Sincell: R package for statistical assessment of cell state hierarchies from single-cell RNA-seq

Miguel Juliá [1,2]***, Amalio Telenti***[3]***, Antonio Rausell***[1,2*]

[1em] [1] Vital-IT group, SIB Swiss Institute of Bioinformatics, 1015 Lausanne, Switzerland
[2] University of Lausanne, 1015 Lausanne, Switzerland
[3] J. Craig Venter Institute, La Jolla, CA 92037

*Correspondence to antonio.rausell (at) isb-sib.ch

May 19, 2021

Abstract

Cell differentiation processes are achieved through a continuum of hierarchical intermediate cell-states that might be captured by single-cell RNA seq. Existing computational approaches for the assessment of cell-state hierarchies from single-cell data might be formalized under a general framework composed of i) a metric to assess cell-to-cell similarities (combined or not with a dimensionality reduction step), and ii) a graph-building algorithm (optionally making use of a cells-clustering step). *Sincell* R package implements a methodological toolbox allowing flexible workflows under such framework. Furthermore, *Sincell* contributes new algorithms to provide cell-state hierarchies with statistical support while accounting for stochastic factors in single-cell RNA seq. Graphical representations and functional association tests are provided to interpret hierarchies. Some functionalities are illustrated in a real case study where their ability to discriminate noisy from stable cell-state hierarchies is demonstrated.

Sincell **version:** 1.24.0 [1]

[1]This document used vignette from *Bioconductor* package *DESeq2* as knitr template

If you use *Sincell* in published research, please cite:

Juliá M, Telenti A, Rausell A. Sincell - R package for the statistical assessment of cell state hierarchies from single-cell RNA-seq data. BioRxiv 2014 doi:

Contents

1 Introduction

Unbiased transcriptome profiling of individual cells through RNA-seq allows capturing the continuum of cell states transited through a differentiation or activation process. Such continuum can be represented by a sequential ordering of cell-states ultimately leading to heterogeneous cell types within a sample. The assessment of cell-state hierarchies from single RNA-seq is challenging due to the intrinsic stochasticity of gene expression and technical noise. Statistical support is further needed in experimental settings where the number of cells is several orders of magnitude lower than the number of genes profiled.

A number of computational methods have been developed to assess cell-state hierarchies from single-cell data generated with different technologies (see Juliá et al Bioarxiv 2014, Supplementary Table 1). The different methods can be dissected into prototypical steps: a) the assessment of a cell-to-cell similarity matrix by a given metric (preceded or not with a dimensionality reduction step), and b) a graph-building algorithm that can optionally make use of a cells-clustering step.

Sincell integrates the various components of that general workflow in a way that they can be combined in a user-defined manner (see Juliá et al Bioarxiv 2014, Figure 1). Different alternatives are provided for each step of the analysis and new algorithms are contributed. Notably, *Sincell* implements algorithms to assess the statistical support of cell-state hierarchies derived from single-cell RNAseq. Different graphical representations and functional association tests are proposed to help the user interpreting the results. A complete description of the package and algorithms herein can be found in Juliá et al (Bioarxiv, 2014)

2 Installing *Sincell* package

Sincell depends on following CRAN-R packages: entropy; fastICA; fields; ggplot2; igraph; MASS; proxy; reshape2; Rtsne; scatterplot3d, TSP.

Installing *Sincell* package from Bioconductor will also install all its dependencies:

```
if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
BiocManager::install("sincell")
```

In this vignette we will further make use of libraries "biomaRt" and "monocle" from Bioconductor:

```
packages.bioconductor<-c("biomaRt","monocle")
packages.bioconductor.2install <- packages [!(packages.bioconductor
     %in%  installed.packages()[, "Package"])]
if(length(packages.bioconductor.2install)>0){
  for (i in 1:length(packages.bioconductor.2install)){
    BiocManager::install(packages.bioconductor.2install[i])
  }
}
```

We may now load package *Sincell*

```
library(sincell)
```

3 Loading an expression matrix into a *Sincell* object

Sincell workflow starts from an expression matrix comprising the expression levels of each single-cell in the experiment (displayed by columns) for each detected gene (displayed by rows). Before starting using *Sincell*, quality controls to filter out individual cells from the analysis have to be performed by the user. Expression levels need also to be previously normalized to account for library size or technical variability (e.g. through the use of spike-in molecules).

Some *Sincell* functions are computationally intensive. Implementation of some *Sincell*'s algorithms in C++ as well as parallelization of *Sincell*'s functions permit to decrease running times. However, working with a gene expression matrix of several thousands of genes can lead to long computing times depending on the available hardware. If time or computation capacity is an issue, we recommend the user to restrict the analysis to the most variable protein coding genes in the dataset. These genes drive most of the signal to assess cell-state hierarchies and restricting the analyses to them should not bias the final results. Typically, the ensemble of *Sincell* routines on an expression matrix with 2000 genes and 180 individual cells takes less than one hour in a laptop with 8GB RAM. Nevertheless, selecting the most variable genes is not straightforward due to the mean-variance relationship. To select the most variable genes we refer the user to the two following excellent tutorials:

i) the HSC Harvard Catalyst Single-Cell Workshop 2014: RNA-seq (http://pklab.med.harvard.edu/scw2014/subpop_tutorial.html; Section "Identifying highly variable genes")

and ii) Bioconductor RNA-Seq workflow (http://master.bioconductor.org/help/workflows/rnaseqGene/#de; Section "The rlog transformation").

Quoting from this last document by Love M, Anders S and Huber W:

"Many common statistical methods for exploratory analysis of multidimensional data, especially methods for clustering and ordination (e.g., principal-component analysis and the like), work best for (at least approximately) homoskedastic data; this means that the variance of an observed quantity (here, the expression strength of a gene) does not depend on the mean. In RNA-Seq data, however, variance grows with the mean. For example, if one performs PCA (principal components analysis) directly on a matrix of normalized read counts, the result typically depends only on the few most strongly expressed genes because they show the largest absolute differences between samples"

To avoid this bias, we recommend the user performing a variance stabilizing transformation. The simplest way is taking the logarithm of the normalized count values plus 10 (or 100) pseudocounts. More sophisticated strategies are provided by the function *rlog* and other variance stabilizing transformations discussed in the Bioconductor package *DESeq2*.

