

# Bioconductor workflow for single-cell RNA sequencing data analysis: dimensionality reduction, clustering, and pseudotime ordering

Author Name1<sup>1</sup> and Author Name2<sup>2</sup>

<sup>1</sup>Address of author1

<sup>2</sup>Address of author2

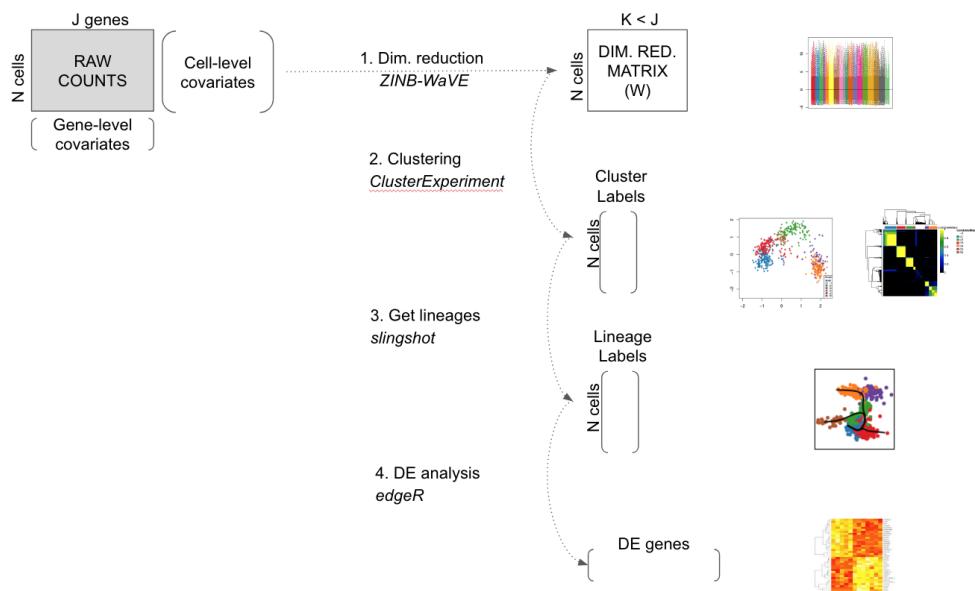
---

**Abstract** Abstracts should be up to 300 words and provide a succinct summary of the article. Although the abstract should explain why the article might be interesting, care should be taken not to inappropriately over-emphasise the importance of the work described in the article. Citations should not be used in the abstract, and the use of abbreviations should be minimized.

---

## Keywords

single-cell, workflow, single-cell-genomics, bioconductor, rna-seq, gene-expression, statistical-models, r



**Figure 1.** Workflow for analyzing scRNA-seq datasets. On the right side, major plots generated by the workflow.

EAP: small request. Can everyone put a line between the beginning of a r chunk and text? It makes it nicely formated for my text editor.

## Introduction

Single-cell RNA sequencing (scRNA-seq) is a powerful and promising class of high-throughput assays that enables researchers to measure genome-wide transcription levels for individual cells. To properly account for features specific to scRNA-seq, such as zero inflation and high levels of technical noise, several novel statistical methods have been developed to tackle different problems like dimensionality reduction, clustering, and the inference of cell lineages. While each individual method is useful to analyze scRNA-seq data, there is an increasing need for creating integrated workflows of these new statistical tools. This is all the more true that novel sequencing technologies are able to sequence more and more cells in a single sequencing run. For example, the Chromium Single Cell 3' Solution was recently used to sequence and profile about 1.3 million cells from embryonic mice brains.

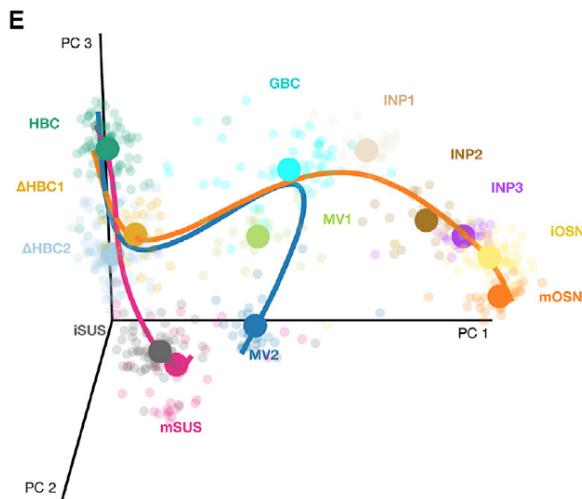
Some workflows have already been developed to analyse scRNA-seq data with several useful methods for quality control, visualisation and pre-processing of the data. The workflow for low-level analyses of scRNA-seq data described in (A. T. Lun, McCarthy, and Marioni 2016) or the package *scater* (McCarthy et al. 2017) are examples of workflows based on software packages from the open-source Bioconductor project (Huber et al. 2015) and cover basic steps including quality control, data exploration and normalization. In these workflows, single-cell expression data are organized in objects of the *SCESet* class allowing integrated analysis. However, these workflows are mostly used to prepare the data for further downstream analysis and do not focus on steps like clustering and pseudotime ordering.

Here, we propose an integrated workflow with downstream analysis where our workflow has three main steps: dimensionality reduction adjusting for gene and cell-level covariates, robust and stable clustering using resampling-based sequential ensemble clustering, and inference of branching lineages and ordering of the cells by developmental progression. Along the workflow, we use a unique *SummarizedExperiment* object to create an easy to use sequence of steps where for each steps a different R package is used.

## Analysis of olfactory stem cell lineages using scRNA-seq data

### Overview

The workflow is illustrated using data from a scRNA-seq study of stem cell differentiation in the mouse olfactory epithelium (Fletcher et al. 2017). The olfactory epithelium contains mature olfactory sensory neurons that are continuously renewed in the epithelium via neurogenesis through differentiation of globose basal cells (GBCs) which are the actively proliferating cells in the epithelium. When a severe injury to the entire tissue happens, the olfactory epithelium can regenerate from normally quiescent stem cells called horizontal basal cells (HBCs) which become activated to differentiate and reconstitute all major cell types in the epithelium. The dataset we work with in this workflow was generated to study the differentiation from the HBCs stem cells to different cell types present in the epithelium. To map the developmental trajectories of the multiple cell lineages arising from the olfactory epithelium's HBC stem cell, a scRNA-seq analysis was performed on FACS-purified cells using the Fluidigm C1 microfluidics cell capture platform followed by Illumina sequencing.



**Figure 2.** Stem cell differentiation in the mouse olfactory epithelium. This figure has been reproduced with kind permission from [@Fletcher2017].

Then, the expression of each gene was quantified by counting the total number of reads mapped to each cell to get a cell by gene matrix of raw counts. Different lineages were assigned to each cell in the dataset using a set of statistical tools. Finally, results were confirmed experimentally using *in vivo* lineage tracing. Details on the data generation and steps of the analysis are available in (Fletcher et al. 2017).

It was found that the first major bifurcation in the HBC lineage trajectory occurs prior to cell division, producing either mature sustentacular (mSUS) cells or GBCs. Then, the GBC lineage, in turn, branches to give rise to mature olfactory sensory neurons (mOSN), microvillous (MV) cells, and cells of the Bowman's gland. See Figure 2. In this study, we show possible steps to recover the lineages found in the original study from the matrix of raw counts publicly available.

FP: ask Russell if we can use the Figure 2 here.

