genes
- Principal Components Analysis
  - linear transformation of input data
  - usually to tens of dimensions
  - removes much of the noise; retains most of the signal
  - useful as input to many downstream analyses (clustering, etc.)
- UMAP and/or tSNE
  - reduce down to 2 or 3 dimensions
  - transformation is highly non-linear
  - much slower than PCA
  - nice for visualization, but be careful!
    - distances between points may be misleading
    - similar challenge to squashing a globe onto a flat map… but more extreme!



https://doi.org/10.1038/nbt.4314

Clustering Cells

Dimensionality reduction often results in visible “clusters”, but how do we define those?

Many methods!

- hierarchical clustering
  - join closest points/groups recursively
- k-means clustering
  - pick a number k, then find the “best” way to divide cells into that many groups
  - assumes clusters are “spherical”
- graph-based clustering
  - Connect cells to other cells with similar expression, then divide up the graph into clusters

Graph-based Clustering

Step 1: Calculate similarity matrix among points

Step 2: Build a weighted network graph connecting points to their neighbors

Step 3: Divide network graph into “neighborhoods” based on connection patterns

Many options at each step! The algorithms can determine how many clusters to assign.

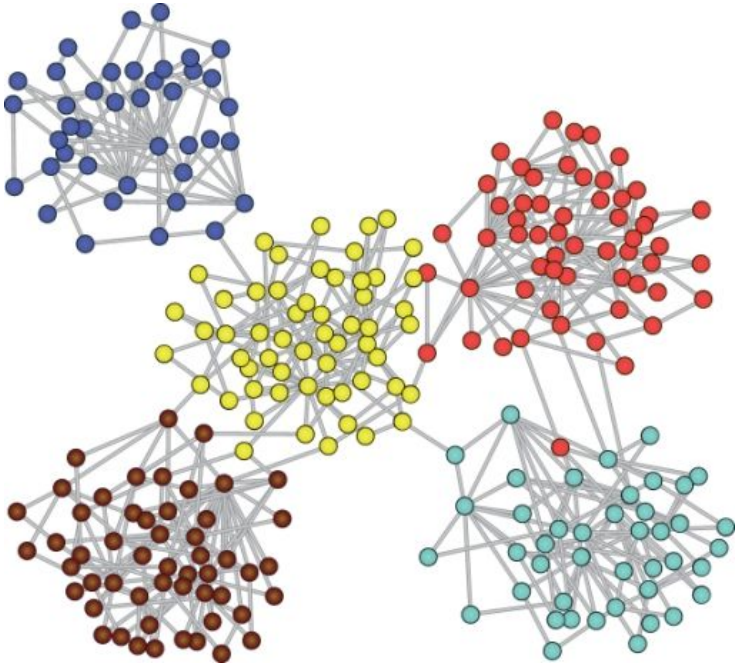


Image from:
https://github.com/benedekrozemberczki/awesome-community-detection

What do the clusters represent?

- Groups of cells with distinct gene expression patterns
- What does that mean?
  - maybe cell types?
  - sometimes cell states?
  - perhaps perturbations?

- Interpretation will vary based on the sample you are using!
  - do not expect a simple mapping of clusters to cell types

- Clustering is usually somewhat stochastic
  - parameter choice and random seeds will affect clusters
  - use caution when interpreting clustering results!
  - quantitative methods to evaluate cluster quality exist, but can be challenging to interpret