Data Infrastructure and Import

Content adapted from
https://osca.bioconductor.org/data-infrastructure.html

Background

Background

Common data infrastructure makes life easier

- Can switch between different tools more easily
- No need to convert between formats

What data do we need to analyse a scRNA-seq experiment?

What data do we start with?

Gene counts

	Cell 1	**Cell 2**	**...**	**Cell N**
Gene 1	0	1	...	0
Gene 2	1	3	...	0
...	...	...	...	...
Gene M	2	2		4

What data do we start with?

Information about the cells

	Barcode	**Donor**	**...**	**Treatment**
Cell 1	ACTGTA	D1	...	Drug
Cell 2	TGCATA	D1	...	Control
...	...	...	...	...
Cell N	CCTATA	D6		Drug

What data do we start with?

Information about the genes

	ID	Symbol	...	Chromosome
Gene 1	ENSG00000155816	FMN2	...	1
Gene 2	ENSG00000229807	XIST	...	X
...	...	...	...	...
Gene M	ENSG00000139618	BRCA2		13

What data do we create in an analysis?

Log-normalized gene expression

	Cell 1	**Cell 2**	**...**	**Cell N**
Gene 1	0	0.6	...	0
Gene 2	0.3	0.8	...	0
...	...	...	...	...
Gene M	0.35	0.67		2.1

What data do we create in an analysis?

Dimensionality reduction

	PCA 1	PCA 2	...	PCA K
Cell 1	0.93	1.28	...	0.03
Cell 2	0.32	1.22	...	0.09
...	...	...	...	...
Cell N	-0.66	1.00		0.15

	t-SNE 1	t-SNE 2
Cell 1	1.24	8.93
Cell 2	-0.33	7.85
...	...	...
Cell N	0.46	3.41

How to coordinate this?

Gene counts

	Cell 1	Cell 2	...	Cell N
Gene 1	0	1	...	0
Gene 2	1	3	...	0
...	...	...	...	...
Gene M	2	2		4

Information about the cells

	Barcode	Donor	...	Treatment
Cell 1	ACTGTA	D1	...	Drug
Cell 2	TGCATA	D1	...	Control
...	...	...	...	...
~~**Cell N**~~	~~CCTATA~~	~~D6~~		~~Drug~~

Information about the genes

	ID	Symbol	...	Chromosome
Gene 1	ENSG00000155816	FMN2	...	1
Gene 2	ENSG00000229807	XIST	...	X
...	...	...	...	...
Gene M	ENSG00000139618	BRCA2		13

How to coordinate this?

Gene counts

	Cell 1	Cell 2	...	Cell N
Gene 1	0	1	...	0
Gene 2	1	3	...	0
...	...	...	...	...
Gene M	2	2		4

Information about the cells

	Barcode	Donor	...	Treatment
Cell 1	ACTGTA	D1	...	Drug
Cell 2	TGCATA	D1	...	Control
...	...	...	...	...
Cell N	CCTATA	D6		Drug

Information about the genes

	ID	Symbol	...	Chromosome
Gene 1	ENSG00000155816	FMN2	...	1
Gene 2	ENSG00000229807	XIST	...	X
...	...	...	...	...
Gene M	ENSG00000139618	BRCA2		13

How to coordinate this?

Gene counts

	Cell 1	Cell 2	...	Cell N
Gene 1	0	1	...	0
Gene 2	1	3	...	0
...	...	...	...	...
Gene M	2	2		4

Information about the cells

	Barcode	Donor	...	Treatment
Cell 1	ACTGTA	D1	...	Drug
Cell 2	TGCATA	D1	...	Control
...	...	...	...	...
Cell N	CCTATA	D6		Drug

Information about the genes

	ID	Symbol	...	Chromosome
Gene 1	ENSG00000155816	FMN2	...	1
Gene 2	ENSG00000229807	XIST	...	X
...	...	...	...	...
Gene M	ENSG00000139618	BRCA2		13

How to coordinate this?

Gene counts

	Cell 1	Cell 2	...	Cell N
Gene 1	0	1	...	0
~~**Gene 2**~~	~~1~~	~~0~~	~~...~~	~~0~~
...	...	...	...	...
Gene M	2	2		4

Information about the cells

	Barcode	Donor	...	Treatment
Cell 1	ACTGTA	D1	...	Drug
Cell 2	TGCATA	D1	...	Control
...	...	...	...	...
~~**Cell N**~~	~~CCTATA~~	~~D6~~		~~Drug~~

Information about the genes

	ID	Symbol	...	Chromosome
Gene 1	ENSG00000155816	FMN2	...	1
~~**Gene 2**~~	~~ENSG00000229807~~	~~XIST~~	~~...~~	~~X~~
...	...	...	...	...
Gene M	ENSG00000139618	BRCA2		13

How to coordinate this?

Gene counts

	Cell 1	Cell 2	...	Cell N
Gene 1	0	1	...	0
~~**Gene 2**~~	~~1~~	~~0~~	~~...~~	~~0~~
...	...	...	...	...
Gene M	2	2		4

Also have to coordinate derived data (log-normalized gene expression, PCA, t-SNE, etc.)

