Scaffold: simulating single-cell RNA-seq data

Rhonda Bacher

30 May 2021

Contents

1 Introduction

`Scaffold` is an R package for generating scRNA-seq data by statistically modelling each step of the experimental process. Simulation parameters can be estimated from real datasets, and a comprehensive plotting function is provided to generate summary figures comparing the real and simulated data. This vignette provides details and examples using the Scaffold package.

2 Installation

To install `Scaffold` via GitHub:

```
devtools::install_github("rhondabacher/scaffold")
```

3 Run Scaffold

After successful installation, the package must be loaded into the working space:

```
library(scaffold)
```

3.1 Estimate simulation parameters

The most common way to use Scaffold is to use an input dataset from which various parameters are estimated and then used to simulate data. The input dataset can be formatted as a data matrix or of the `SingleCellExperiment` class. The `scaffold` package provides the `uneq_ec_data` dataset to demonstrate formatting data as a `SingleCellExperiment` class:

```
# Loading the example data
data(uneq_ec_data)

# Loading the SingleCellExperiment package
if (!requireNamespace("SingleCellExperiment", quietly = TRUE))
  BiocManager::install("SingleCellExperiment")
library(SingleCellExperiment)

# Creating the SingleCellExperiment class object:
sce <- SingleCellExperiment(list(counts = uneq_ec_data))
sce
```

```
## class: SingleCellExperiment
## dim: 17490 41
## metadata(0):
## assays(1): counts
## rownames(17490): A1BG A1CF ... ZZEF1 ZZZ3
## rowData names(0):
## colnames(41): ECb2_072 ECb2_073 ... ECb2_123 ECb2_125
## colData names(0):
## reducedDimNames(0):
## altExpNames(0):
```

Now that we have the appropriate format for our input dataset, we first run the `estimateScaffoldParameters` function to set or estimate simulation parameters. The default parameters are set to simulate nonUMI data from the C1/Smart-seq protocol. To change this, adjust the parameters `useUMI` (logical; whether to make the protocol UMI based) and `protocol` ("C1", "droplet" and "10X" are all acceptable). If the input dataset already contains UMI counts, then this should be specified by setting parameters `sceUMI = TRUE`.

```
scaffoldParams <- estimateScaffoldParameters(sce, sceUMI=FALSE, useUMI = FALSE,
protocol="C1")
```

The `estimateScaffoldParameters` function documentation describes all the parameter options. A table of all the parameter options with notes is also given in this vignette in Section 6.

3.2 Simulate data

Once the `ScaffoldParams` object is constructed, simulations can be run with the `simulateScaffold` function. The inputs to this function are the parameter object and the original sce dataset.

```
sce_sim <- simulateScaffold(scaffoldParams, sce)
```

The output `sce_sim` is an object of class `SingleCellExperiment` that contains the simulated gene counts.

To access the simulated counts:

```
simcounts <- counts(sce_sim)
simcounts[1:5,1:5]
```

4 Advanced simulation options

4.1 Simulating UMI, droplet, or 10X data

To simulate data with UMIs or from the droplet/10X protocols the following parameters can be adjusted:

```
## The following will set the parameters as if the C1 protocol was UMI based:
scaffoldParams <- estimateScaffoldParameters(sce, sceUMI = FALSE, useUMI = TRUE,
 protocol="C1")

## The following will set the parameters as if the droplet/10X protocol was used
.
scaffoldParams <- estimateScaffoldParameters(sce, sceUMI = FALSE, useUMI = TRUE,
 protocol="10X")
```

Let's look at an example generating UMI counts. When the UMI option is used, the output of `simulateScaffold` contains two matrices, one having the read counts and the other with the umi counts.

```
scaffoldParams <- estimateScaffoldParameters(sce, sceUMI = FALSE, useUMI = TRUE,
 protocol="C1")
sce_sim <- simulateScaffold(scaffoldParams, sce)
```

```
## [1] "Estimating capture efficiency..."
## [1] "Finished estimating capture efficiency!"
## [1] "Starting capture step (lysis and reverse transcription)..."
## [1] "Finished capture step!"
## [1] "Starting preamplify step..."
## [1] "Finished preamplify!"
## [1] "Starting library prep and sequencing..."
## [1] "Beginning formatting output..."
## [1] "Processing output for 41 cells."
## [1] "Finished sequencing and data formatting!"
```

```
sce_sim
```

```
## class: SingleCellExperiment
## dim: 17490 41
## metadata(1): initialSimCounts
## assays(2): counts umi_counts
## rownames(17490): A1BG A1CF ... ZZEF1 ZZZ3
## rowData names(0):
## colnames(41): Cell_1 Cell_2 ... Cell_40 Cell_41
## colData names(2): capEfficiency cellPopulation
## reducedDimNames(0):
## altExpNames(0):
```