To restrict the analysis to protein coding genes, a list of them can be downloaded from Ensembl BioMart (http://www.ensembl.org/biomart/) for different organisms and in different types of gene id's (e.g. Entrez, Ensembl, HGNC). Bioconductor package *biomaRt* provides access to that information from the R environment (see below).

Once quality control, normalization and variance stabilization (e.g. log transformation with 100 pseudocounts) have been performed and the most variable genes identified (e.g. the 2000 most variable protein coding genes), we are ready to start working with *Sincell*.

Function *sc_InitializingSincellObject()* initializes a *Sincell* object from the gene expression matrix

```
# Do not run
SincellObject<-sc_InitializingSincellObject(ExpressionMatrix)
```

In this vignette we will illustrate *Sincell* usage on a publicly available single-cell RNA-seq dataset taken from Trapnell et al 2014. In this work, authors generated single-cell RNA-seq libraries for differentiating myoblasts at 0, 24, 48 and 72 hours. Original data can be accessed at GEO database accession number GSE52529 (ftp://ftp.ncbi.nlm.nih.gov/geo/series/GSE52nnn/GSE52529/suppl/

GSE52529_fpkm_matrix.txt.gz). Following Trapnell et al 2014 and the vignette of its associated *Bioconductor* package *monocle* , the expression matrix is restricted to genes differentially expressed between cells from times 0 and the ensemble of cells of times 24, 28 and 72 hours of differentiation. Steps to achieve this are reported below quoting *monocle*'s vignette, though we recommend not to run them here due to the long computing time needed to complete them.

```
# Within R console we load monocle package:
library(monocle)
```

```
# WARNING: do not run

HSMM <- detectGenes(HSMM, min_expr = 0.1)
expressed_genes <- row.names(subset(fData(HSMM), num_cells_expressed >= 50))
# The vector expressed_genes now holds the identifiers for genes expressed in
# at least 50 cells of the data set.

# Keeping expressed genes with q-value < 0.01
diff_test_res <- differentialGeneTest(HSMM[expressed_genes,],
                          fullModelFormulaStr = "expression~Media")
ordering_genes <- row.names(subset(diff_test_res, qval < 0.01))
HSMM <- HSMM[ordering_genes,]
```

In order to keep the running time of this vignette short, we provide the expression matrix produced by previous steps:

```
data(ExpressionMatrix)
```

From the expression matrix we get the log-transformed expression matrix

```
EMlog <- unique(
       log(ExpressionMatrix[which(apply(ExpressionMatrix,1,var)>0),]+1)
       )
EMlog <- as.matrix( EMlog[as.logical(apply(!is.nan(EMlog),1,sum)),])
```

Optionally, we can change gene Ensembl identifiers to HGCN symbols using biomaRt

```
GeneEnsemblID<-rownames(EMlog)
head(GeneEnsemblID)
```

```
## [1] "ENSG00000000460.12" "ENSG00000001630.11" "ENSG00000003989.12"
## [4] "ENSG00000005448.12" "ENSG00000010292.8"  "ENSG00000011426.6"
```

```
GeneEnsemblID <- sapply( strsplit(GeneEnsemblID, split=".",fixed=TRUE),"[",1)
head(GeneEnsemblID)
```

```
## [1] "ENSG00000000460" "ENSG00000001630" "ENSG00000003989" "ENSG00000005448"
## [5] "ENSG00000010292" "ENSG00000011426"
```

```
library("biomaRt")
ensembl = useMart( "ensembl", dataset = "hsapiens_gene_ensembl" )
genemap <- getBM( attributes = c("ensembl_gene_id", "entrezgene_id", "hgnc_symbol"),
        filters = "ensembl_gene_id", values = GeneEnsemblID, mart = ensembl )
idx <- match(GeneEnsemblID, genemap$ensembl_gene_id )
GeneEntrez <- genemap$entrezgene_id[ idx ]
GeneHGCN <- genemap$hgnc_symbol[ idx ]

rownames(EMlog)[!is.na(GeneHGCN)&(GeneHGCN!="")]<-
        GeneHGCN[!is.na(GeneHGCN)&(GeneHGCN!="")]
head(rownames(EMlog))
```

```
## [1] "C1orf112" "CYP51A1"  "SLC7A2"   "WDR54"    "NCAPD2"   "ANLN"
```

Finally, we can initialize a *Sincell* object from the gene expression matrix. This done with the function *sc_InitializingSincellObject()*. At this stage, genes with a variance equal to zero are filtered out from the gene expression matrix.

```
SO<-sc_InitializingSincellObject(EMlog)
```

Within a Sincell object, the loaded gene expression matrix can be accessed as a named list member "expressionmatrix"

```
expressionmatrix<-SO[["expressionmatrix"]]
```

4 Assessment of a cell-to-cell distance matrix

4.1 Assessment of a cell-to-cell distance matrix with a metric of choice

The first requirement to obtain a cell-state hierarchy is to assess a cell-to-cell distance matrix. *Sincell*'s function *sc_distanceObj()* provides both linear and non-linear distances: Euclidean distance, Pearson and Spearman correlation, L1 distance, Cosine distance and Mutual Information.

```
## Assessment of a cell-to-cell distance matrix
# help(sc_distanceObj())

# Euclidean distance
SO<-sc_distanceObj(SO, method="euclidean")
cell2celldist_Euclidean<-SO[["cell2celldist"]]

# Cosine distance
SO<-sc_distanceObj(SO, method="cosine")
cell2celldist_Cosine<-SO[["cell2celldist"]]

# Distance based on 1-Pearson correlation
SO<-sc_distanceObj(SO, method="pearson")
cell2celldist_Pearson<-SO[["cell2celldist"]]

# Distance based on 1-Spearman correlation
SO<- sc_distanceObj(SO, method="spearman")
cell2celldist_Spearman<-SO[["cell2celldist"]]

# L1 distance
SO<- sc_distanceObj(SO, method="L1")
cell2celldist_L1<-SO[["cell2celldist"]]
```

```
# Mutual information distance is assessed by making bins of expression levels
# as defined by the "bines" parameter. This function internally calls function
#  mi.empirical from package "entropy" using unit="log2".
SO<-sc_distanceObj(SO, method="MI", bins=c(-Inf,0,1,2,Inf))
cell2celldist_MI<-SO[["cell2celldist"]]
```