Information about the cells

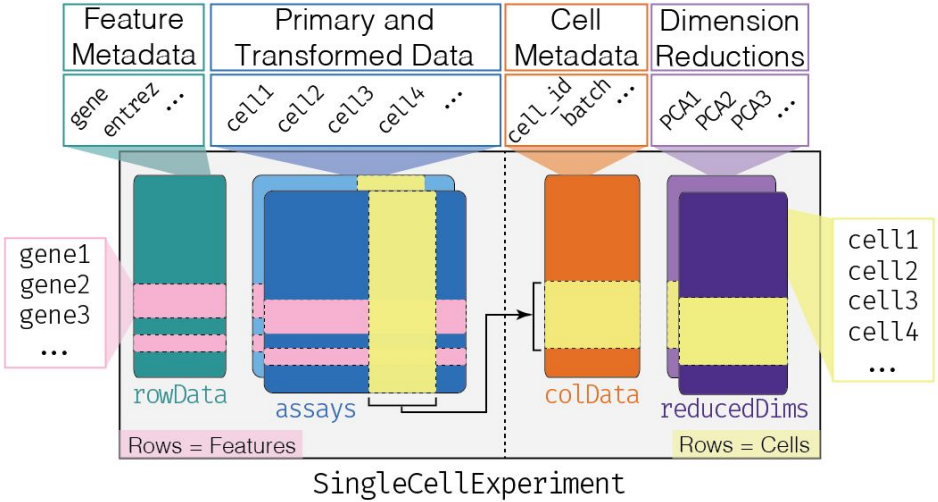
	Barcode	Donor	...	Treatment
Cell 1	ACTGTA	D1	...	Drug
Cell 2	TGCATA	D1	...	Control
...	...	...	...	...
~~**Cell N**~~	~~CCTATA~~	~~D6~~		~~Drug~~

Information about the genes

	ID	Symbol	...	Chromosome
Gene 1	ENSG00000155816	FMN2	...	1
~~**Gene 2**~~	~~ENSG00000229807~~	~~XIST~~	~~...~~	~~X~~
...	...	...	...	...
Gene M	ENSG00000139618	BRCA2		13

SingleCellExperiment

The anatomy of a *SingleCellExperiment*



Each piece of (meta)data in the *SingleCellExperiment* is represented by a separate 'slot'

An analogy

- An *SCE* is a cargo ship
- Each 'slot' is a cargo box
- Certain cargo boxes (slots) expect certain types of cargo (data)

Illustrative dataset: 416B

Illustrative dataset: 416B

```
library(scRNAseq)
sce.416b <- LunSpikeInData(which="416b")
```

- Immortalized mouse myeloid progenitor cell line processed using SmartSeq2
- https://osca.bioconductor.org/lun-416b-cell-line-smart-seq2.html

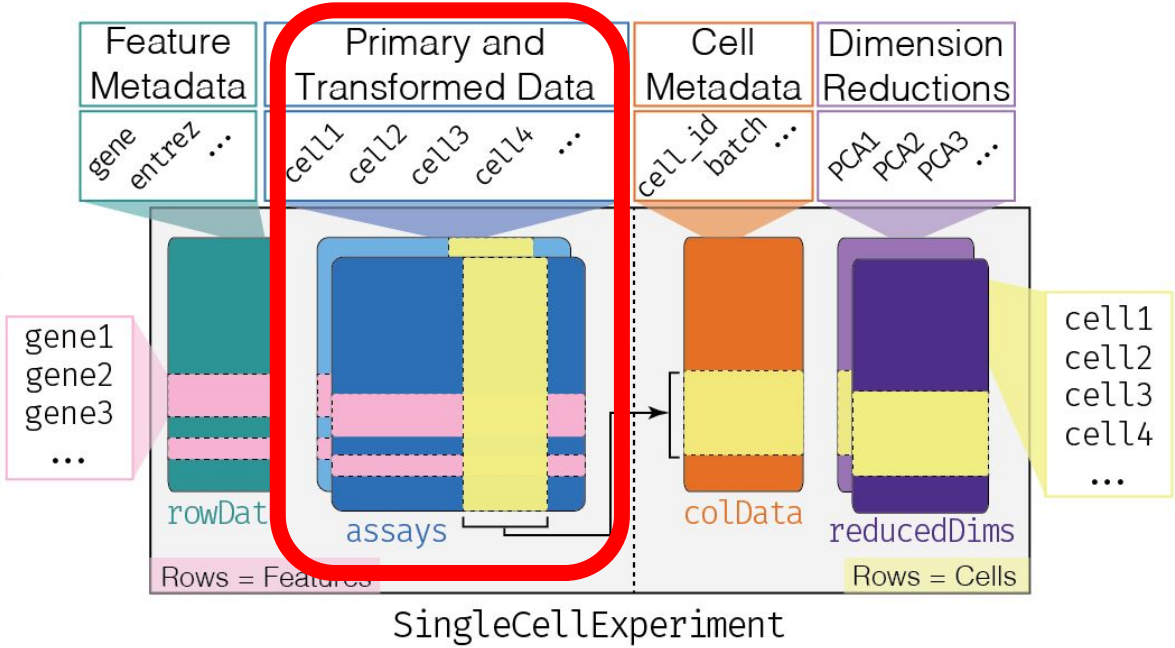
Lun, A. T. L., Calero-Nieto, F. J., Haim-Vilmovsky, L., Göttgens, B. & Marioni, J. C. Assessing the reliability of spike-in normalization for analyses of single-cell RNA sequencing data. *Genome Res.* **27**, 1795–1806 (2017)

Storing primary experimental data

Storing primary experimental data

Gene counts

	Cell 1	Cell 2	...	Cell N
Gene 1	0	1	...	0
Gene 2	1	3	...	0
...	...	...	...	...
Gene M	2	2		4



Filling the assays slot

```r
# Load the SingleCellExperiment package
library(SingleCellExperiment)
# Extract the count matrix from the 416b dataset
counts.416b <- counts(sce.416b)
# Construct a new SCE from the counts matrix
sce <- SingleCellExperiment(
  assays = list(counts = counts.416b))
```