To access the umi counts:

```
simumis <- sce_sim@assays@data$umi_counts
simumis[1:5,1:5]
```

```
##         Cell_1 Cell_2 Cell_3 Cell_4 Cell_5
## A1BG         0      0      0      0      1
## A1CF         0      0      0      0      0
## A2M          1      0      1      2      0
## A2ML1        0      1      0      0      0
## A3GALT2      0      0      0      0      0
```

4.2 Altering parameters

While many of the parameters can be estimated from the data and set based on a specific protocol, it may be of interest to compare how changing a parameter affects downstream properties of the data. To do this, a base simulation should be run so that the initial mRNA counts are generated and all parameters are pre-estimated.

Below we demonstrate code to simulate a baseline and a single parameter change:

```
# Baseline simulation
scaffoldParams <- estimateScaffoldParameters(sce)
sce_sim <- simulateScaffold(scaffoldParams, sce)
```

Now we want to examine the effect of only a single parameter, e.g. equalization amount. The previous output contains metadata and cell-specific data that can be passed to new simulations. The metadata object contains the simulated initial mRNA counts, this ensures all simulations are on the same starting population. The capture efficiency is estimated within simulateScaffold and output as column data. It is accessed via `colData()$capEfficiency`.

```
# Some estimates are finalized in simulateScaffold so we want to pull those out:
scaffoldParams@captureEfficiency <- colData(sce_sim)$capEfficiency

# Change one parameter of interest
print(scaffoldParams@equalizationAmount) # Previous setting
scaffoldParams@equalizationAmount <- 0 # Update setting
sce_sim_eq <- simulateScaffold(scaffoldParams, sce,
                               inputInitial = sce_sim@metadata$initialSimCounts)
```

4.3 Simulate multiple populations

Scaffold can simulate multiple populations through the `numCells` and `usePops` parameters in the `estimateScaffoldParameters` function. First, specify a vector with the number of cells for each population in `numCells`. The parameter `usePops` is a list object and should have the following elements:

- `propGenes` - The proportion of genes having distinct expression compared to the first (reference) population. The first population will be simulated based only on the input dataset and estimated/set parameters.
- `fc_mean` - The average fold-change for expression differences compared to the first population.
- `fc_sd` - Standard deviation of fold-change for expression differences compared to the first population.

Fold-changes are simulated from a Normal(`fc_mean`, `fc_sd`) and the direction is chosen at random. These three elements are vectors with length equal to the number of populations. The first value for each of these elements should be zero to indicate no changes should be made to the reference population.