### Pre-processing

Counts for all genes in each cell were obtained from NCBI Gene Expression Omnibus (GEO) with accession number GSE95601. Before filtering, the dataset has 849 cells and 28361 detected genes.

```
load("../data/GSE95601_oeHBCdiff_Cufflinks_eSet.Rda")

# count matrix
E <- assayData(Cufflinks_eSet)$counts_table

# Remove undetected genes
E <- na.omit(E)
E <- E[rowSums(E)>0,]
dim(E)

## [1] 28361    849
```

We removed the genes corresponding to ERCC spike-ins and the CreER gene as it corresponds to a marker gene for the HBCs cells. Davide, is that correct?

```
# remove ERCC and CreER genes
cre <- E["CreER",]
ercc <- E[grep("ERCC-", rownames(E)),]
E <- E[grep("ERCC-", rownames(E), invert = TRUE),]
E <- E[-which(rownames(E)=="CreER"),]
dim(E)

## [1] 28284    849
```

Along the workflow, we use a SummarizedExperiment object to keep the counts and metadata in a unique object. In the metadata, we dispose of quality control measures for each cell, information on the batches in which the cells were sequenced, and clusters and lineages found in the original publication (Fletcher et al. 2017). Cells with a cluster of -2 are cells not assigned to a cluster in the original publication.

```

# Extract QC metrics
qc <- as.matrix(protocolData(Cufflinks_eSet)$data)[,c(1:5, 10:18)]
qc <- cbind(qc, CreER = cre, ERCC_reads = colSums(ercc))

# Extract metadata
batch <- droplevels(pData(Cufflinks_eSet)$MD_c1_run_id)
bio <- droplevels(pData(Cufflinks_eSet)$MD_expt_condition)
clusterLabels <- read.table("../data/oeHBCdiff_clusterLabels.txt",
                           sep = "\t", stringsAsFactors = FALSE)
m <- match(colnames(E), clusterLabels[, 1])

# Create metadata data.frame
metadata <- data.frame("Experiment" = bio,
                       "Batch" = batch,
                       "clusterLabels" = clusterLabels[m, 2],
                       qc)

# symbol for cell not assigned to a lineage in original data
metadata$clusterLabels[is.na(metadata$clusterLabels)] <- -2

se <- SummarizedExperiment(assays = list(counts = E),
                           colData = metadata)
se

## class: SummarizedExperiment
## dim: 28284 849
## metadata(0):
## assays(1): counts
## rownames(28284): Xkr4 LOC102640625 ... Ggcx.1 eGFP
## rowData names(0):
## colnames(849): OEP01_N706_S501 OEP01_N701_S501 ... OEL23_N704_S503
##   OEL23_N703_S502
## colData names(19): Experiment Batch ... CreER ERCC_reads

```

Using bioconductor package scone, we removed low quality cells that did not pass our quality control rule, that is ... See Figure 3. Davide, would you like to describe your choices for scone here?

```

# Metric-based Filtering sample-filtering
data("housekeeping")
hk = rownames(se)[toupper(rownames(se)) %in% housekeeping$V1]

mfilt <- metric_sample_filter(assay(se), nreads = colData(se)$NREADS,
                             ralign = colData(se)$RALIGN,
                             pos_controls = rownames(se) %in% hk,
                             zcut = 3, mixture = FALSE,
                             plot = TRUE)

# Simplify to a single logical
mfilt <- !apply(simplify2array(mfilt[!is.na(mfilt)]), 1, any)
se <- se[, mfilt]
dim(se)

## [1] 28284    747

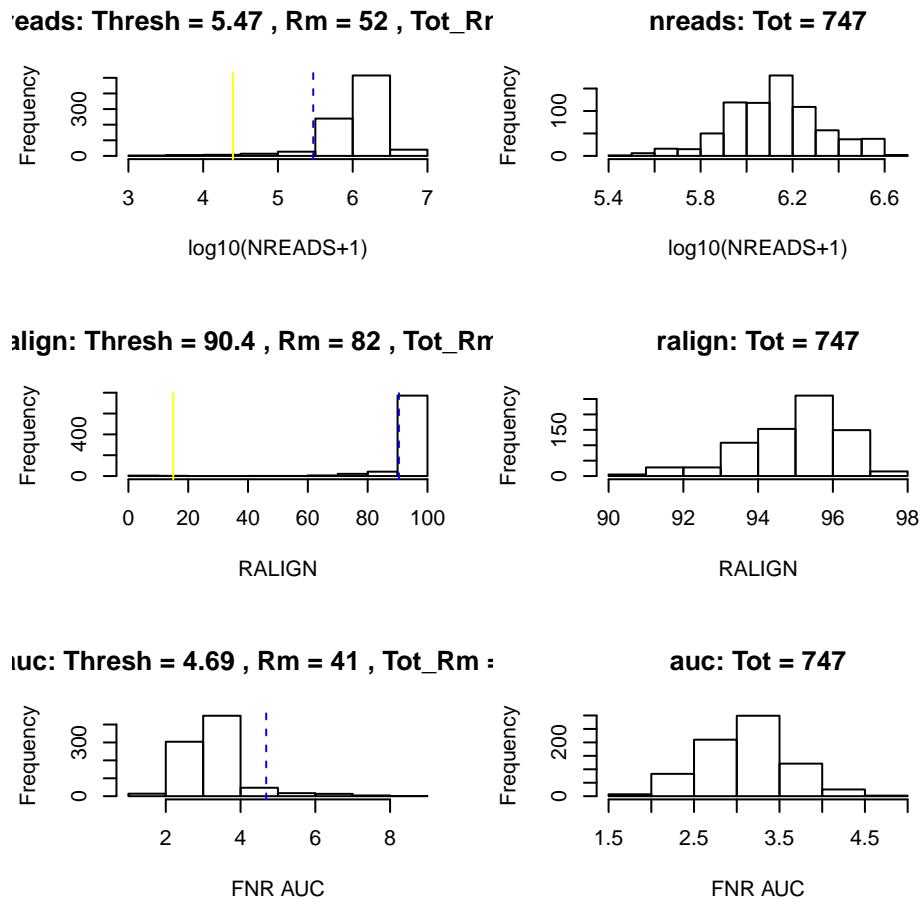
```

Finally, to speed up the computations, we kept only the 1000 most variable genes. We recommend not to be so stringent in your analysis and perform the analysis on more genes.

```

# filtering to top 1000 most variable
vars <- rowVars(log1p(assay(se)))
names(vars) <- rownames(se)
vars <- sort(vars, decreasing = TRUE)
core <- se[names(vars)[1:1000],]
save(core, file="../data/oe_se_1000Var.rda")

```



**Figure 3.** Plot generated by scone to filter low quality cells.

### Dataset structure

Overall, after the preprocessing steps, our dataset has 1000 genes and 747 cells.

```
core

## class: SummarizedExperiment
## dim: 1000 747
## metadata(0):
## assays(1): counts
## rownames(1000): Cbr2 Cyp2f2 ... Rnf13 Atp7b
## rowData names(0):
## colnames(747): OEP01_N706_S501 OEP01_N701_S501 ... OEL23_N704_S503
##   OEL23_N703_S502
## colData names(19): Experiment Batch ... CreER ERCC_reads
```

Metadata for the cells is stored in the object colData from the SummarizedExperiment object. Cells have been processed in 18 different batches.

```
batch <- colData(core)$Batch
col_batch = c(brewer.pal(9, "Set1"), brewer.pal(8, "Dark2"),
             brewer.pal(8, "Accent")[1])
names(col_batch) = unique(batch)
table(batch)
```

	GBC08A	GBC08B	GBC09A	GBC09B	P01	P02	P03A	P03B	P04	P05
##	39	40	35	22	31	48	51	40	20	23
##	P06	P10	P11	P12	P13	P14	Y01	Y04		
##	51	40	50	50	60	47	58	42		

In original work (Fletcher et al. 2017), cells have been clustered into 14 different clusters where 151 cells were not assigned to a cluster.

```

clus.labels <- colData(core)[, "clusterLabels"]
col_clus <- c("transparent", brewer.pal(12, "Set3"), brewer.pal(8, "Set2"))
col_clus <- col_clus[1:length(unique(clus.labels))]
names(col_clus) <- sort(unique(clus.labels))
table(clus.labels)

## clus.labels
## -2 1 2 3 4 5 7 8 9 10 11 12 14 15
## 151 90 25 54 35 93 58 27 74 26 21 35 26 32