4.2 Assessment of a cell-to-cell distance matrix with an intermediate dimensionality reduction step

The cell-to-cell distance matrix can be assessed on the new dimensions obtained from a dimensionality reduction algorithm. *Sincell*'s function *sc_DimensionalityReduction()* provides access to the main types of dimensionality reduction techniques either linear or non-linear, from which cell-to-cell distances can be assessed. The methods provided are: Principal Component Analysis (PCA), Independent Component Analysis (ICA), t-Distributed Stochastic Neighbor Embedding (tSNE), classical Multidimensional Scaling and non-metric Multidimensional Scaling.

For instance, to perform a PCA with 3 dimensions:

```
# help(sc_DimensionalityReductionObj)

# Principal Component Analysis (PCA)
SO <- sc_DimensionalityReductionObj(SO, method="PCA", dim=3)
cellsLowDimensionalSpace_PCA<-SO[["cellsLowDimensionalSpace"]]
```

When choosing PCA, the proportion of variance explained by each of the principal axes can be plotted to allow the user judge whether more dimensions should be considered:

```
plot(SO[["EigenValuesPCA"]],las=1,
        main="Proportion of variance explained by\neach PCA principal axis",
        ylab="Proportion of variance",xlab="Principal axes",
        pch=16,ylim=c(0,0.25))
```

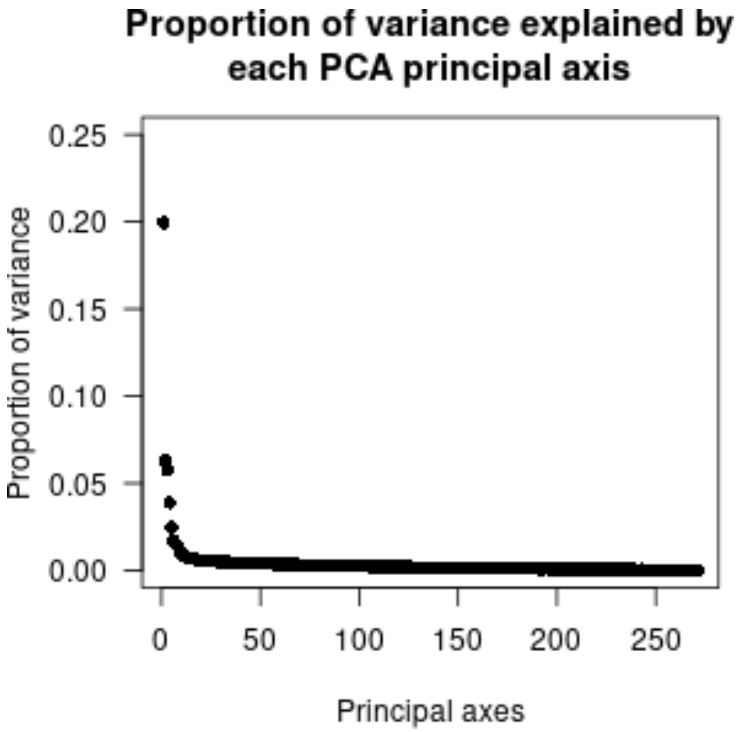

Figure 1: **Proportion of variance explained by each principal axis obtained by a Principal Component Analysis (PCA) of the expression matrix**

ICA and t-SNE are other two types of dimensionality reduction techniques that can be used:

```
# Independent Component Analysis (ICA)
SO <- sc_DimensionalityReductionObj(SO, method="ICA", dim=3)
```

```
cellsLowDimensionalSpace_ICA<-SO[["cellsLowDimensionalSpace"]]
# t-Distributed Stochastic Neighbor Embedding (t-SNE)
SO <- sc_DimensionalityReductionObj(SO, method="tSNE", dim=3)
cellsLowDimensionalSpace_tSNE<-SO[["cellsLowDimensionalSpace"]]
```

The reader should be aware that ICA and tSNE are optimization algorithms, therefore different runs can lead to different solutions on the same input. Those stochastic differences are expected to be small though in certain cases two runs could lead to very different results.

We note also that *Sincell* makes use of the Rtsne implementation of the Barnes-Hut algorithm, which approximates the likelihood. The user should be aware that this is a less accurate version of t-SNE than e.g. the one used as basis of viSNE (Amir,E.D. et al. 2013, Nat Biotechnol 31, 545?552).

To run Multidimensional Scaling:

```
# Classic Multidimensional Scaling (classical-MDS).
SO <- sc_DimensionalityReductionObj(SO, method="classical-MDS", dim=3)
cellsLowDimensionalSpace_classicalMDS<-SO[["cellsLowDimensionalSpace"]]

# Non-metric Multidimensional Scaling (nonmetric-MDS).
SO <- sc_DimensionalityReductionObj(SO, method="nonmetric-MDS", dim=3)
cellsLowDimensionalSpace_nonmetricMDS<-SO[["cellsLowDimensionalSpace"]]
```

Once the dimensionality reduction has been performed, the cell-to-cell distance matrix that has been calculated on the new low dimensional space can be accessed as:

```
# Cell-to-cell distance matrix from coordinates in low dimensional space
cell2celldist <- SO[["cell2celldist"]]
```

4.3 Graphical representation of individual cells in low dimensional space

At this stage, the graphical representation of the individual cells into a low dimensional space can help identifying patterns of relative similarities as well as groups of cell-states:

To represent the first two dimensions of the low-dimensional space:

```
plot(t(cellsLowDimensionalSpace_ICA),col= "black",
     xlab="First axis after dimensionality reduction",
```

```
     ylab="Second axis after dimensionality reduction")

# We can add to the plot the individual cell identifiers
# as indicated by the column's name of the matrix
text(t(cellsLowDimensionalSpace_ICA),colnames(cellsLowDimensionalSpace_ICA),
     pos=3,cex=0.4,col="black")
```