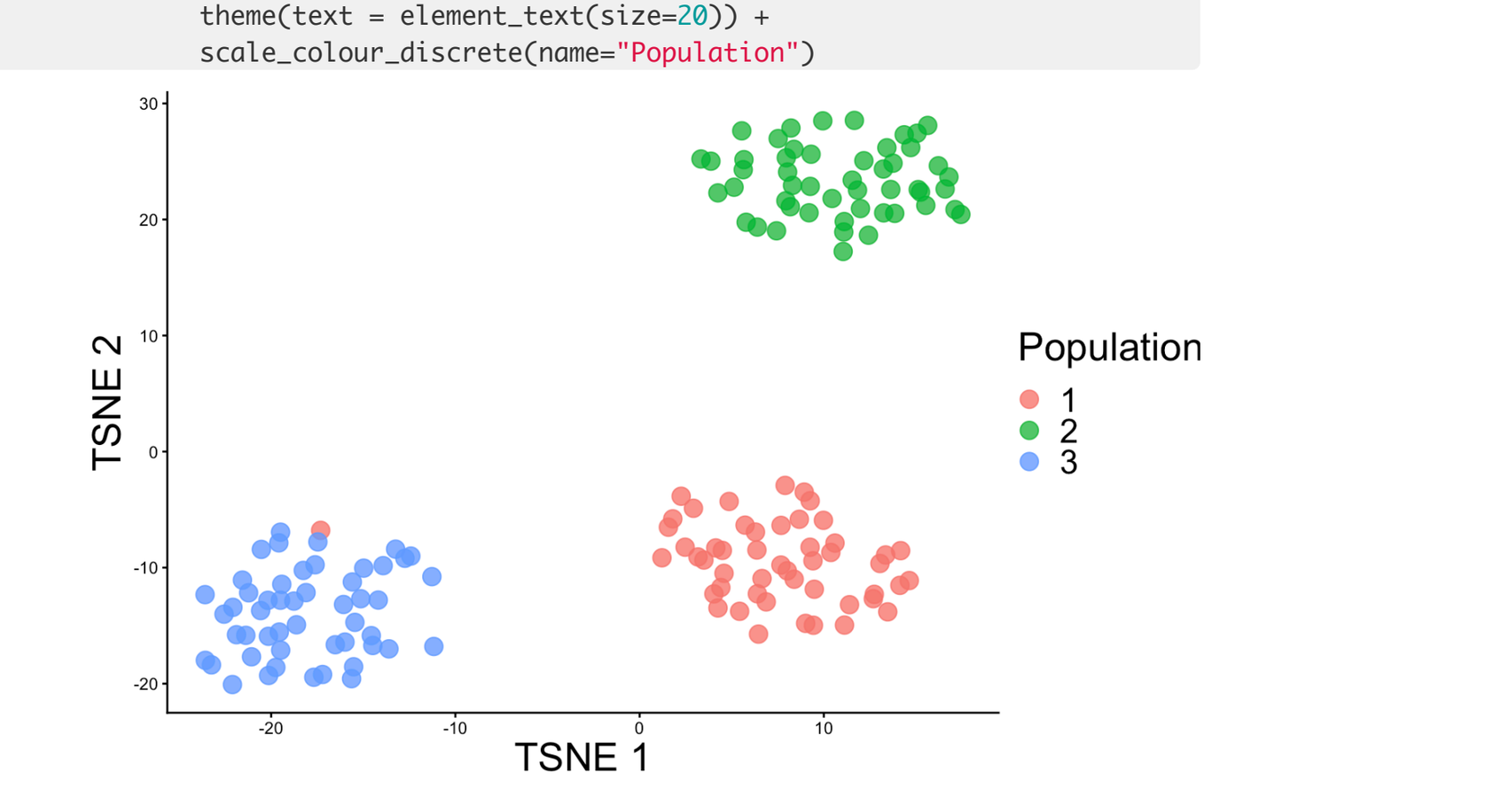
Below is an example simulating three cell populations each having 50 cells. Setting `popHet` ensures that we are simulating a homogeneous population as the reference, this is not required but just for demonstration.

```
scaffoldParams <- estimateScaffoldParameters(sce, numCells = c(50,50,50),
                                  popHet = c(1,1),
                                  usePops = list(propGenes = c(0, .6, .4),
                                                 fc_mean = c(0, 2, 1.5),
                                                 fc_sd = c(0, .4, .4)))
multipop.sce <- simulateScaffold(scaffoldParams, sce)
```

```
## [1] "Estimating capture efficiency..."
## [1] "Finished estimating capture efficiency!"
## [1] "Starting capture step (lysis and reverse transcription)..."
## [1] "Finished capture step!"
## [1] "Starting preamplify step..."
## [1] "Finished preamplify!"
## [1] "Starting library prep and sequencing..."
## [1] "Beginning formatting output..."
## [1] "Processing output for 150 cells."
## [1] "Finished sequencing and data formatting!"
```

Below we can visualize the three distinct populations in a t-SNE plot using the `scater` R package:

```
if (!requireNamespace("scater", quietly = TRUE))
  BiocManager::install("scater")
library(scater)


multipop.sce <- logNormCounts(multipop.sce)
multipop.sce <- runPCA(multipop.sce, name="PCA",
                       ncomponents=15)
multipop.sce <- runTSNE(multipop.sce, perplexity=10,
                        dimred="PCA", n_dimred=10)
plotReducedDim(multipop.sce, dimred = "TSNE", colour_by = "cellPopulation",
               point_alpha = .8, point_size=4) +
             theme(text = element_text(size=20)) +
             scale_colour_discrete(name="Population")
```