Filling the assays slot

```
# Inspect the object we just created
sce
```

Filling the assays slot

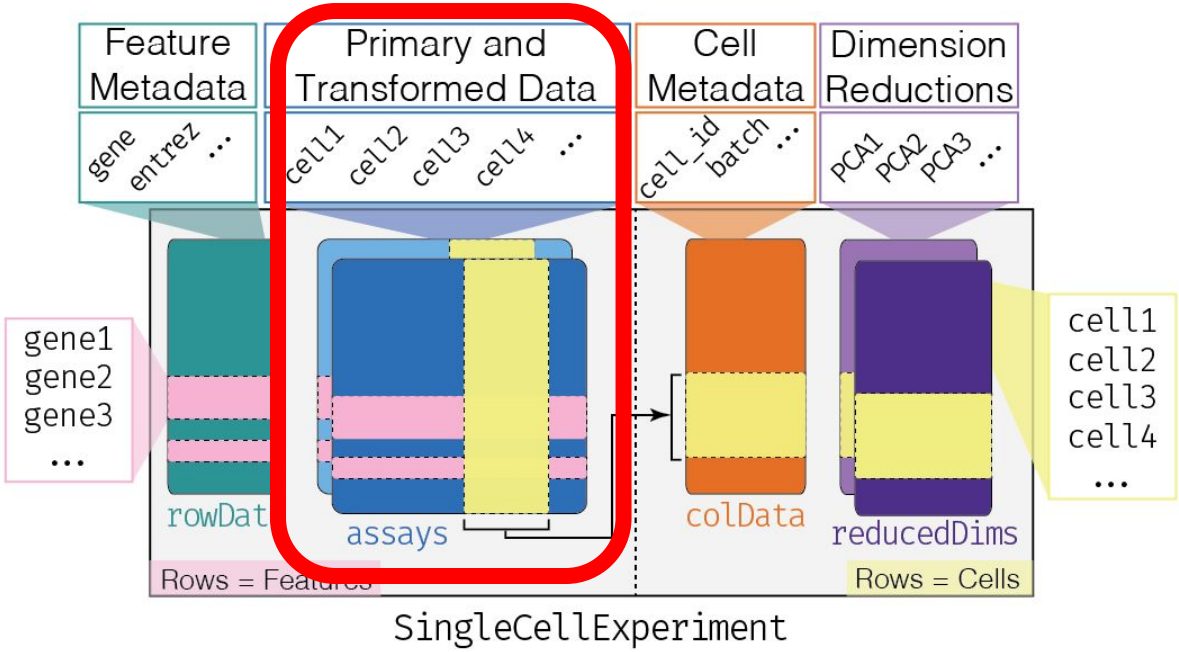
```r
# Access the counts matrix from the assays slot
# WARNING: This will flood RStudio with output!

# 1. The general method
assay(sce, "counts")
# 2. The special method for the assay named "counts"
counts(sce)

# Tip: Limit the output to just a few samples & genes
counts(sce)[1:30, 1:2]
```

Adding to the assays slot

```
sce <- scater::logNormCounts(sce)
# Inspect the object we just updated
sce
```



Adding to the assays slot

```r
sce <- scater::logNormCounts(sce)
# Inspect the object we just updated
sce
```

👉We overwrote our previous 'sce' by reassigning the results back to 'sce' (possible because this particular function returns a 'SingleCellExperiment' that contains the results in addition to the original data)

⚠️ Some functions - especially those outside the single-cell oriented Bioconductor packages - do no, then need to do extra work

Adding more assays

```r
# Access the logcounts matrix from the assays slot
# WARNING: This will flood RStudio with output!

# 1. The general method
assay(sce, "logcounts")
# 2. The special method for the assay named "logcounts"
logcounts(sce)

# Tip: Limit the output to just a few samples & genes
logcounts(sce)[1:30, 1:2]
```

Adding more assays

🤔 What if I want to add a custom assay?

```r
# assign a new entry to assays slot
assay(sce, "counts_100") <- assay(sce, "counts") + 100
# List the assays in the object
assays(sce)
```

Storing metadata

Storing metadata

Information about the cells

	Barcode	Donor	...	Treatment
Cell 1	ACTGTA	D1	...	Drug
Cell 2	TGCATA	D1	...	Control
...	...	...	...	...
Cell N	CCTATA	D6		Drug

Information about the genes

	ID	Symbol	...	Chromosome
Gene 1	ENSG00000155816	FMN2	...	1
Gene 2	ENSG00000229807	XIST	...	X
...	...	...	...	...
Gene M	ENSG00000139618	BRCA2		13

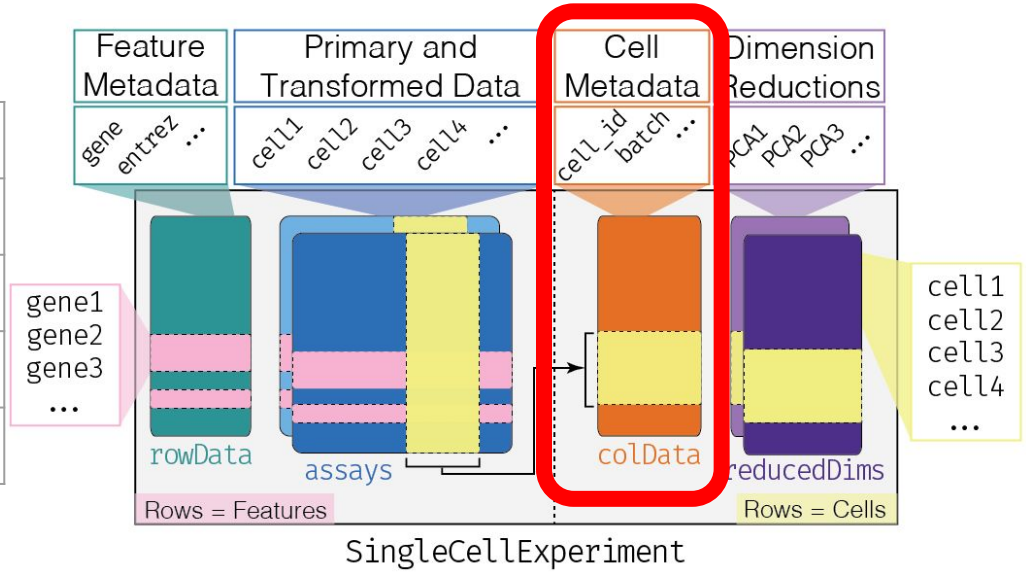
Information about the experiment

- "Look at expression of genes A, B, and C"
- "There could be problems with samples from the first batch because of issue with FACS sort"