If additional information about the individual cells is available, it can be used to further define coloring schemes that would help interpreting the observed variability. For instance, in the data from Trapnell et al 2014, we can use time points of differentiating myoblasts (0, 24, 48 and 72 hours) as the color scheme to interpret the relative position of cells in low dimensional space (Figure **??**, left)

First, we build a color vector for the single-cell RNA-seq libraries according to their time of differentiation at 0, 24, 48 and 72 hours, as indicated in the column's name of the matrix:

```
ColorT0<-"blue"
ColorT24<-"green"
ColorT48<-"orange"
ColorT72<-"red"

colorSCsByTime<- character(dim(SO[["expressionmatrix"]])[2])
colorSCsByTime[grep("T0_", colnames(SO[["expressionmatrix"]]))]<- ColorT0
colorSCsByTime[grep("T24_",colnames(SO[["expressionmatrix"]]))]<- ColorT24
colorSCsByTime[grep("T48_",colnames(SO[["expressionmatrix"]]))]<- ColorT48
colorSCsByTime[grep("T72_",colnames(SO[["expressionmatrix"]]))]<- ColorT72
```

```
mycex<-1.2
mylwd<-4
myps<-1.2
par(bty="o",xaxs="i",yaxs="i",cex.axis=mycex-0.2,cex.main=mycex,cex.lab=mycex,
     las=1,mar=c(5.3,5.3,2.9,1.6),oma=c(1,1,2,6))
plot(t(cellsLowDimensionalSpace_ICA),col= colorSCsByTime, xlab="First axis",
     ylab="Second axis",main="ICA")
par(fig = c(0, 1, 0, 1), oma = c(0, 0, 0, 0), mar = c(0, 0, 0, 0), new = TRUE,
     xpd = TRUE)
plot(0, 0, type = "n", bty = "n", xaxt = "n", yaxt = "n")
legend("right",title="Time\npoint", c("0h","24h","48h","72h"),
     fill= c(ColorT0,ColorT24,ColorT48,ColorT72), inset=c(0.03,0),  bty="n")
```

To further interpret the relative position in the low dimensional space, individual cells can be colored according to the intensity of expression of a user-defined marker gene. *Sincell* implements function *sc_marker2color()* to help creating color scales (Figure **??**, right):

```
myMarker<-"CDK1"
colorSCsByMarker<-sc_marker2color(SO, marker=myMarker, color.minimum="yellow3",
        color.maximum="blue", relative.to.marker=TRUE)
```

```
mycex<-1.2
mylwd<-4
myps<-1.2
par(bty="o",xaxs="i",yaxs="i",cex.axis=mycex-0.2,cex.main=mycex,cex.lab=mycex,
        las=1,mar=c(5.3,5.3,2.9,1.6),oma=c(1,1,2,6))
plot(t(cellsLowDimensionalSpace_ICA[1:2,]),col= colorSCsByMarker,
        xlab="First axis", ylab="Second axis",
        main=paste("ICA - Marker:",myMarker),pch=10)
```

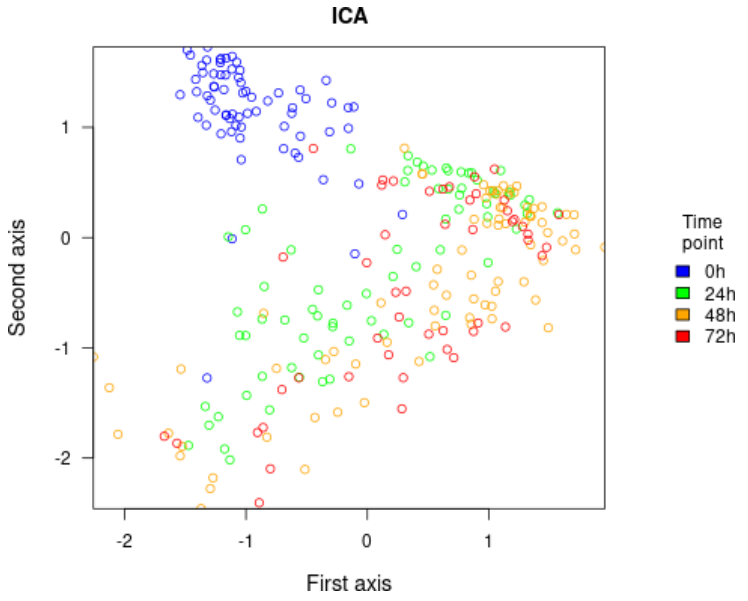

Figure 2:

We may now produce a panel with the results for different dimensionality reduction algorithms exploring up to 3 dimensions:

```
mycex<-1.5
mylwd<-4
myps<-1.2
par(bty="o",xaxs="i",yaxs="i",cex.axis=mycex-0.2,cex.main=mycex,cex.lab=mycex,
```