4.4 Simulate continuous populations

Scaffold can additionally simulate continuous cell populations or those having dyanmically expressed genes (e.g. cell development/differentiation) using the `useDynamic` parameter. We assume some proportion of genes drive the dynamic process and we simulate their initial mRNA counts via a B-spline. The parameter `useDynamic` is a list object and should have the following elements:

- `propGenes` - The proportion of genes to be simulated with dynamic expression.
- `degree` - The degree of the B-spline (default is 2).
- `knots` - Knots for the B-spline or 'locations' of major change along the continuous path of cells. The default is two knots and the locations are drawn from Uniform(0, .5) and Uniform(.5,1). The knot values must be within (0,1).
- `theta` - the directional changes between the edges and knots, this is a vector of length equal to knots+degree+1. The default is to generate all directions from Normal(5, 5).

User-specified values for `knots` and `theta` must be matrices as these are different for each gene.

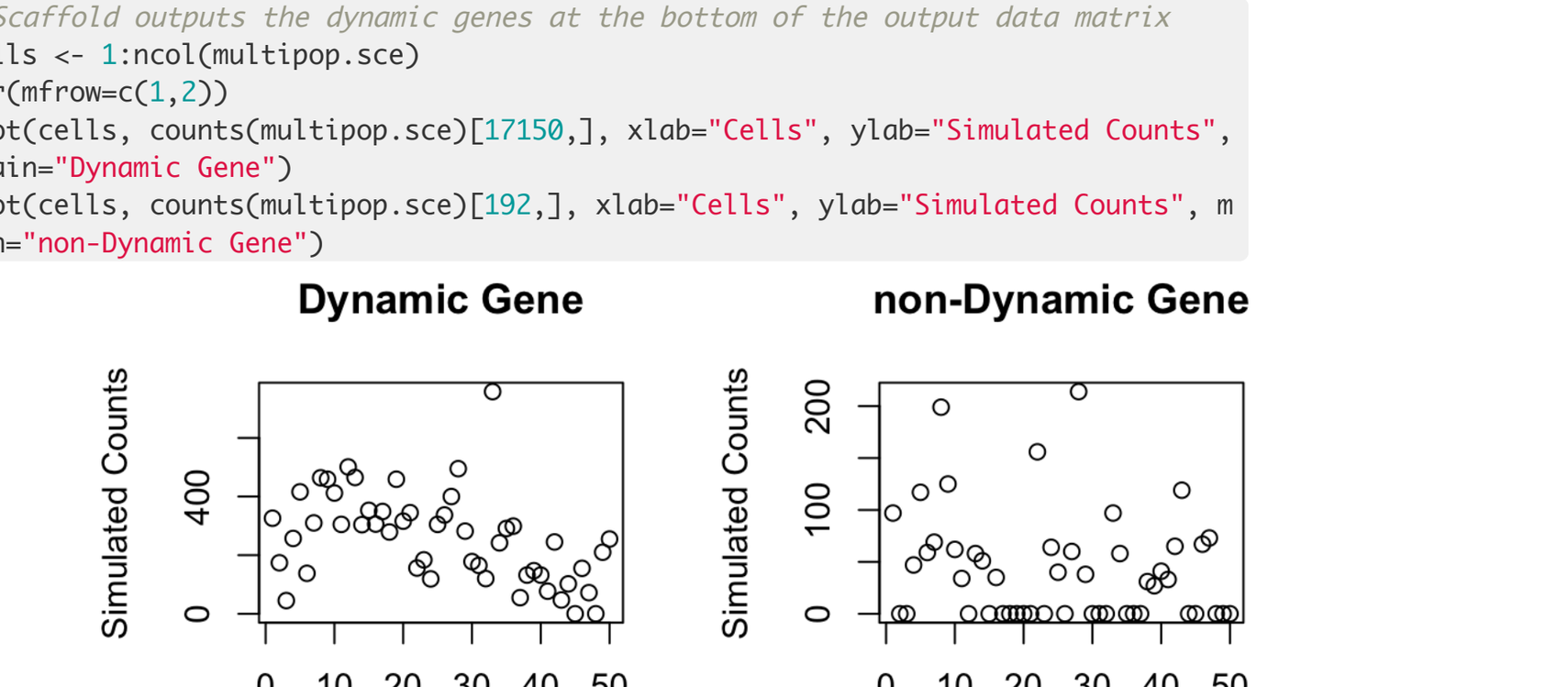
```
set.seed(14)
RcppZiggurat::zsetseed(14)
```

```
# Say we want 15% of genes dynamic
ngenes <- ceiling(.15 * nrow(sce))

myknots <- matrix(runif(2*ngenes, 0, 1), ncol=2, nrow=ngenes)
mytheta <- matrix(rnorm(5, 5, 5), ncol=5, nrow=ngenes)

scaffoldParams <- estimateScaffoldParameters(sce, numCells = 50,
                                  popHet = c(1,1),
                                  useDynamic = list(propGenes = .15,
                                                    degree = 2,
                                                    knots = myknots,
                                                    theta = mytheta))
multipop.sce <- simulateScaffold(scaffoldParams, sce)
```

```
## [1] "Estimating capture efficiency..."
## [1] "Finished estimating capture efficiency!"
## [1] "Starting capture step (lysis and reverse transcription)..."
## [1] "Finished capture step!"
## [1] "Starting preamplify step..."
## [1] "Finished preamplify!"
## [1] "Starting library prep and sequencing..."
## [1] "Beginning formatting output..."
## [1] "Processing output for 50 cells."
## [1] "Finished sequencing and data formatting!"
```