Storing metadata on the columns (cells)

Information about the cells

	Barcode	Donor	...	Treatment
Cell 1	ACTGTA	D1	...	Drug
Cell 2	TGCATA	D1	...	Control
...	...	...	...	...
Cell N	CCTATA	D6		Drug



Filling the colData slot

```r
# Extract the sample metadata from the 416b dataset
colData.416b <- colData(sce.416b)
# Add some of the sample metadata to our SCE
colData(sce) <- colData.416b[, c("phenotype", "block")]
# Inspect the object we just updated
sce
# Access the sample metadata from our SCE
colData(sce)
# Access a specific column of sample metadata from our SCE
sce$block
```

Adding to the colData slot

```r
# Example of function that adds extra fields to colData
sce <- scater::addPerCellQC(sce.416b)
# Access the sample metadata from our updated SCE
colData(sce)
```

Using colData for subsetting

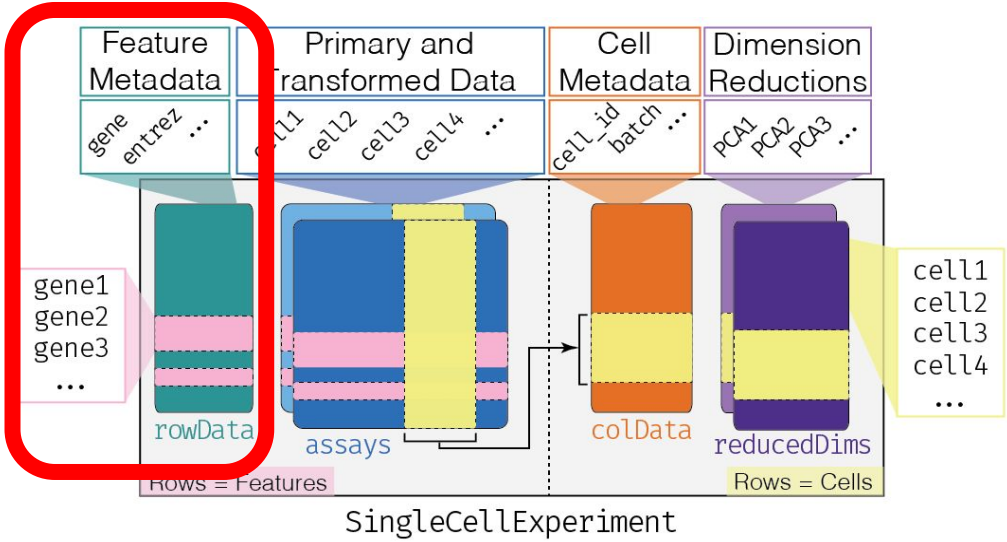
🤔 What if I want to subset the data to certain cells?

```
# E.g., subset data to just wild type cells
# Remember, cells are columns of the SCE
sce[, sce$phenotype == "wild type phenotype"]
```

Storing metadata on the rows (features/genes)

Information about the genes

	ID	Symbol	...	Chromosome
Gene 1	ENSG00000155816	FMN2	...	1
Gene 2	ENSG00000229807	XIST	...	X
...	...	...	...	...
Gene M	ENSG00000139618	BRCA2		13



Adding to the rowData slot

```r
# Access the feature metadata from our SCE
# It's currently empty!
rowData(sce)

# Example of function that adds extra fields to rowData
sce <- scater::addPerFeatureQC(sce)
# Access the feature metadata from our updated SCE
rowData(sce)
```

Adding to the rowData slot

🤔 What if I want to add the chromosome of each gene?

```r
# Download the relevant Ensembl annotation database
# using AnnotationHub resources
library(AnnotationHub)
ah <- AnnotationHub()
query(ah, c("Mus musculus", "Ensembl", "v97"))
```

Adding to the rowData slot

```r
# Annotate each gene with its chromosome location
ensdb <- ah[["AH73905"]]
chromosome <- mapIds(ensdb, keys=rownames(sce),
    keytype="GENEID", column="SEQNAME")
rowData(sce)$chromosome <- chromosome

# Access the feature metadata from our updated SCE
rowData(sce)
```

Using rowData for subsetting

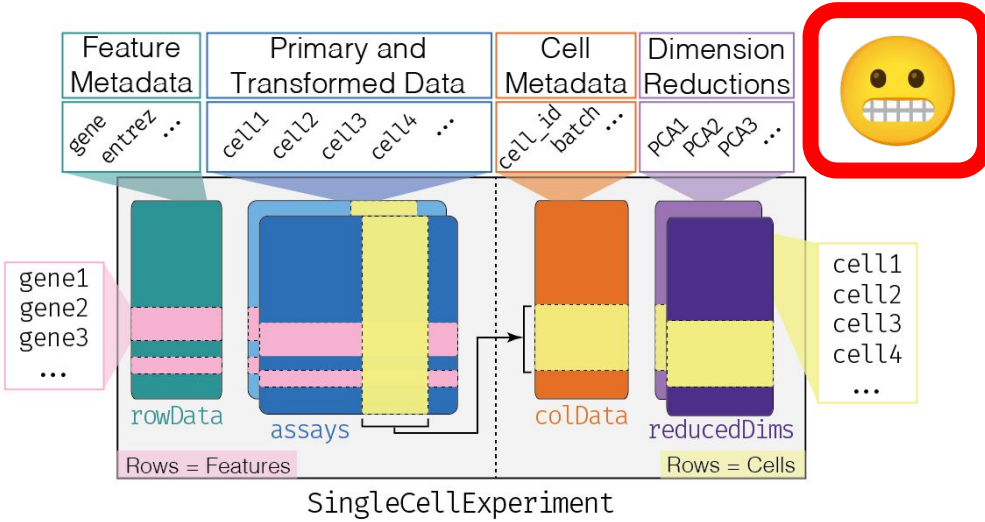
🤔 What if I want to subset the data to certain features?

```
# E.g., subset data to just genes on chromosome 3
# NOTE: which() needed to cope with NA chromosome names
sce[which(rowData(sce)$chromosome == "3"), ]
```