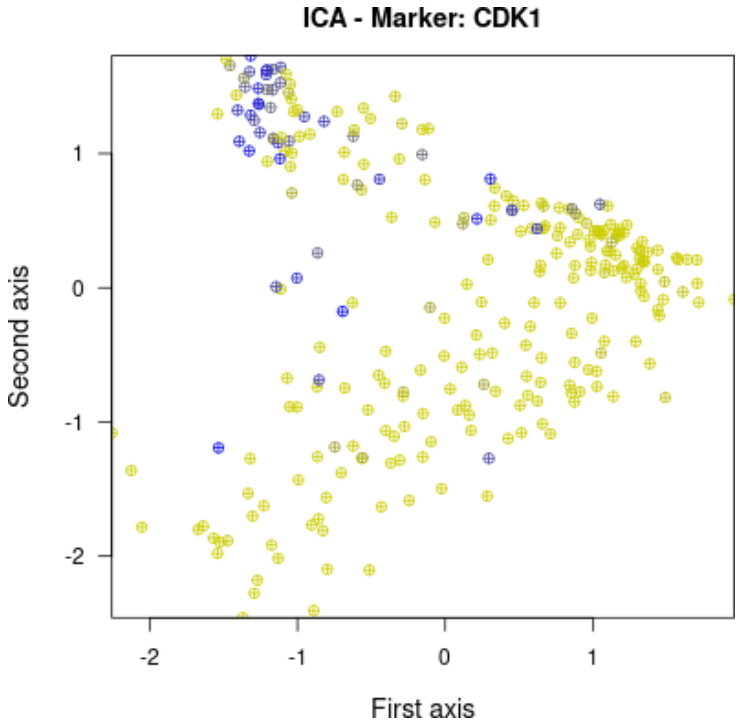

Figure 3: **Representation of individual cells in the low dimensional space obtained by performing an Independent Component Analysis on the initial expression matrix**
Color code on the left represents the time point of the differentiating myoblast library, while on the right it represents the relative expression levels of the marker from minimum (yellow) to maximum (blue).

```
        las=1,mar=c(5.3,5.3,2.9,1.6),oma=c(1,1,2,10))
zones=matrix(c(1:8),ncol=2,byrow=FALSE)

layout(zones)
myxlab="First axis"
myylab="Second axis"
plot(t(cellsLowDimensionalSpace_PCA[1:2,]),col= colorSCsByTime,
        xlab=myxlab, ylab= myylab, main="PCA")
plot(t(cellsLowDimensionalSpace_ICA[1:2,]),col= colorSCsByTime,
        xlab=myxlab, ylab= myylab, main="ICA")
plot(t(cellsLowDimensionalSpace_tSNE[1:2,]),col= colorSCsByTime,
        xlab=myxlab, ylab= myylab, main="tSNE")
plot(t(cellsLowDimensionalSpace_nonmetricMDS[1:2,]),col= colorSCsByTime,
        xlab=myxlab, ylab= myylab, main="nonmetricMDS")

myxlab="Third axis"
myylab="Second axis"
```

```
plot(t(cellsLowDimensionalSpace_PCA[c(3,2),]),col= colorSCsByTime,
        xlab=myxlab, ylab= myylab, main="PCA")
plot(t(cellsLowDimensionalSpace_ICA[c(3,2),]),col= colorSCsByTime,
        xlab=myxlab, ylab= myylab, main="ICA")
plot(t(cellsLowDimensionalSpace_tSNE[c(3,2),]),col= colorSCsByTime,
        xlab=myxlab, ylab= myylab, main="tSNE")
plot(t(cellsLowDimensionalSpace_nonmetricMDS[c(3,2),]),col= colorSCsByTime,
        xlab=myxlab, ylab= myylab, main="nonmetricMDS")

par(fig = c(0, 1, 0, 1), oma = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
        new = TRUE, xpd = TRUE)
plot(0, 0, type = "n", bty = "n", xaxt = "n", yaxt = "n")
legend("right",title="Time\npoint", c("0h","24h","48h","72h"),
        fill= c(ColorT0,ColorT24,ColorT48,ColorT72), inset=c(0.01,0),bty="n")
```

Figure 4 represents individual cells in several low dimensional spaces as calculated by different algorithms: PCA, ICA, tSNE and non-metric MDS. Panels on the left represent axis 1 and 2, and panels on the right axis 3 and 2 of the corresponding algorithm. Color code corresponds to the time of differentiation as indicated in the legend. It can be observed that the non-metric Multidimensional Scaling is able to separate in its 1st and 3rd dimension the samples according to their time of differentiation. This example shows that exploring different algorithms as well as different dimensions can help capturing the internal structure of the data.

Further inspection of low dimensional spaces can be done with a 3D visualization as provided by the rgl package:

```
library(rgl)
# Coloring by time point
plot3d(t(cellsLowDimensionalSpace_nonmetricMDS),
        cex=1, size=2, type="s", col= colorSCsByTime)
plot3d(t(cellsLowDimensionalSpace_ICA),
        cex=1, size=2, type="s", col= colorSCsByTime)
plot3d(t(cellsLowDimensionalSpace_PCA),
        cex=1, size=2, type="s", col= colorSCsByTime)
plot3d(t(cellsLowDimensionalSpace_tSNE),
        cex=1, size=2, type="s", col= colorSCsByTime)

# Coloring by marker
plot3d(t(cellsLowDimensionalSpace_nonmetricMDS), cex=1, size=2, type="s", col= colorSCsByMa
```