Let's plot one of the genes that we simulated to be dynamic and a non-dynamic gene:

```
# Scaffold outputs the dynamic genes at the bottom of the output data matrix
cells <- 1:ncol(multipop.sce)
par(mfrow=c(1,2))
plot(cells, counts(multipop.sce)[17150,], xlab="Cells", ylab="Simulated Counts",
 main="Dynamic Gene")
plot(cells, counts(multipop.sce)[192,], xlab="Cells", ylab="Simulated Counts", m
ain="non-Dynamic Gene")
```



4.5 Simulating large cell populations

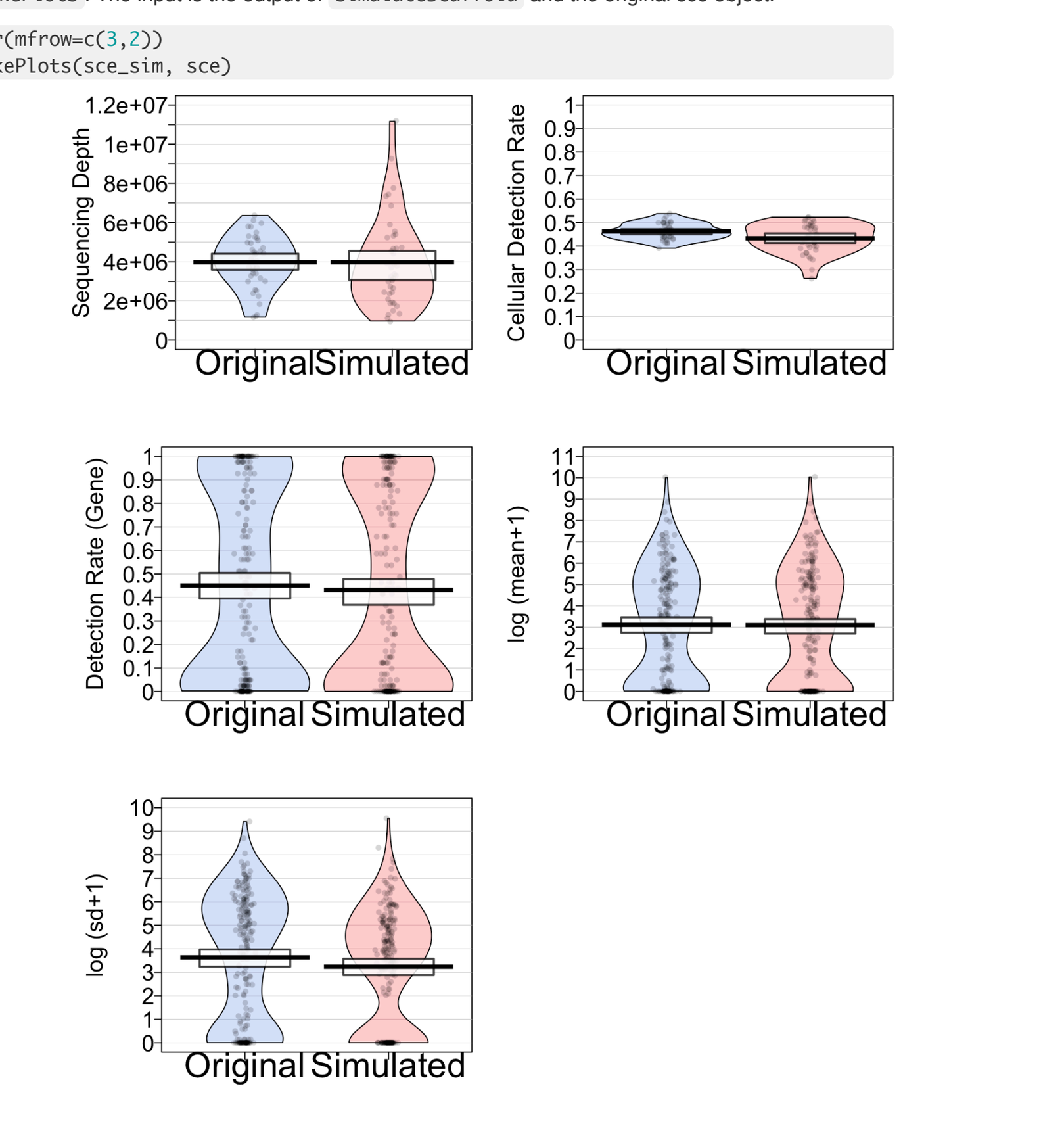
To simulate more than 5k cells, it's best to run Scaffold in batches especially when using UMIs as tracking the unique transcripts is memory intensive. Similar to the section for altering parameters, we can similarly estimate the baseline simulation parameters and then batches can be generated consecutively or in parallel (e.g. across high throughput compute cores).

```
# Baseline simulation
scaffoldParams <- estimateScaffoldParameters(sce)
sce_sim <- simulateScaffold(scaffoldParams, sce)

# To keep all parameters the same, pull out the capture efficiencies and the ini
tial mRNA counts.
## Then this can be used to cells e.g 2k at a time.
scaffoldParams@captureEfficiency <- colData(sce_sim)$capEfficiency
scaffoldParams@numCells <- 2000
sce_sim_eq <- simulateScaffold(scaffoldParams, sce,
                               inputInitial = sce_sim@metadata$initialSimCounts)
```

5 Summary plots of simulated data

`scaffold` offers a function for generating summary figures of the simulation results called `makePlots`. The input is the output of `simulateScaffold` and the original sce object.

```
par(mfrow=c(3,2))