Storing other metadata

Information about the experiment

- "Look at expression of genes A, B, and C"
- "There could be problems with samples from the first batch because of issue with FACS sort"



Adding to the metadata slot

```r
# Access the metadata from our SCE
# It's currently empty!
metadata(sce)

# The metadata slot is Vegas - anything goes
metadata(sce) <- list(
  favourite_genes=c("Shh", "Nck1", "Diablo"),
  analyst=c("Pete"))

# Access the metadata from our updated SCE
metadata(sce)
```

Storing single-cell-specific data

Background

So far, we've covered:

- 'assays' (primary data)
- 'colData' (cell metadata)
- 'rowData' (feature metadata)
- 'metadata' (experiment-level metadata)

🤓 The above is a *SummarizedExperiment*

So why do we need *SingleCellExperiment*?

For single-cell data, we commonly also have:

- Dimensionality reduction results (e.g., t-SNE, UMAP)
- Special or 'alternative' types of features (e.g., spike-in RNAs, antibody-derived sequencing tags)

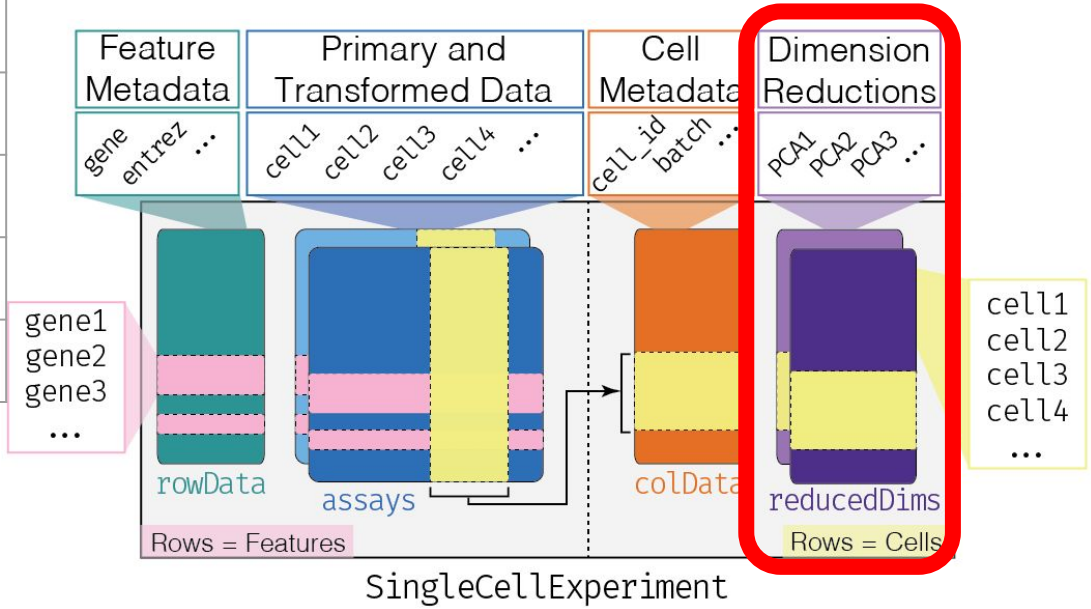
🤓 *SingleCellExperiment* is an extension of *SummarizedExperiment*

Storing dimensionality reduction results

Dimensionality reduction

	PCA 1	PCA 2	...	PCA K
Cell 1	0.93	1.28	...	0.03
Cell 2	0.32	1.22	...	0.09
...	...	...	...	...
Cell N	-0.66	1.00		0.15

	t-SNE 1	t-SNE 2
Cell 1	1.24	8.93
Cell 2	-0.33	7.85
...	...	...
Cell N	0.46	3.41



Adding to the reducedDims slot

🤔 What if I want to store a dimensionality reduced version of the data?

```r
# E.g., add the PCA of logcounts
# NOTE: We'll learn more about PCA later
sce <- scater::runPCA(sce)
# Inspect the object we just updated
sce
# Access the PCA matrix from the reducedDims slot
reducedDim(sce, "PCA")
```

Adding to the reducedDims slot

🤔 What if I want to store a dimensionality reduced version of the data?

```r
# E.g., add a t-SNE representation of logcounts
# NOTE: We'll learn more about t-SNE later
sce <- scater::runTSNE(sce)
# Inspect the object we just updated
sce
# Access the t-SNE matrix from the reducedDims slot
reducedDim(sce, "TSNE")
```

Adding to the reducedDims slot

🤔 What if I want to store a dimensionality reduced version of the data?