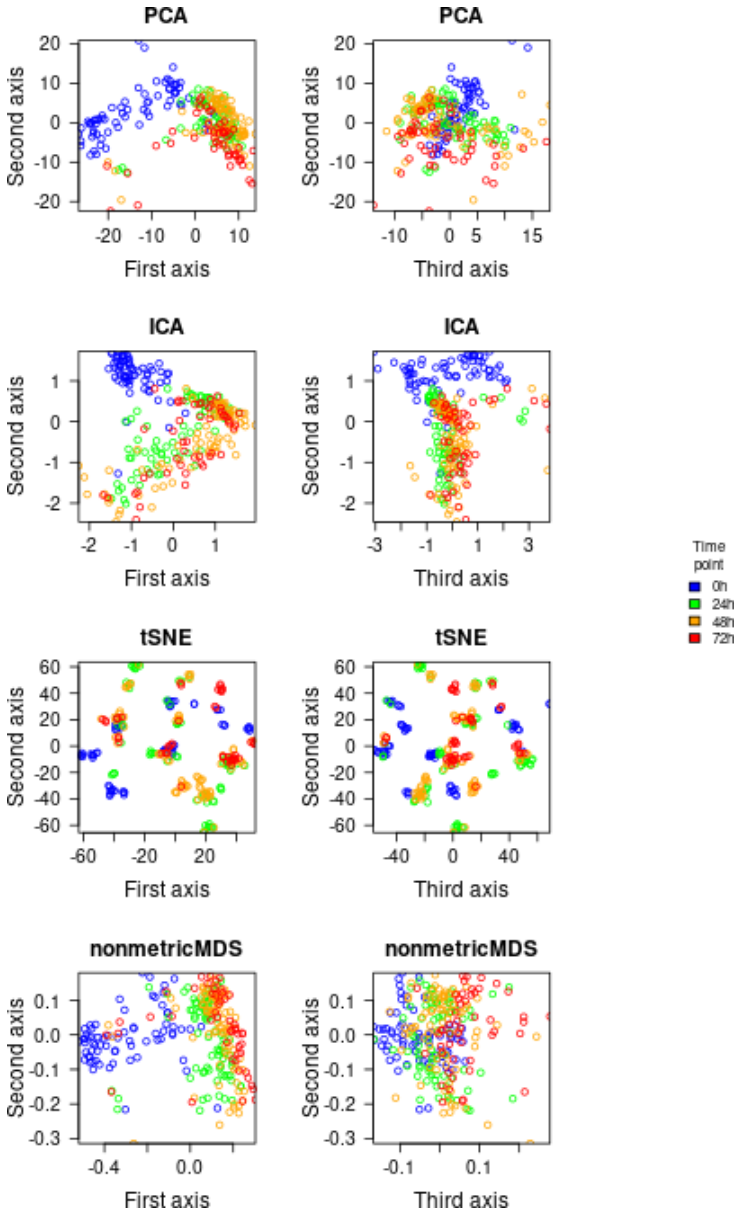

Figure 4: Representation of individual cells in low dimensional space (see text)

For the following steps in the vignette we will use an ICA dimensionality reduction in two dimensions:

```
# Independent Component Analysis (ICA)
SO <- sc_DimensionalityReductionObj(SO, method="ICA", dim=2)
```

5 Assessment of a cell-state hierarchy

In Sincell, a cell-state hierarchy is obtained by applying a graph-building algorithm on the previously calculated cell-to-cell distance matrix. Graph-building algorithms may be applied on the cells taken individually or in groups of highly similar cells, as determined by an intermediate clustering step.

5.1 Optional clustering of individual cells

Identifying groups of cells particularly homogeneous among them by using a clustering step is recommended to avoid misleading representations produced by a graph-building algorithm: i.e. when two cells, despite being very similar, fall far apart in the graph not because they are different but because they are connected through intermediate cells that are even more similar to either two.

Sincell's function *sc_clusterObj()* calculates a disconnected graph where the connected components are the clusters generated by a clustering method. This function provides access to the different clustering methods reported below, and the cells grouped in each cluster can be easily accessed through the function clusters() from igraph package:

```
# help(sc_clusterObj)

# Clusters defined as subgraphs generated by a maximum pair-wise distance cut-off:
# From a totally connected graph where all cells are connected to each other,
# the algorithm keeps only pairs of cells connected by a distance lower than
# a given threshold
SO<- sc_clusterObj(SO, clust.method="max.distance", max.distance=0.5)
cellsClustering_SubgraphDist <-SO[["cellsClustering"]]
clusters(SO[["cellsClustering"]])

# Clusters defined as subgraphs generated by a given rank-percentile of the
# shortest pair-wise distances:
# From a totally connected graph where all cells are connected to each other,
# the algorithm  keeps only the top "x" percent of shortest pairwise distances.
# In the example, only the shortest 10\% of all pairwise distances are retained
# to define subgraphs.
SO<- sc_clusterObj(SO, clust.method="percent", shortest.rank.percent=10)
cellsClustering_SubgraphPercent <-SO[["cellsClustering"]]
clusters(SO[["cellsClustering"]])

# K-Nearest Neighbours (K-NN) clustering.
# In the example k is set up to 3
```

```
SO <- sc_clusterObj (SO, clust.method="knn", mutual=FALSE, k=3)
cellsClustering_KNN <-SO[["cellsClustering"]]
clusters(SO[["cellsClustering"]])

# K-Mutual Nearest Neighbours clustering.
# We present here a variant from K-NN clustering in which only k reciprocal
# nearest neighbours are clustered together.
SO <- sc_clusterObj (SO, clust.method="knn", mutual=TRUE, k=3)
cellsClustering_KMNN <-SO[["cellsClustering"]]
clusters(SO[["cellsClustering"]])
```

For single-cell data we recommend using K-NN. Other methods available in the *sc_clusterObj()* function are:

```
# K-medoids, as calculated by function pam() in package "cluster" on
# the distance matrix SO[["cell2celldist"]] with a predefined number of groups k
SO <- sc_clusterObj (SO, clust.method="k-medoids", mutual=TRUE, k=3)
cellsClustering_kmedoids <-SO[["cellsClustering"]]
clusters(SO[["cellsClustering"]])

# Agglomerative clustering as calculated by function hclust with different
# methods and "cutting" the tree in a given number of groups k with
# function cutree()
SO <- sc_clusterObj (SO, clust.method="complete", mutual=TRUE, k=3)
cellsClustering_hclustcomplete <-SO[["cellsClustering"]]
clusters(SO[["cellsClustering"]])
```

When applying hierarchical agglomerative clustering or k-medoids clustering, we recommend the user no to create clusters of large sizes that, later in the workflow, might risk masking meaningful gradients in the graphs/hierarchies. We refer the reader to common clustering and plotting R functions to further explore the optimal number of clusters to be used as a parameter in the previous hierarchical and k-means clustering: http://www.statmethods.net/advstats/cluster.html

5.2 Graph-building step

Sincell's function *sc_GraphBuilderObj()* provides access to three different algorithms to assess a graph from a cell-to-cell distance matrix: the Minimum Spanning Tree (MST), the Maximum Similarity Spanning Tree (SST) and the

Iterative Mutual Clustering Graph (IMC). SST and IMC are two new graph-building algorithms implemented in *Sincell* for the first time and are described in Juliá et al (Bioarxiv, 2014).