makePlots(sce_sim, sce)
```



6 Table of simulation parameters

Parameter	Usage/Default	Notes/Tips
Protocol to simulate from	C1/Smart-seq (default), 10X, droplet	Droplet and 10X have the same default settings.
Whether to use UMI	True, False (default)	This is automatically set to TRUE for 10X/droplet protocols.
Number of cells	Estimated from input data	
Number of genes	Estimated from input data	
Gene means	Estimated from input data	
Total transcripts	Assumes 300,000 per cell	
Degree of heterogeneity	Estimated from input data	A cell-specific scale factor simulate underlying cell population heterogeneity.
Capture efficiency	Estimated from input data	
Number of pre-amplification cycles	For C1/Smart-seq default is 18	Usually this is specified in the experimental protocol
Number of amplification cycles	Default is 12 for all protocols	Usually this is specified in the experimental protocol
Pre-amplification efficiency	Generated from N(0.95, 0.02) for C1/Smart-seq data	This step is not used in 10X/droplet protocols.
Amplification efficiency	Generated from N(0.95, 0.02) for all protocols	Cell-specific values for C1/Fluidigm and only one value for 10X/droplet
Equalization amount	Value in [0,1]	0 (complete equalization), 1 (no equalization; default), any other value is partial equalization, see manuscript Methods for details.
Tagmentation efficiency	Generated from N(0.95, 0.02) for all protocols	Cell-specific values for C1/Fluidigm and only one value for 10X/droplet
Total sequencing depth	Default is the sum of all counts in the input data	

7 Frequently Asked Questions

- How long does Scaffold take to run?