```

Note that some of the batches are confounded with the clusters found in the original work which could be a problem when we correct for batch effect in the dimensionality reduction step.

```
table(data.frame(batch = as.vector(batch),
                 cluster = clus.labels))
```

```

##           cluster
## batch    -2 1 2 3 4 5 7 8 9 10 11 12 14 15
## GBC08A   3 0 2 12 9 0 0 0 0 0 2 0 2 9
## GBC08B   8 0 7 5 3 0 0 0 1 2 3 0 5 6
## GBC09A   6 0 1 5 8 0 0 0 1 1 0 0 6 7
## GBC09B  12 0 2 1 3 0 0 0 1 0 0 0 0 3 0
## P01      7 0 2 4 3 15 0 0 0 0 0 0 0 0 0
## P02      5 2 0 9 3 15 3 3 2 3 0 2 1 0
## P03A     15 3 0 2 0 12 2 9 4 2 0 2 0 0
## P03B     9 1 2 1 1 11 1 2 8 1 1 2 0 0
## P04      8 0 0 0 0 9 1 0 1 1 0 0 0 0
## P05      3 0 0 0 1 11 3 0 1 0 2 2 0 0
## P06      12 1 2 3 0 8 2 4 8 4 1 2 2 2
## P10      7 3 1 4 0 3 5 8 1 0 2 5 0 1
## P11      6 2 1 1 0 1 5 1 22 3 1 6 0 1
## P12      10 0 2 0 0 4 10 0 8 2 3 6 4 1
## P13      13 1 2 4 0 4 15 0 4 5 6 1 3 2
## P14      9 0 0 1 2 0 11 0 12 2 0 7 0 3
## Y01      8 46 1 1 2 0 0 0 0 0 0 0 0 0
## Y04     10 31 0 1 0 0 0 0 0 0 0 0 0 0

```

### Dimensionality reduction adjusting for batches

In scRNA-seq analysis, dimensionality reduction is often used as a preliminary step prior to downstream analysis where clustering, pseudotime ordering, or differential expression analysis are performed. It allows the data to become more tractable, both from a statistical (cf. curse of dimensionality) and computational point of view. Additionally, technical noise can be reduced while preserving the often intrinsically low dimensional signal of interest (Dijk et al. 2017).

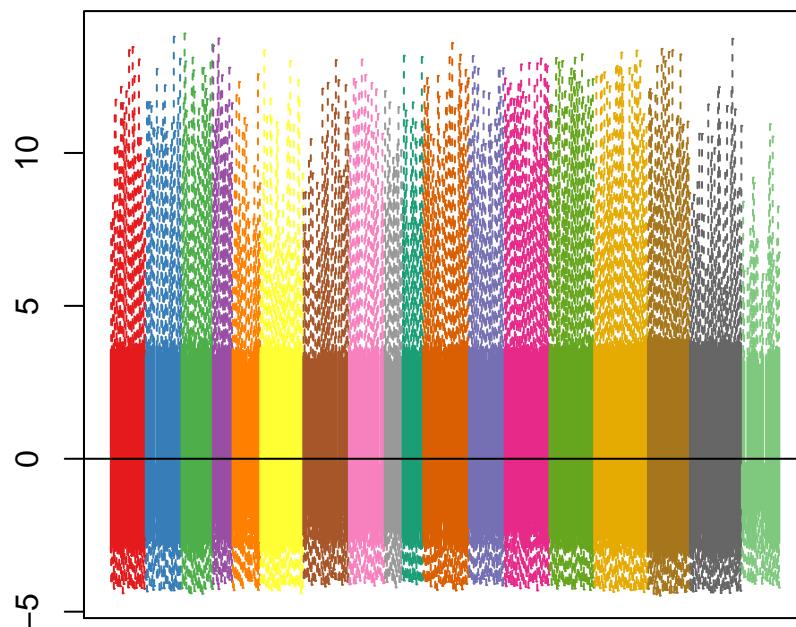
Here, we perform dimensionality reduction using bioconductor package zinbwave which allows to fit a zero-inflated negative binomial model (ZINB-WaVE) to get a low-dimensional representation of the data while accounting for zero inflation (dropouts), over-dispersion, and the count nature of the data. The model can include a sample-level intercept, which serves as a global-scaling normalization factor. The user can also include both gene-level and sample-level covariates. The inclusion of observed and unobserved sample-level covariates enables normalization for complex, non-linear effects (often referred to as batch effects), while gene-level covariates may be used to adjust for sequence composition effects, such as gene length and GC-content effects. You can read more details about the ZINB-WaVE model in (Risso et al. 2017).

As with most of the dimensionality reduction methods, the user needs to specify the number of dimensions of the new low dimensional space. Here, we use K = 50 dimensions in the low dimensional space and adjusted for the batches.

```

fn <- '../data/zinb_batch.rda'
if (runZinb & !file.exists(fn)){
  print(system.time(se <- zinbDimRed(core, K = 50, X = 'Batch',
                                         residuals = TRUE,
                                         normalizedValues = TRUE)))
  save(se, file = fn)
} else{
  load(fn)
}

```



**Figure 4.** Boxplot of normalized values. Different colors are for different batches.

DR: the chunk above and the similar one for clusterExperiment are fine for now, but in the published workflow, we should just rely on the markdown cache system (the code above doesn't check for changes to the file or code – just that a version of the file exists). FP: Yes.

### Normalized values

When function `zinbDimRed` is called, normalized values of the counts can be computed and used for visualization (e.g. in heatmaps). The normalized values are the residuals of the fit of the ZINB-WaVE model where the gene and cell covariates were specified by the user when calling function `zinbDimRed`, but no low dimensional matrix is computed. In that case, the residuals represent the counts adjusted for the gene and cell covariates and should capture the biological signal of interest. To compute the residuals, the deviance is used.

```
norm <- assays(se)$normalizedValues
if (sum(is.infinite(norm))>0){
  maxNorm = max(norm[!is.infinite(norm)])
  assays(se)$normalizedValues[is.infinite(norm)] <- maxNorm
  norm <- assays(se)$normalizedValues
}
norm[1:3,1:3]

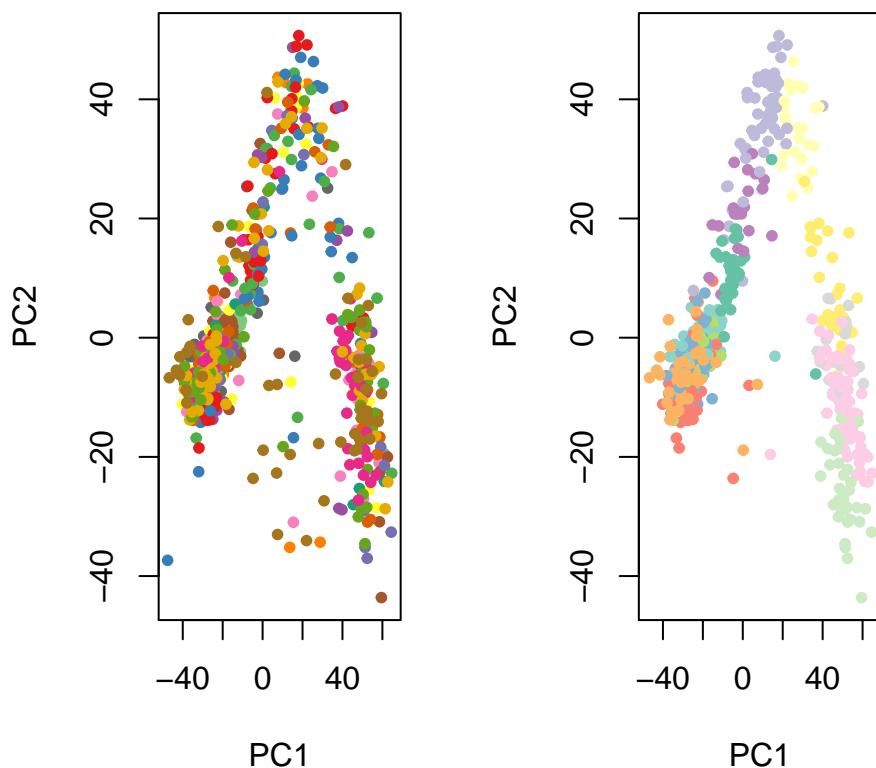
##      OEP01_N706_S501 OEP01_N701_S501 OEP01_N707_S507
## Cbr2        4.557371    4.375069   -4.142697
## Cyp2f2      4.321644    4.283266    4.090283
## Gstm1       4.796498    4.663366    4.416324
```