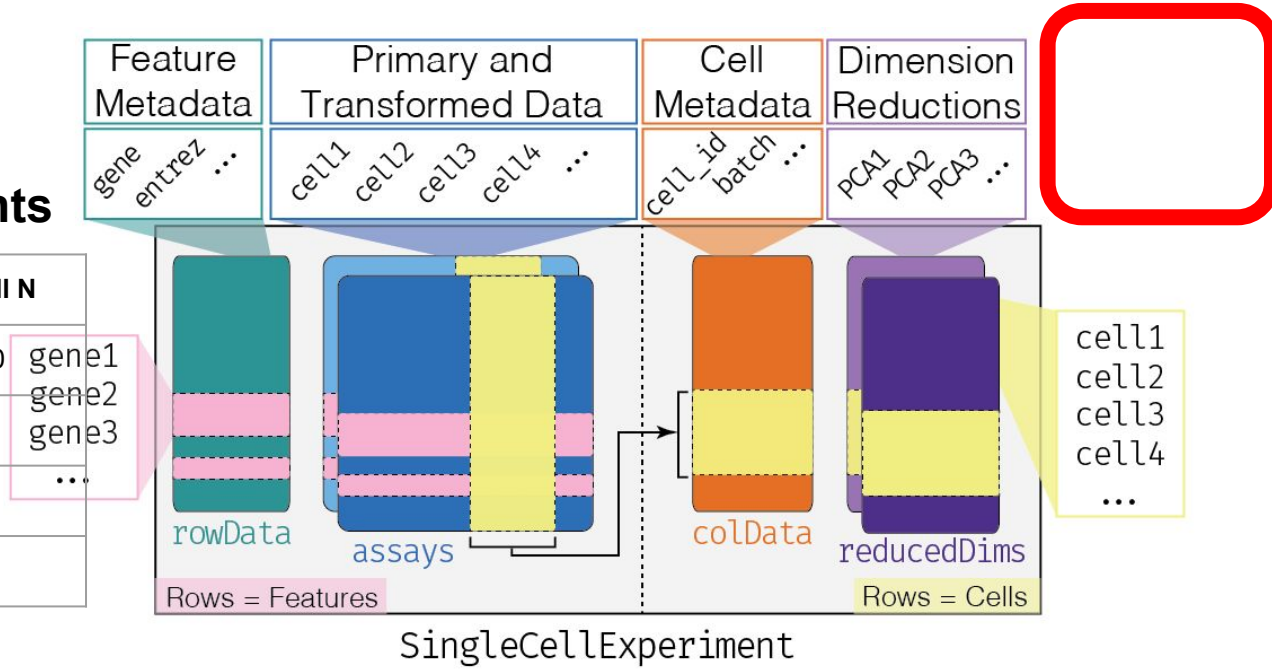
```
# E.g., add a 'manual' UMAP representation of logcounts
# NOTE: We'll learn more about UMAP later and a
#        simpler way to compute it.
u <- uwot::umap(t(logcounts(sce)), n_component=2)
# Add the UMAP matrix to the reducedDims slot
# Access the UMAP matrix from the reducedDims slot
reducedDim(sce, "UMAP") <- u

# List the dimensionality reduction results stored in # the
object
reducedDims(sce)
```

Storing 'alternative experiments'

Antibody-derived tag counts

	Cell 1	Cell 2	...	Cell N
AB 1	60	1	...	450
AB 2	160	300	...	23
...	...	...	...	...
AB M'	20	50		9



Adding an alternative experiment

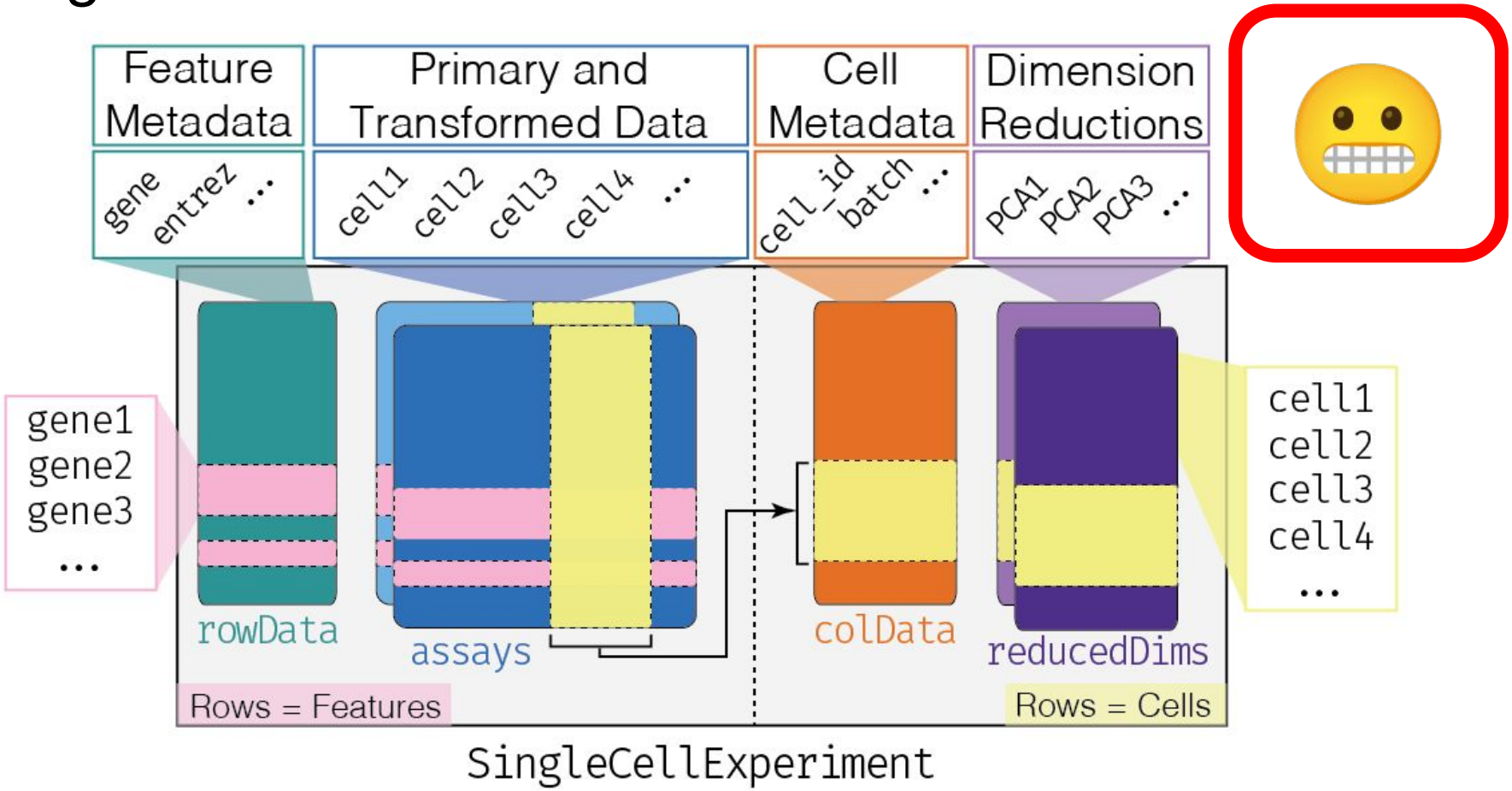
```r
# Extract the ERCC SCE from the 416b dataset
ercc.sce.416b <- altExp(sce.416b, "ERCC")
# Inspect the ERCC SCE
ercc.sce.416b

# Add the ERCC SCE as an alternative experiment to our SCE
altExp(sce, "ERCC") <- ercc.sce.416b
# Inspect the object we just updated
sce
# List the alternative experiments stored in the object
altExps(sce)
```