Below are time trials for different protocol settings and number of cells. For UMI protocols, tracking the unique molecules is memory intensive and presents a limitation. We have included a section on how to generate larger numbers of cells in Section 2.5.

Number of Cells	Runtime (minutes)	Protocol
100	0.14	C1, non-UMI
1000	1.55	C1, non-UMI
5000	9.41	C1, non-UMI
1000	3.73	UMI/10X
5000	24.09	UMI/10X

8 Session info

Here is the output of `sessionInfo` on the system on which this document was compiled:

```
sessionInfo()
```

```
## R version 4.0.4 (2021-02-15)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Mojave 10.14.6
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.d
ylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/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] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] scater_1.18.6               ggplot2_3.3.3
##  [3] SingleCellExperiment_1.12.0 SummarizedExperiment_1.20.0
##  [5] Biobase_2.50.0              GenomicRanges_1.42.0
##  [7] GenomeInfoDb_1.26.7         IRanges_2.24.1
##  [9] S4Vectors_0.28.1            BiocGenerics_0.36.1
## [11] MatrixGenerics_1.2.1        matrixStats_0.58.0
## [13] scaffold_0.2.0              BiocStyle_2.18.1
##
## loaded via a namespace (and not attached):
##  [1] bitops_1.0-7              tools_4.0.4
##  [3] bslib_0.2.5.1             utf8_1.2.1
##  [5] R6_2.5.0                  irlba_2.3.3
##  [7] vipor_0.4.5               DBI_1.1.1
##  [9] colorspace_2.0-1          withr_2.4.2
## [11] tidyselect_1.1.1          gridExtra_2.3
## [13] compiler_4.0.4            iotools_0.3-1
## [15] BiocNeighbors_1.8.2       logging_0.10-108
## [17] DelayedArray_0.16.3       labeling_0.4.2
## [19] bookdown_0.22             sass_0.4.0
## [21] scales_1.1.1              mvtnorm_1.1-1
## [23] pbapply_1.4-3             stringr_1.4.0
## [25] digest_0.6.27             rmarkdown_2.8
## [27] XVector_0.30.0            jpeg_0.1-8.1
## [29] wrswoR_1.1.1              pkgconfig_2.0.3
## [31] htmltools_0.5.1.1         sparseMatrixStats_1.2.1
## [33] highr_0.9                 GlobalOptions_0.1.2
## [35] rlang_0.4.11              DelayedMatrixStats_1.12.3
## [37] shape_1.4.6               farver_2.1.0
## [39] jquerylib_0.1.4           generics_0.1.0
## [41] jsonlite_1.7.2            gtools_3.8.2
## [43] BiocParallel_1.24.1       dplyr_1.0.6
## [45] RCurl_1.98-1.3            magrittr_2.0.1
## [47] BiocSingular_1.6.0        GenomeInfoDbData_1.2.3
## [49] scuttle_1.0.4             Matrix_1.3-3
## [51] Rcpp_1.0.6                ggbeeswarm_0.6.0
## [53] munsell_0.5.0             fansi_0.4.2
## [55] viridis_0.6.1             lifecycle_1.0.0
## [57] RcppZiggurat_0.1.6        stringi_1.6.2
## [59] yaml_2.2.1                zlibbioc_1.36.0
## [61] Rtsne_0.15                grid_4.0.4
## [63] crayon_1.4.1              lattice_0.20-44
## [65] cowplot_1.1.1             beachmat_2.6.4
## [67] splines_4.0.4             circlize_0.4.12
## [69] magick_2.7.2              knitr_1.33
## [71] pillar_1.6.1              glue_1.4.2
## [73] evaluate_0.14             data.table_1.14.0
## [75] BiocManager_1.30.15       vctrs_0.3.8
## [77] MatrixModels_0.5-0        BayesFactor_0.9.12-4.2
## [79] gtable_0.3.0              purrr_0.3.4
## [81] assertthat_0.2.1          xfun_0.23
## [83] rsvd_1.0.5                Rfast_2.0.3
## [85] coda_0.19-4               viridisLite_0.4.0
## [87] tibble_3.1.2              yarrr_0.1.5
## [89] beeswarm_0.3.1            ellipsis_0.3.2
```