As expected, the normalized values are uniformly distributed for the different batches. See Figure 4.

```
norm_order <- norm[, order(as.numeric(batch))]
col_order <- col_batch[batch[order(as.numeric(batch))]]
boxplot(norm_order, col = col_order, staplewex = 0, outline = 0,
        border = col_order, xaxt = 'n')
abline(h=0)
```

The principal component analysis (PCA) of the normalized values shows that, as expected, cells do not cluster by batch but by original published clusters. See Figure 5. Overall, looking at the normalized values, it seems that the batch effect has been removed.

```
pca <- prcomp(t(norm))
par(mfrow = c(1,2))
plot(pca$x, col = col_batch[batch], pch = 20, main = '')
plot(pca$x, col = col_clus[as.character(clus.labels)], pch = 20, main = '')
```



**Figure 5.** PCA of normalized values where each point represent a cell. In the left panel, cells have been colored according to their batch. In the right panel, colors correspond to the original published clusters.

#### Dimensionality reduction

Calling function `zinbDimRed` from package `zinbwave`, we performed dimensionality reduction where the number of dimensions in the low dimensional space is  $K = 50$ . We call the matrix in the low dimensional space  $W$ . We can visualize  $W$  in two dimensions by performing multidimensional scaling (MDS) on the distance matrix of  $W$  where the euclidian distance is used. To make sure that  $W$  captured the biological signal of interest, we plotted the MDS of  $W$  with colors corresponding to the original published clusters. See Figure 6.

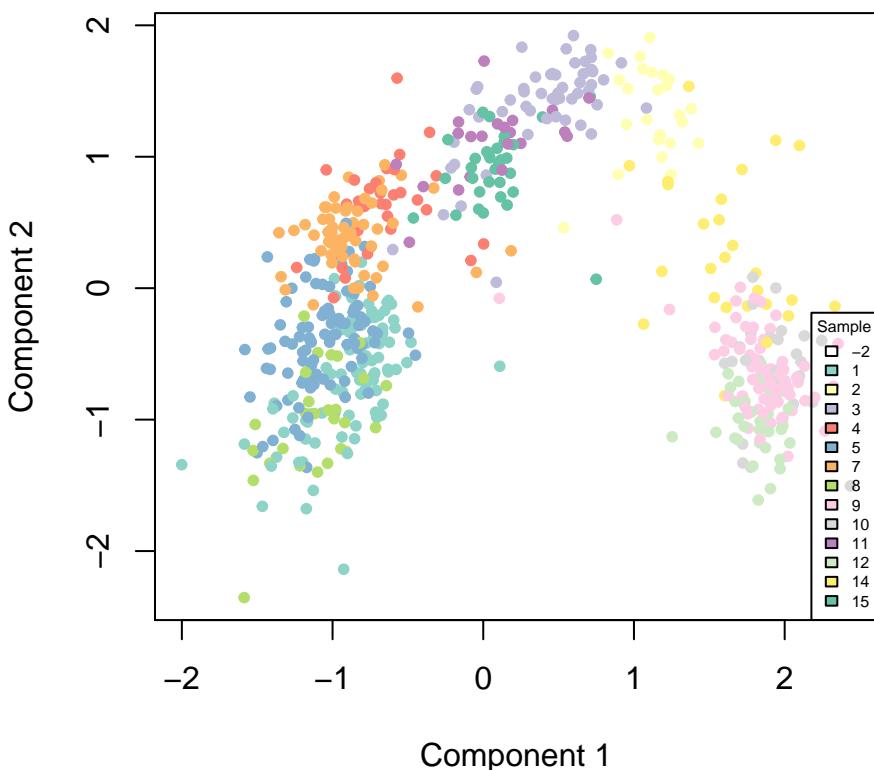
```
W <- colData(se)[, grep('`~W', colnames(colData(se)))]
W <- as.matrix(W)
d <- dist(W)
fit <- cmdscale(d, eig = TRUE, k = 2)
plot(fit$points, col = col_clus[as.character(clus.labels)], main = '',
     pch = 20, xlab = 'Component 1', ylab = 'Component 2')
legend(x = 'bottomright', legend = unique(names(col_clus)), cex = .5,
       fill = unique(col_clus), title = 'Sample')
```

#### Clustering of the cells

The next step of the workflow is to cluster the cells according to the low dimensional matrix  $W$  computed in the previous step. We used function `RSEC` from bioconductor package `clusterExperiment`. Function `RSEC` implements a resampling-based sequential ensemble clustering method for identifying stable and tight cell clusters. The method aggregates multiple clustering results obtained from different base clustering algorithms and applications of a given algorithm to resampled versions of the learning set.

Elizabeth, would you like to more details on the rational and arguments of the different functions?

```
fn <- '../data/RSEC_W_combineMinSize10.rda'
if (runClus & !file.exists(fn)){
  #symbol for samples missing from original clustering
  seObj <- SummarizedExperiment(t(W), colData = colData(core))
  print(system.time(ceObj <- RSEC(seObj, k0s = 4:15, alphas = c(0.1),
                                    betas = 0.8, dimReduce="none",
                                    clusterFunction = "hierarchical01", minSizes=1,
                                    ncores = NCORES, isCount=FALSE,
```



**Figure 6.** MDS of the low dimensional matrix ‘W’ where each point represent a cell. Cells have been colored according to the original published clusters.

```

subsampleArgs = list(resamp.num=100,
                     clusterFunction="kmeans",
                     clusterArgs=list(nstart=10)),
seqArgs = list(k.min=3, top.can=5), verbose=TRUE,
combineProportion = 0.7,
mergeMethod = "none", random.seed=424242,
combineMinSize = 10)))
save(ceObj, file = fn)
} else{
  load(fn)
}

```

We can visualize the resulting clusterings using the `plotClusters` function. See Figure 7 where cells are in the columns, and different clusterings on the rows. Each sample is color coded based on its clustering for that row, where the colors have been chosen to try to match up clusters across different clusterings that show large overlap. The first raw correspond to a consensus clustering across the many different clusterings performed.

```
plotClusters(ceObj, colPalette = c(bigPalette, rainbow(199)))
```

We can also visualize the proportion of times the consensus clusters were together across the different clusterings. See Figure 8.

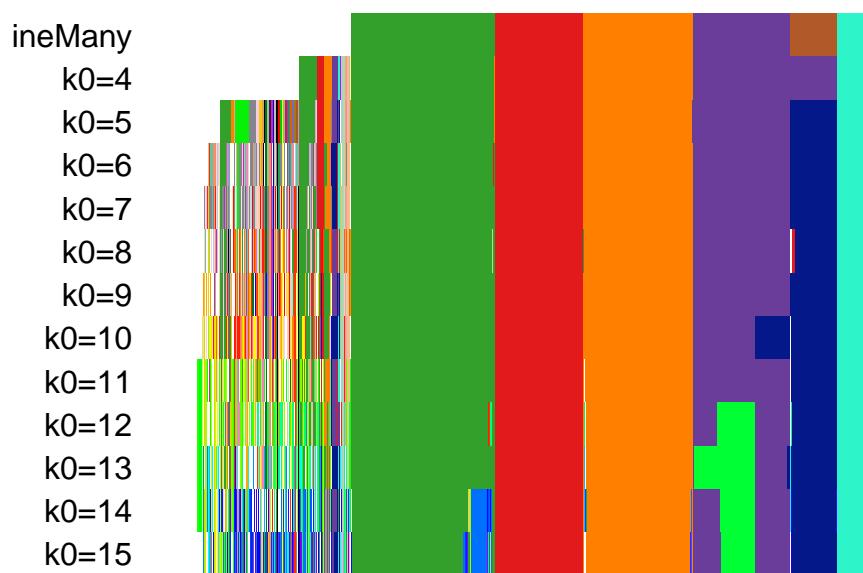
```
plotCoClustering(ceObj)
```

```
## Warning in .makeColors(clusters, colors = bigPalette): too many clusters to
## have unique color assignments
```