Why use alternative experiments?

```r
# Subsetting the SCE by sample also subsets the
# alternative experiments
sce.subset <- sce[, 1:10]
ncol(sce.subset)
ncol(altExp(sce.subset))
```

Storing size factors



Adding size factors

```r
# Extract existing size factors (these were added
# when we ran scater::logNormCounts(sce))
sizeFactors(sce)

# 'Automatically' replace size factors
sce <- scran::computeSumFactors(sce)
sizeFactors(sce)

# 'Manually' replace size factors
sizeFactors(sce) <- scater::librarySizeFactors(sce)
sizeFactors(sce)
```

Importing your data to construct a *SingleCellExperiment*

Importing your data to construct a *SingleCellExperiment*

- I ran CellRanger
  - 👉 `DropletUtils::read10xCounts()`
- I ran scPipe
  - 👉 `scPipe::create_sce_by_dir()`
- I got a bunch of files (e.g., .csv or .mtx files)
  - 👉 General file importer
    - `utils::read.delim()`
    - `data.table::fread()`
  - 👉 Specialist file importer
    - `scater::readSparseCounts()`
    - `Matrix::readMM()`
  - 😩 Lots of duct tape and swearing

I ran CellRanger

Download and extract example data

```
# Download example data processed with CellRanger
# Aside: Using BiocFileCache means we only download the
#        data once
library(BiocFileCache)
bfc <- BiocFileCache()
pbmc.url <-
"http://cf.10xgenomics.com/samples/cell-vdj/3.1.0/vdj_v1_h
s_pbmc3/vdj_v1_hs_pbmc3_filtered_feature_bc_matrix.tar.gz"
pbmc.data <- bfcrpath(bfc, pbmc.url)

# Extract the files to a temporary location
untar(pbmc.data, exdir=tempdir())
```

Typical CellRanger output files

```r
# List the files we downloaded and extracted
# These files are typically CellRanger outputs
pbmc.dir <- file.path(tempdir(),
  "filtered_feature_bc_matrix")
list.files(pbmc.dir)
```

Import CellRanger outputs as a SingleCellExperiment

```r
# Import the data as a SingleCellExperiment
library(DropletUtils)
sce.pbmc <- read10xCounts(pbmc.dir)
# Inspect the object we just constructed
sce.pbmc
```

Some polish

```r
# Store the CITE-seq data in an alternative experiment
sce.pbmc <- splitAltExps(sce.pbmc, rowData(sce.pbmc)$Type)
# Inspect the object we just updated
sce.pbmc
```

I ran scPipe

Download and extract example data

```r
# Download example data processed with scPipe
library(BiocFileCache)
bfc <- BiocFileCache()
sis_seq.url <-
"https://github.com/LuyiTian/SIS-seq_script/archive/master
.zip"
sis_seq.data <- bfcrpath(bfc, sis_seq.url)

# Extract the files to a temporary location
unzip(sis_seq.data, exdir=tempdir())
```

Typical scPipe outputs

```r
# List (some of) the files we downloaded and extracted
# These files are typical scPipe outputs
sis_seq.dir <- file.path(tempdir(),
  "SIS-seq_script-master", "data", "BcorKO_scRNAseq",
  "RPI10")
list.files(sis_seq.dir)
```

Import scPipe outputs as a SingleCellExperiment

```r
# Import the data as a SingleCellExperiment
library(scPipe)
sce.sis_seq <- create_sce_by_dir(sis_seq.dir)
# Inspect the object we just constructed
sce.sis_seq
```

I got a bunch of files

Download example data

```r
# Download example bunch o' files dataset
library(BiocFileCache)
bfc <- BiocFileCache()
lun_counts.url <-
"https://www.ebi.ac.uk/arrayexpress/files/E-MTAB-5522/E-MT
AB-5522.processed.1.zip"
lun_counts.data <- bfcrpath(bfc, lun_counts.url)
lun_coldata.url <-
"https://www.ebi.ac.uk/arrayexpress/files/E-MTAB-5522/E-MTA
B-5522.sdrf.txt"
lun_coldata.data <- bfcrpath(bfc, lun_coldata.url)
```

Extract example data

```r
# Extract the counts files to a temporary location
lun_counts.dir <- tempfile("lun_counts.")
unzip(lun_counts.data, exdir=lun_counts.dir)
```

Typical (actually, not too bad) bunch o' files

```
# List the files we downloaded and extracted
list.files(lun_counts.dir)
```

Import the count matrix and tidy it up

```r
# Import the count matrix (for 1 plate)
lun.counts <- read.delim(
  file.path(lun_counts.dir, "counts_Calero_20160113.tsv"),
  header=TRUE,
  row.names=1,
  check.names=FALSE)
# Store the gene lengths for later
gene.lengths <- lun.counts$Length
# Convert the gene counts to a matrix
lun.counts <- as.matrix(lun.counts[, -1])
```

Import the sample metadata (skipping the tidying)

```r
# Import the sample metadata
lun.coldata <- read.delim(lun_coldata.data,
  check.names=FALSE, stringsAsFactors=FALSE)
library(S4Vectors)
lun.coldata <- as(lun.coldata, "DataFrame")

# Match up the sample metadata to the counts matrix
m <- match(
  colnames(lun.counts),
  lun.coldata$`Source Name`)
lun.coldata <- lun.coldata[m, ]
```