For example, to assess MST, SST or IMC graphs on the cell-to-cell distance matrix previously assessed for the Sincell Object "SO" we may run:

```
# help(sc_GraphBuilderObj)

# Minimum Spanning Tree (MST)
SO<- sc_GraphBuilderObj(SO, graph.algorithm="MST",
        graph.using.cells.clustering=FALSE)
cellstateHierarchy_MST<-SO[["cellstateHierarchy"]]
```

```
# Maximum Similarity Spanning Tree (SST)
SO<- sc_GraphBuilderObj(SO, graph.algorithm="SST",
        graph.using.cells.clustering=FALSE)
cellstateHierarchy_SST<-SO[["cellstateHierarchy"]]
```

```
# Iterative Mutual Clustering Graph (IMC)
SO<- sc_GraphBuilderObj(SO, graph.algorithm="IMC")
cellstateHierarchy_IMC<-SO[["cellstateHierarchy"]]
```

Optionally, algorithms in *sc_GraphBuilderObj()* can use clusters of cells to i) overlay connections between pairs of cells belonging to the same cluster (in the case of MST):

```
# Minimum Spanning Tree (MST) with previous clustering of cells

# MST with K-Mutual Nearest Neighbours clustering
SO <- sc_clusterObj (SO, clust.method="knn", mutual=TRUE, k=5)
SO<- sc_GraphBuilderObj(SO, graph.algorithm="MST",
        graph.using.cells.clustering=TRUE)
cellstateHierarchy_MST_clustKNN<-SO[["cellstateHierarchy"]]

# MST with K-medoids
SO <- sc_clusterObj (SO, clust.method="k-medoids", k=15)
SO<- sc_GraphBuilderObj(SO, graph.algorithm="MST",
        graph.using.cells.clustering=TRUE)
cellstateHierarchy_MST_clustKmedoids<-SO[["cellstateHierarchy"]]
```

or ii) treat those clusters as atomic elements in the graph-building process together with non-clustered cells (in the case of SST):

```
# SST with previous clustering of cells

# SST with K-Mutual Nearest Neighbours clustering
SO <- sc_clusterObj (SO, clust.method="knn", mutual=TRUE, k=5)
SO<- sc_GraphBuilderObj(SO, graph.algorithm="SST",
        graph.using.cells.clustering=TRUE)
cellstateHierarchy_SST_clustKNN<-SO[["cellstateHierarchy"]]

# SST with K-medoids
SO <- sc_clusterObj (SO, , clust.method="k-medoids", k=15)
SO<- sc_GraphBuilderObj(SO, graph.algorithm="SST",
        graph.using.cells.clustering=TRUE)
cellstateHierarchy_SST_clustKmedoids<-SO[["cellstateHierarchy"]]
```

By definition, IMC builds a connected graph through iterations on the clustering results produced by the K-Mutual Nearest Neighbour (K-MNN) algorithm, so there is no need to indicate a value for parameter "graph.using.cells.clustering".

5.3 Graphical representation of cell-state hierarchies

In this section, we illustrate how to obtain graphical representations of the graphs generated with *Sincell*'s function *sc_GraphBuilderObj()*. These hierarchies are "igraph" graph objects (see "igraph" R package documentation at http://igraph.org/r/) representing a totally connected graph. A number of visual displays can help interpreting the cell-state hierarchy. Figure 5 is generated with the following code:

```
# Plotting parameters
vertex.size=5;
edge.color="black";
edge.width=2;
vertex.label.cex=0.2;
vertex.label.dist=1
vertex.color=colorSCsByTime
vertex.label="";
layout.graph=layout.kamada.kawai;

par(bty="o",xaxs="i",yaxs="i",cex.axis=mycex-0.2,cex.main=mycex,cex.lab=mycex,
        las=1,mar=c(5.3,5.3,2.9,1.6),oma=c(1,1,2,10))
zones=matrix(c(1:4),ncol=2,byrow=FALSE)
graphics::layout(zones)
```

```
plot.igraph(cellstateHierarchy_MST ,main="MST",
        vertex.color=vertex.color,vertex.label=vertex.label,
        vertex.size=vertex.size,edge.color=edge.color,
        edge.width=edge.width,vertex.color=vertex.color,
        vertex.label.cex=vertex.label.cex,layout=layout.graph)

plot.igraph(cellstateHierarchy_SST_clustKNN ,main="SST - KNN cluster",
        vertex.color=vertex.color,vertex.label=vertex.label,
        vertex.size=vertex.size,edge.color=edge.color,
        edge.width=edge.width,vertex.color=vertex.color,
        vertex.label.cex=vertex.label.cex,layout=layout.graph)

plot.igraph(cellstateHierarchy_SST_clustKmedoids ,main="SST - K-medoids",
        vertex.color=vertex.color,vertex.label=vertex.label,
        vertex.size=vertex.size,edge.color=edge.color,
        edge.width=edge.width,vertex.color=vertex.color,
        vertex.label.cex=vertex.label.cex,layout=layout.graph)

plot.igraph(cellstateHierarchy_IMC ,main="IMC",
        vertex.color=vertex.color,vertex.label=vertex.label,
        vertex.size=vertex.size,edge.color=edge.color,
        edge.width=edge.width,vertex.color=vertex.color,
        vertex.label.cex=vertex.label.cex,layout=layout.graph)
```

Figure 5 represents four of the cell-state hierarchies calculated in Section 5.2. The color code corresponds to the time of differentiation of myoblast samples, as indicated in the legend. It can be observed that cells from time point 0h are highly homogeneous, clustering together in the hierarchy. However, Cells from time points 24, 48 and 72 hours are intertwined in the hierarchy. Indeed, some cells from different time points are more similar among them than among other cells from the same timepoint, suggesting a heterogeneity of cell-states (i.e. pseudotimes) within sample.

As previously described for the graphical representation of individual cells in low dimensional space (see Section 4.3), different color schemes can be used:

i) a continuous color scale reflecting the expression levels of a maker of choice:

```
# To color by marker, set following parameter to:
 vertex.color=colorSCsByMarker
```