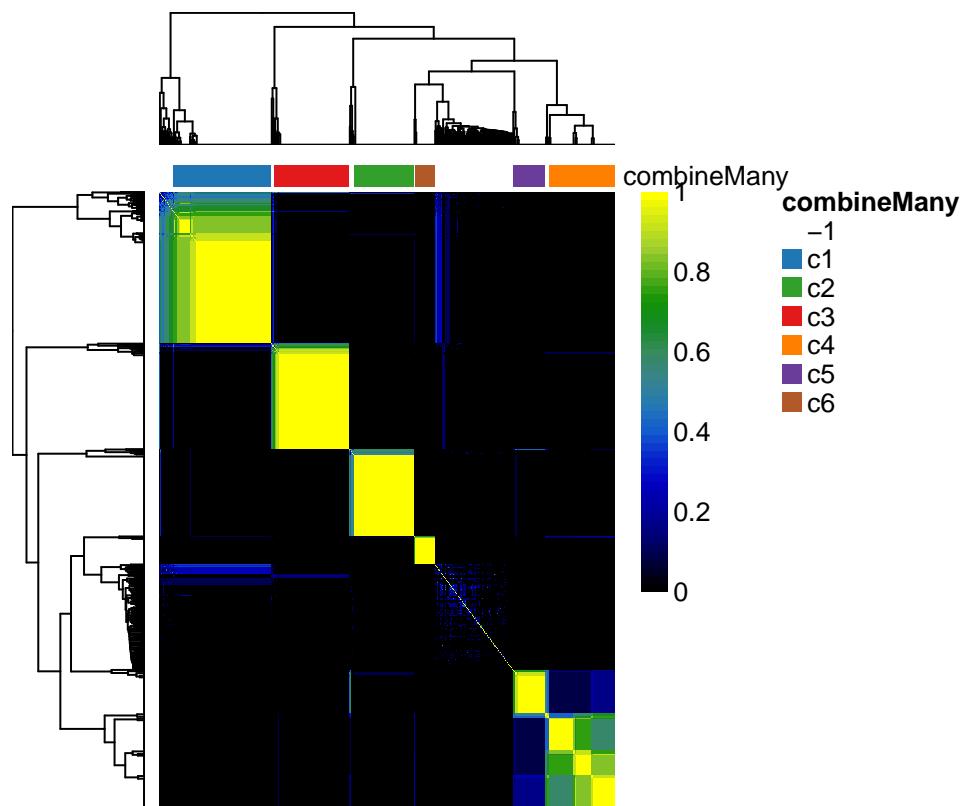
Over stable and robust clustering, the number of cells in each cluster is as follow

```
table(primaryClusterNamed(ceObj))
```

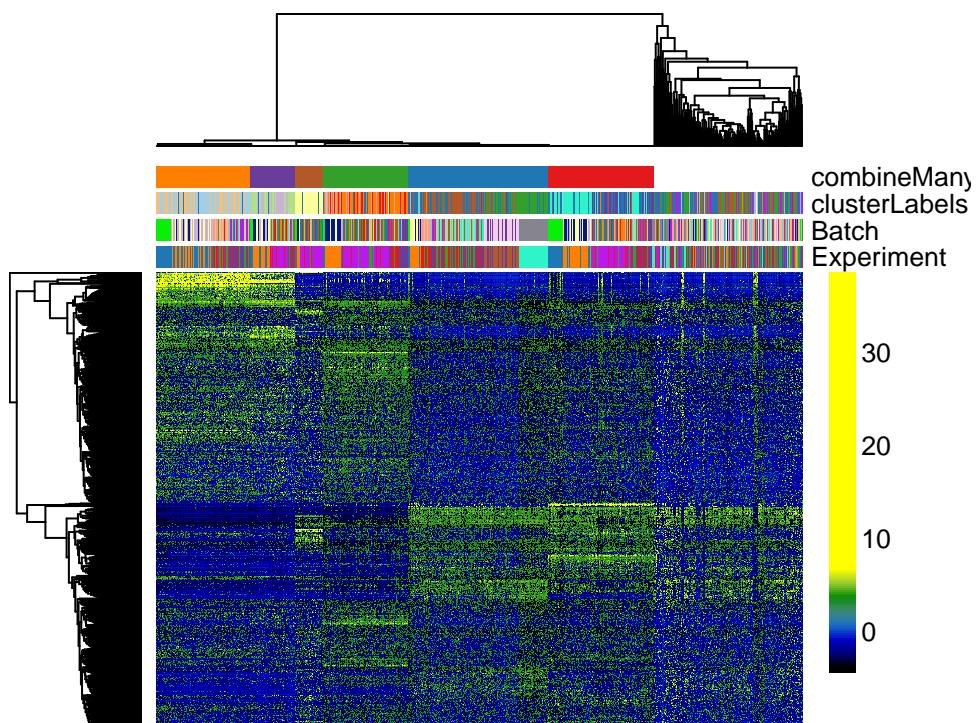
```
##
## -1  c1  c2  c3  c4  c5  c6
## 172 161  98 123 108  52  33
```



**Figure 7.** Clusters found using function RSEC from clusterExperiment package.



**Figure 8.** Heatmap of the proportion of times the clusters were together across the different clusterings.



**Figure 9.** Heatmap of the normalized values for the 1000 most variable genes where genes are in the rows and cells in the columns.

We can now plot a heatmap of the normalized values for the 1000 most variable genes where the cells have been clustered using the RSEC function. See Figure 9.

```
# set colors for cell clusterings
colData(ceObj)$clusterLabels <- as.factor(colData(ceObj)$clusterLabels)
origClusterColors <- bigPalette[1:nlevels(colData(ceObj)$clusterLabels)]
experimentColors <- bigPalette[1:nlevels(colData(ceObj)$Experiment)]
batchColors <- bigPalette[1:nlevels(colData(ceObj)$Batch)]
metaColors <- list("Experiment" = experimentColors,
                    "Batch" = batchColors,
                    "clusterLabels" = origClusterColors)

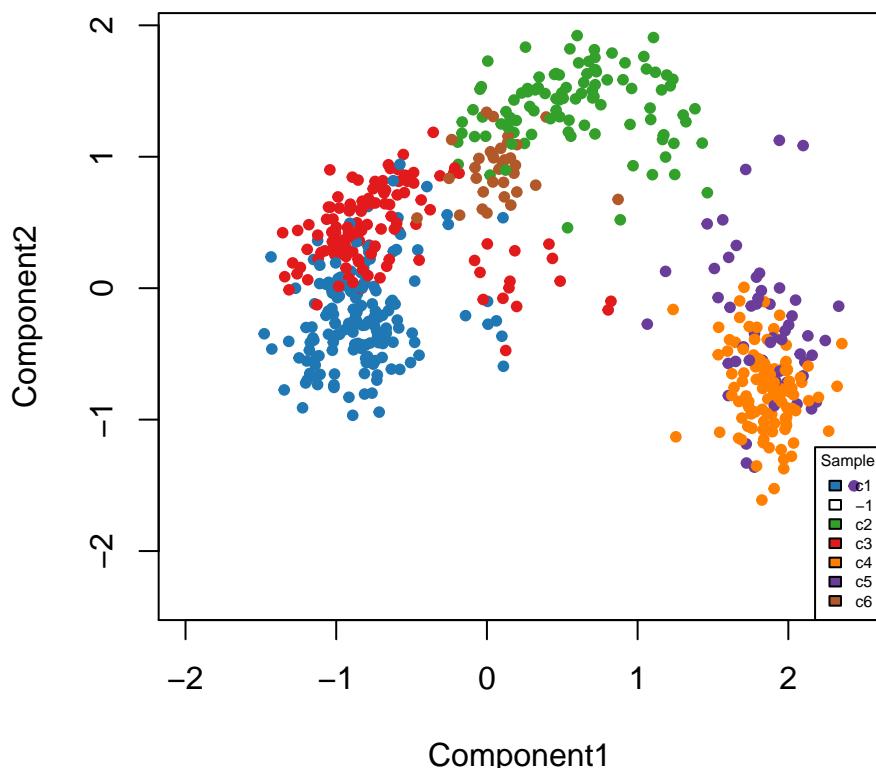
plotHeatmap(ceObj, visualizeData = assays(se)$normalizedValues,
            whichClusters = "primary", clusterFeaturesData = "all",
            clusterSamplesData = "dendrogramValue", breaks = 0.99,
            sampleData = c("clusterLabels", "Batch", "Experiment"),
            clusterLegend = metaColors, annLegend = FALSE, main = '')
```