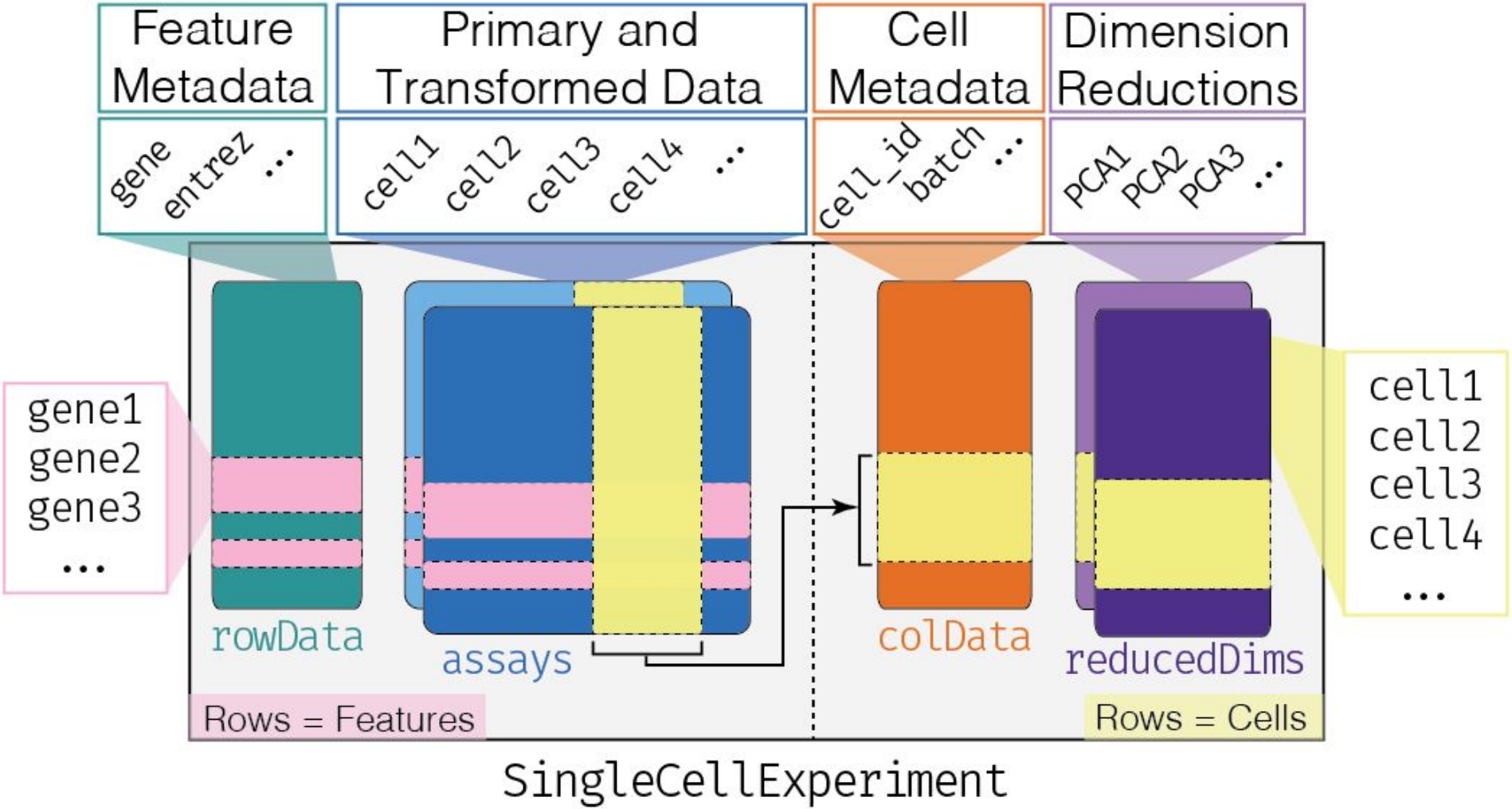
Construct the feature metadata

```
# Construct the feature metadata
lun.rowdata <- DataFrame(Length = gene.lengths)
```

Bring in the duct tape to stick everything together

```r
# Construct the SingleCellExperiment
lun.sce <- SingleCellExperiment(
  assays = list(assays = lun.counts),
  colData = lun.coldata,
  rowData = lun.rowdata)
# Inspect the object we just constructed
lun.sce
```

Summary and recommendations


Feature Metadata
gene
entrez
...
Primary and Transformed Data
cell1
cell2
cell3
cell4
...
Cell Metadata
cell_id
batch
...
Dimension Reductions
PCA1
PCA2
PCA3
...
gene1
gene2
gene3
...
rowData
assays
colData
reducedDims
cell1
cell2
cell3
cell4
...
Rows = Features
Rows = Cells
SingleCellExperiment

Summary and recommendations

👉Use a *SingleCellExperiment* if using R / Bioconductor to analyse your single-cell data

- ➕ Keep experimental data, metadata, and derived data in-sync
- ➕ Interoperability with 70+ single-cell-related Bioconductor packages
- ➕ Convenient to share with collaborators for further analysis

😩 But I need to use Seurat, scanpy, etc. for a certain step

- 👉 Extract the bit needed (e.g., counts matrix, PCA) from the *SCE*
- 👀 https://osca.bioconductor.org/interoperability.html (in-progress)

Importing your data to construct a *SingleCellExperiment*

- I got the data from SCORE
  - 👉 `sce <- readRDS("path/to/SCE.rds")`
- I ran CellRanger
  - 👉 `DropletUtils::read10xCounts()`
- I ran scPipe
  - 👉 `scPipe::create_sce_by_dir()`
- I got a bunch of files (e.g., .csv or .mtx files)
  - 👉 General file importer
    - `utils::read.delim()`
    - `data.table::fread()`
  - 👉 Specialist file importer
    - `scater::readSparseCounts()`
    - `Matrix::readMM()`

Where we're at