Finally, we can visualize the cells in a two dimensional space by plotting the MDS of the low dimensional matrix  $W$  coloring the cells by their newly found clusters. See Figure 10 which is the same as Figure 6 but where colors are for the newly found clusters instead of the clusters found in the orginal published work.

```
palDF <- ceObj@clusterLegend[[1]]
pal <- palDF[, 'color']
names(pal) <- palDF[, 'name']
pal["-1"] = "transparent"
plot(fit$points, col = pal[primaryClusterNamed(ceObj)], main = '',
      pch = 20,
      xlab = 'Component1', ylab = 'Component2')
legend(x = 'bottomright', legend = names(pal), cex = .5,
       fill = pal, title = 'Sample')
```

### Pseudotime ordering

We now want to demonstrate how to use the R software package `slingshot` to infer branching lineages and order cells by developmental progression. We connect the clusters identified by RSEC with a minimum spanning tree to learn the global lineage structure. Then, we refine this structure and order cells using highly stable simultaneous principal curves to infer smooth, branching lineages.



**Figure 10.** MDS of the low dimensional matrix with  $K = 50$  dimensions. Colors correspond to the clusters found using our workflow.

From the original published work, we know that the start clusters should be the clusters with HBCs and end clusters the clusters with MV, mOSN, and mSUS cells. Additionally, we know that the clusters with the GBCs should be a junction before the differentiation between MV and mOSN cells. See Figure 2. The correspondence between the clusters we found and the original clusters is as follow

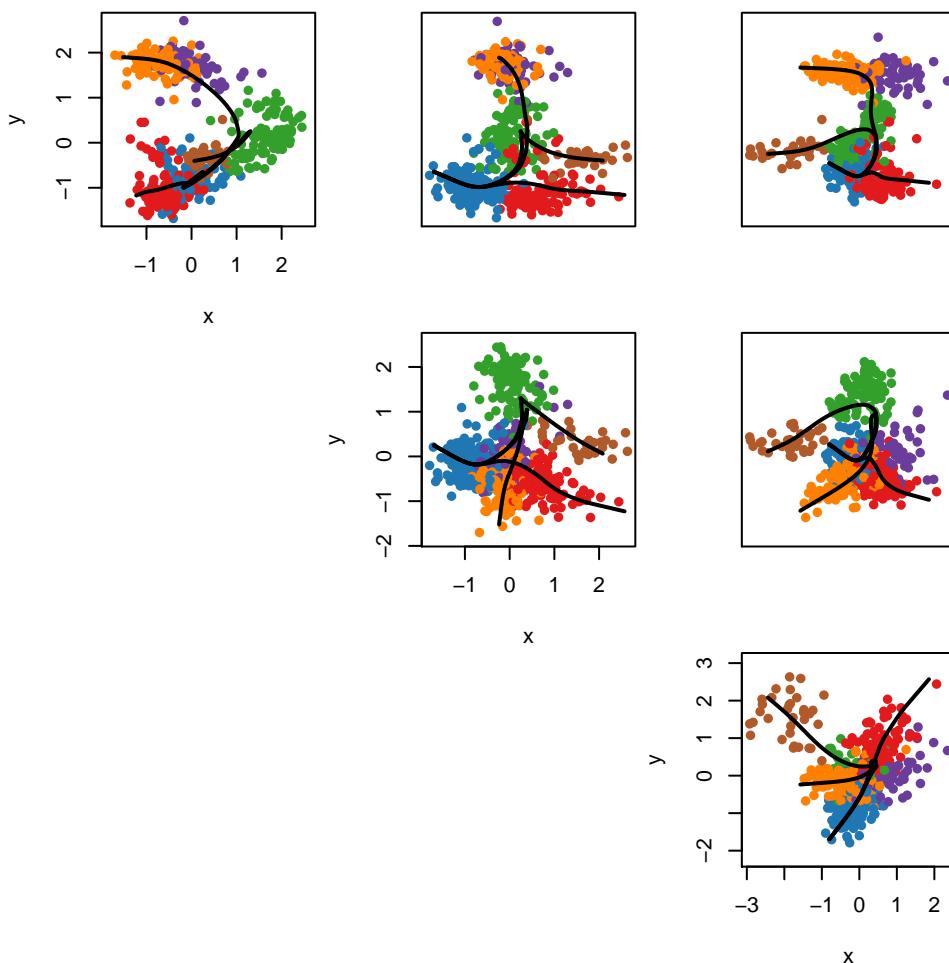
```
table(data.frame(original = clus.labels, ours = primaryClusterNamed(ce0bj)))
```

```

##          ours
## original -1 c1 c2 c3 c4 c5 c6
##      -2 51 50  6 33  5  3  3
##       1 41 49  0  0  0  0  0
##       2  1  0 24  0  0  0  0
##       3  2  2 49  1  0  0  0
##       4  1  2  0 32  0  0  0
##       5 36 55  0  2  0  0  0
##       7  3  1  0 54  0  0  0
##       8 27  0  0  0  0  0  0
##       9  2  0  1  1 68  2  0
##      10  0  0  0  0  0 26  0
##      11  3  2 16  0  0  0  0
##      12  0  0  0  0 35  0  0
##      14  4  0  1  0  0 21  0
##      15  1  0  1  0  0  0 30

```

Cluster name	Description	Color	Correspondence
c1	HBCs	blue	original 1, 5
c2	GBCs	green	original 2, 3, 11
c3	mSUS	red	original 4, 7
c4	mOSN	orange	original 9, 12
c5	Immature Neuron	purple	original 10, 14
c6	MV	brown	original 15



**Figure 11.** Cells colored by cluster in a 4-dimensional space with smooth curves representing each inferred lineage.

To order the clusters and find lineages, we use package `slingshot` where the input is the MDS of low dimensional matrix  $W$  where the number of dimensions used for MDS is  $k = 4$ . We found that slingshot algorithm is stable to the number of dimensions  $k$  of the MDS. Here, we use the unsupervised version of slingshot where we only provide the information of the start cluster but not the end clusters. Using slingshot, we recovered the order of the clusters found in the published work. See Figures 11 and 12.

Kelly, would you like to give details about slingshot algorithm?

```

our_cl <- primaryClusterNamed(ce0bj)
cl = our_cl[our_cl != "-1"]
pal = pal[names(pal) != '-1']
X <- W[our_cl != "-1", ]
mds <- cmdscale(dist(X), eig = TRUE, k = 4)
X <- mds$points

lineages <- get_lineages(X, clus.labels = cl, start.clus = "c1")
curves <- get_curves(X, clus.labels = cl, lineages = lineages)

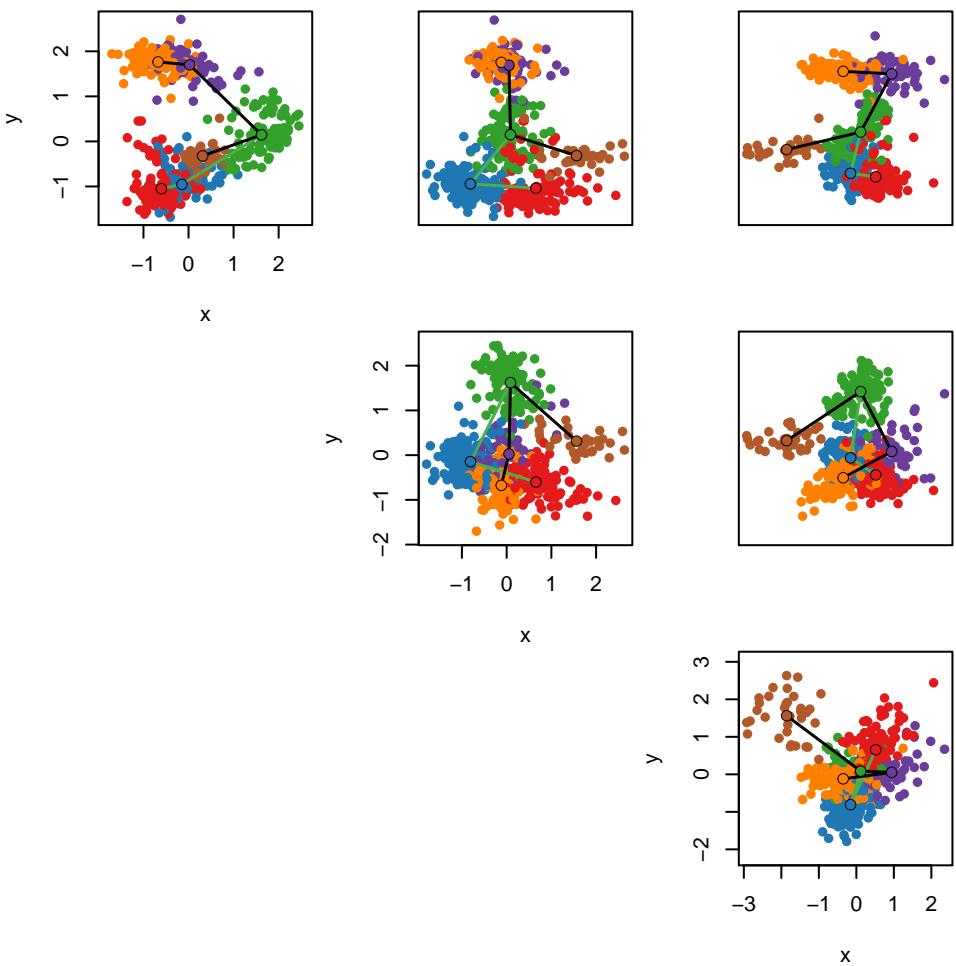
plot_curves(X, cl, curves, col.clus = pal)

plot_tree(X, cl, lineages, col.clus = pal)

print(lineages$lineage1)

## [1] "c1" "c2" "c5" "c4"

```



**Figure 12.** Cells colored by cluster in a 4-dimensional space with connecting lines between cluster centers representing the inferred structure.

```

print(lineages$lineage2)

## [1] "c1" "c2" "c6"

print(lineages$lineage3)

## [1] "c1" "c3"

```

In the workflow, we recover a reasonable pseudotime ordering of the clusters using the unsupervised version of slingshot. However, in some other cases, we have noticed that we need to give more guidance to the algorithm to find the correct ordering. Slingshot has the option for the user to provide the end cluster(s). Here is the code to use slingshot in a supervised setting

```

lineages <- get_lineages(X, clus.labels = cl, start.clus = "c1",
                           end.clus = c("c3", "c6"))
curves <- get_curves(X, clus.labels = cl, lineages = lineages)
plot_curves(X, cl, curves, col.clus = pal)
plot_tree(X, cl, lineages, col.clus = pal)

print(lineages$lineage1)
print(lineages$lineage2)
print(lineages$lineage3)

```

## DE analysis

Kelly, would you like to complete this part? Here is the kind of plots we would want to present.

```

de <- read.csv('../data/oe_markers.txt', stringsAsFactors = F, header = F)
de <- de$V1
plotHeatmap(ceObj,
            visualizeData = assays(se)$normalizedValues[rownames(se) %in% de, ],
            clusterSamplesData = "dendrogramValue",
            whichClusters = "primary",
            main = 'Normalized values, 1000 most variable genes',
            breaks = 0.99)

```

## Further developments

In an effort to improve the workflow and analysis of scRNA-seq data, we are currently exploring statistical methods to perform DE analysis using the models we developed. On the one hand, we are developing a DE analysis method using the framework of slingshot which would take advantage of the lineages found. On the other hand, we are working on a DE analysis method using the ZINB-WaVE model which would account for the dropouts and gene and cell-level covariates. Additionally, we are working on a statistical method using the ZINB-WaVE model to impute the dropouts in the scRNA-seq data. This matrix with imputed counts could for example be used in the workflow in the dimensionality reduction, clustering or pseudo ordering steps. Finally, a new R class called SingleCellExperiment should be released soon. This new class would essentially be a SummarizedExperiment class with a couple of additional slots, the most important of which is reducedDims, which, much like the assays slot of SummarizedExperiment can contain one or more matrices with reduced dimensional matrices. This new SingleCellExperiment class would be a valuable addition to the workflow as we could store in one object the raw counts as well as the low dimensional matrix created by the dimensionality reduction step. Once this class is implemented, we would like to incorporate it to the workflow.

## Conclusion

This workflow provides guidance to perform downstream analysis of scRNA-seq datasets in R. We propose a workflow in four steps: dimensionality reduction adjusting for cell-level covariates, robust and stable clustering, pseudotime ordering, and DE analysis between the lineages. The workflow is general and flexible allowing the user to substitute the statistical method used in each step of the workflow by a different method. We hope our proposed workflow will ease the technical aspect of scRNA-seq data analysis and help focusing on discovering new biological insights.

## Software availability

This section will be generated by the Editorial Office before publication. Authors are asked to provide some initial information to assist the Editorial Office, as detailed below.

1. URL link to where the software can be downloaded from or used by a non-coder (AUTHOR TO PROVIDE; optional)
2. URL link to the author's version control system repository containing the source code (AUTHOR TO PROVIDE; required)
3. Link to source code as at time of publication (*F1000Research* TO GENERATE)
4. Link to archived source code as at time of publication (*F1000Research* TO GENERATE)
5. Software license (AUTHOR TO PROVIDE; required)

The four packages used in the workflow (scone, zinbwave, clusterExperiment, and slingshot) are bioconductor packages and are available on github at respectively <https://github.com/YosefLab/scone>, <https://github.com/drisso/zinbwave>, <https://github.com/epurdom/clusterExperiment>, and <https://github.com/kstreet13/slingshot>. The source code for this package can be found at <https://github.com/fperraudeau/singlecellworkflow> under license XXX.

```
sessionInfo()
```

```
## R version 3.4.0 (2017-04-21)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Sierra 10.12.5
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/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] rARPACK_0.11-0          digest_0.6.12
## [3] RColorBrewer_1.1-2      Rtsne_0.13
## [5] magrittr_1.5              gplots_3.0.1
## [7] ggplot2_2.2.1            doParallel_1.0.10
## [9] iterators_1.0.8           foreach_1.4.3
## [11] slingshot_0.0.3-5        printrcurve_1.1-12
## [13] GEOquery_2.42.0           zinbwave_0.99.4.2
## [15] BiocParallel_1.10.1       clusterExperiment_1.3.1
## [17] scone_1.0.0               scRNASeq_1.2.0
## [19] SummarizedExperiment_1.6.3 DelayedArray_0.2.7
## [21] matrixStats_0.52.2         Biobase_2.36.2
## [23] GenomicRanges_1.28.3       GenomeInfoDb_1.12.2
## [25] IRanges_2.10.2             S4Vectors_0.14.3
## [27] BiocGenerics_0.22.0        knitr_1.16
## [29] BiocStyle_2.4.0
##
## loaded via a namespace (and not attached):
## [1] shinydashboard_0.6.0          R.utils_2.5.0
## [3] RSQLite_1.1-2                 AnnotationDbi_1.38.1
## [5] htmlwidgets_0.8                grid_3.4.0
## [7] trimcluster_0.1-2             RNeXML_2.0.7
## [9] DESeq_1.28.0                  munsell_0.4.3
## [11] codetools_0.2-15             statmod_1.4.29
## [13] scran_1.4.4                  DT_0.2
## [15] miniUI_0.1.1                 colorspace_1.3-2
## [17] energy_1.7-0                 uuid_0.1-2
## [19] pspline_1.0-17              robustbase_0.92-7
## [21] bayesm_3.0-2                 NMF_0.20.6
## [23] tximport_1.4.0                GenomeInfoDbData_0.99.0
## [25] hwriter_1.3.2                 hdf5_2.20.0
## [27] rprojroot_1.2                 EDASeq_2.10.0
```

```

## [29] diptest_0.75-7           R6_2.2.1
## [31] ggbeeswarm_0.5.3         taxize_0.8.4
## [33] locfit_1.5-9.1          flexmix_2.3-14
## [35] bitops_1.0-6            reshape_0.8.6
## [37] assertthat_0.2.0        scales_0.4.1
## [39] nnet_7.3-12             beeswarm_0.2.3
## [41] gtable_0.2.0            phylobase_0.8.4
## [43] RUVSeq_1.10.0           bold_0.4.0
## [45] rlang_0.1.1              genefilter_1.58.1
## [47] splines_3.4.0           rtracklayer_1.36.3
## [49] lazyeval_0.2.0           hexbin_1.27.1
## [51] rgl_0.98.1               yaml_2.1.14
## [53] reshape2_1.4.2           abind_1.4-5
## [55] GenomicFeatures_1.28.3   backports_1.1.0
## [57] httpuv_1.3.3             tensorA_0.36
## [59] tools_3.4.0              bookdown_0.4
## [61] gridBase_0.4-7           stabledist_0.7-1
## [63] dynamicTreeCut_1.63-1    Rcpp_0.12.11
## [65] plyr_1.8.4                progress_1.1.2
## [67] visNetwork_1.0.3          zlibbioc_1.22.0
## [69] purrrr_0.2.2.2            RCurl_1.95-4.8
## [71] prettyunits_1.0.2         viridis_0.4.0
## [73] zoo_1.8-0                 cluster_2.0.6
## [75] data.table_1.10.4         RSpectra_0.12-0
## [77] mvtnorm_1.0-6             whisker_0.3-2
## [79] gsl_1.9-10.3              aroma.light_3.6.0
## [81] mime_0.5                  evaluate_0.10
## [83] xtable_1.8-2              XML_3.98-1.7
## [85] mclust_5.3                gridExtra_2.2.1
## [87] compiler_3.4.0            biomaRt_2.32.1
## [89] scater_1.4.0              tibble_1.3.3
## [91] KernSmooth_2.23-15        R.oo_1.21.0
## [93] htmltools_0.3.6           pcaPP_1.9-61
## [95] segmented_0.5-2.0          BiocWorkflowTools_1.2.0
## [97] tidyR_0.6.3                geneplotter_1.54.0
## [99] howmany_0.3-1              DBI_0.6-1
## [101] MASS_7.3-47               fpc_2.1-10
## [103] MAST_1.2.1               boot_1.3-19
## [105] compositions_1.40-1      ade4_1.7-6
## [107] ShortRead_1.34.0          Matrix_1.2-10
## [109] R.methodsS3_1.7.1          gdata_2.18.0
## [111] igraph_1.0.1              rncl_0.8.2
## [113] GenomicAlignments_1.12.1   registry_0.3
## [115] numDeriv_2016.8-1          locfdr_1.1-8
## [117] plotly_4.7.0              xml2_1.1.1
## [119] annotate_1.54.0            viper_0.4.5
## [121] rngtools_1.2.4             pkgmaker_0.22
## [123] XVector_0.16.0            stringr_1.2.0
## [125] copula_0.999-16           ADGofTest_0.3
## [127] softImpute_1.4             Biostrings_2.44.1
## [129] rmarkdown_1.5               dendextend_1.5.2
## [131] edgeR_3.18.1              kernlab_0.9-25
## [133] shiny_1.0.3                Rsamtools_1.28.0
## [135] gtools_3.5.0               modeltools_0.2-21
## [137] rjson_0.2.15              nlme_3.1-131
## [139] jsonlite_1.5              viridisLite_0.2.0
## [141] limma_3.32.2              lattice_0.20-35
## [143] httr_1.2.1                DEoptimR_1.0-8
## [145] survival_2.41-3            glue_1.0.0
## [147] FNN_1.1                  prabclus_2.2-6
## [149] glmnet_2.0-10              class_7.3-14
## [151] stringi_1.1.5              mixtools_1.1.0
## [153] latticeExtra_0.6-28         caTools_1.17.1
## [155] memoise_1.1.0              dplyr_0.7.0
## [157] ape_4.1

```

### **Author contributions**

In order to give appropriate credit to each author of an article, the individual contributions of each author to the manuscript should be detailed in this section. We recommend using author initials and then stating briefly how they contributed.

### **Competing interests**

All financial, personal, or professional competing interests for any of the authors that could be construed to unduly influence the content of the article must be disclosed and will be displayed alongside the article. If there are no relevant competing interests to declare, please add the following: 'No competing interests were disclosed'.

### **Grant information**

Please state who funded the work discussed in this article, whether it is your employer, a grant funder etc. Please do not list funding that you have that is not relevant to this specific piece of research. For each funder, please state the funder's name, the grant number where applicable, and the individual to whom the grant was assigned. If your work was not funded by any grants, please include the line: 'The author(s) declared that no grants were involved in supporting this work.'

### **Acknowledgments**

This section should acknowledge anyone who contributed to the research or the article but who does not qualify as an author based on the criteria provided earlier (e.g. someone or an organization that provided writing assistance). Please state how they contributed; authors should obtain permission to acknowledge from all those mentioned in the Acknowledgments section.

### **References**

- Dijk, David van, Juozas Nainys, Roshan Sharma, Pooja Kathail, Ambrose J Carr, Kevin R Moon, Linas Mazutis, Guy Wolf, Smita Krishnaswamy, and Dana Pe'er. 2017. "MAGIC: A diffusion-based imputation method reveals gene-gene interactions in single-cell RNA-sequencing data." *BioRxiv*. doi:10.1101/111591.
- Fletcher, Russell B, Diya Das, Levi Gadye, Kelly N Street, Ariane Baudhuin, Allon Wagner, Michael B Cole, et al. 2017. "Deconstructing Olfactory Stem Cell Trajectories at Single-Cell Resolution." *Cell Stem Cell* 20 (6). Elsevier: 817–830.e8. doi:10.1016/j.stem.2017.04.003.
- Huber, Wolfgang, Vincent J Carey, Robert Gentleman, Simon Anders, Marc Carlson, Benilton S Carvalho, Hector Corrada Bravo, et al. 2015. "Orchestrating high-throughput genomic analysis with Bioconductor." *Nature Methods* 12 (2): 115–21. doi:10.1038/nmeth.3252.
- Lun, Aaron T.L., Davis J. McCarthy, and John C. Marioni. 2016. "A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor." *F1000Research* 5 (October): 2122. doi:10.12688/f1000research.9501.2.
- McCarthy, Davis J., Kieran R. Campbell, Aaron T. L. Lun, and Quin F. Wills. 2017. "Scater: pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R." *Bioinformatics*, January, btw777. doi:10.1093/bioinformatics/btw777.
- Risso, Davide, Fanny Perraudeau, Svetlana Gribkova, Sandrine Dudoit, and Jean-Philippe Vert. 2017. "ZINB-WaVE: A general and flexible method for signal extraction from single-cell RNA-seq data." doi:10.1101/125112.