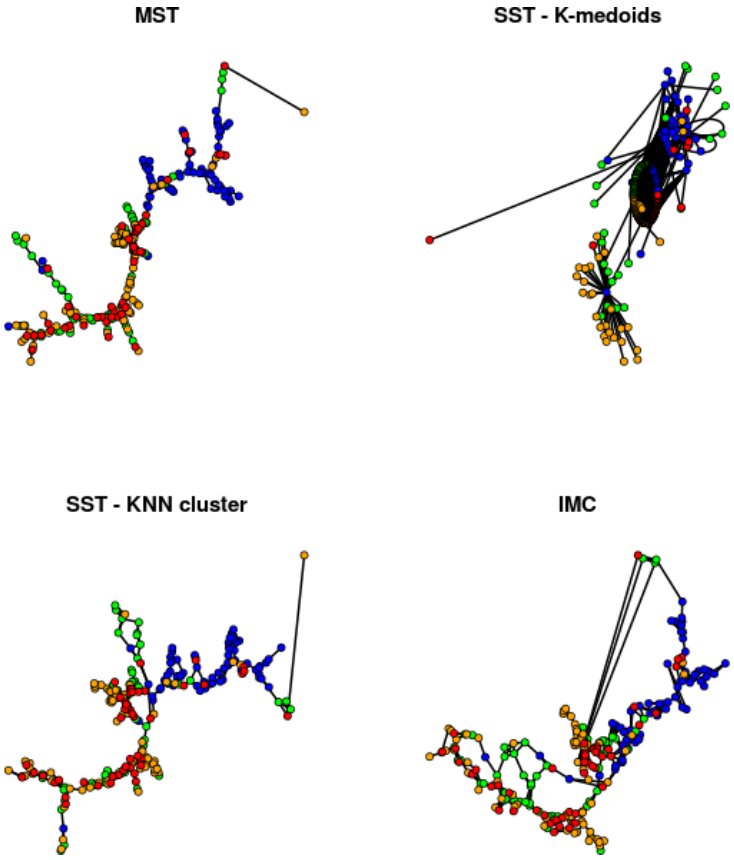

Figure 5: Representation of cell-state hierarchies assessed with different methods (see text)

and ii) a color code representing qualitative information about the samples (e.g. time points of differentiating myoblasts in Trapnell et al 2014: 0, 24, 48 and 72 hours)

```
# To color by time point:
vertex.color=colorSCsByTime
```

To represent individual cell identifiers in the graphs, set:

```
vertex.label=colnames(SO[["expressionmatrix"]]);
```

Different layout options can also be explored to better represent the graph:

a) Using the layout of cells provided by the low dimensional spaced used (if any) to assess the cell-to-cell distance matrix. For example:

```
layout=t(cellsLowDimensionalSpace_ICA[1:2,])
```

b) or using other igraph layouts, for instance:

```
layout=layout.fruchterman.reingold
layout=layout.reingold.tilford
layout=layout.graphopt
layout=layout.svd
layout=layout.lgl
```

5.4 Comparisson of hierarchies assessed by different algorithms on the same data

Sincell's function *sc_ComparissonOfGraphs()* provides a way to compare the graphs resulting from the different algorithms. The distance between two graphs is assessed as 1 minus their similarity, which is calculated as the Spearman rank correlation between the two graphs of the shortest distance for all pairs of cells. The function prints the distance matrix comparing the graphs and plots a hierarchical clustering of the hierarchies:

```
sc_ComparissonOfGraphs(
  cellstateHierarchy_MST,
  cellstateHierarchy_SST_clustKNN,
  cellstateHierarchy_SST_clustKmedoids,
  # graph.names=c("MST","SST-KNNcluster","SST-K-medoids"),
  cellstateHierarchy_IMC,
  graph.names=c("MST","SST-KNNcluster","SST-K-medoids","IMC")
)
```

```
##                   MST SST-KNNcluster SST-K-medoids
## SST-KNNcluster 0.0950
## SST-K-medoids  0.7597         0.7292
## IMC            0.0972         0.0409        0.6923
```

6 Algorithmic strategies to provide statistical support to cell-state hierarchies from single-cell RNAseq

The fact that a cell-state hierarchy is obtained by using previous algorithms does not necessarily imply that it reflects a true biological scenario of cell activation/differentiation. It might well be that the hierarchy obtained is mainly driven by noise due to either biological or technical factors. Furthermore, the

relative contribution of stochastic factors to the observed differences across cells is expected to be higher if cells within a sample are in a homogeneous steady-state. In that case, a low cell-to-cell heterogeneity will lead to cell-state hierarchies very sensitive to small variations in the initial gene expression data. On the other extreme, high levels of cell-to-cell heterogeneity driven by a real granularity in an activation/differentiation process will translate into robust hierarchies that can be reproduced despite stochastic perturbations of the data.

To help discriminating reliable cell-state hierarchies from noisy rearrangements, *Sincell* implements two algorithms: i) a strategy relying on a gene resampling procedure and ii) an algorithm based on random cell substitution with in silico-generated cell replicates.

As in previous sections, we illustrate the use of these algorithms on the four single-cell RNA-seq libraries for differentiating myoblasts at 0, 24, 48 and 72 hours generated by Trapnell et al 2014. However, this time we will analyse the 4 libraries independently and ask whether a cell-state hierarchy obtained from each of them separately is actually statistically supported. This mimics a scenario where a potential user analyses a single-cell library from one experiment (e.g. condition) and tries to figure out whether the biological sample contains heterogeneity of cell-states that might be represented in the form of a hierarchy with statistical support.

Let's obtain first the cell-state hierarchy for each time using the first two dimensions of a dimensionality reduction with Independent Component Analysis (ICA) and a Minimum Spanning Tree (MST)

```
t0 <- grep("T0_", colnames(EMlog))
t24 <- grep("T24_", colnames(EMlog))
t48 <- grep("T48_", colnames(EMlog))
t72 <- grep("T72_", colnames(EMlog))

EMlog_t0<-EMlog[,t0]
EMlog_t24<-EMlog[,t24]
EMlog_t48<-EMlog[,t48]
EMlog_t72<-EMlog[,t72]

dim(EMlog_t0)

## [1] 575  69

dim(EMlog_t24)

## [1] 575  74

dim(EMlog_t48)
```

```
## [1] 575  79

dim(EMlog_t72)

## [1] 575  49

SO_t0 <- sc_InitializingSincellObject(EMlog_t0)
SO_t0 <- sc_DimensionalityReductionObj(SO_t0, method="ICA",dim=2)
SO_t0 <- sc_GraphBuilderObj(SO_t0, graph.algorithm="MST",
        graph.using.cells.clustering=FALSE)

SO_t24 <- sc_InitializingSincellObject(EMlog_t24)
SO_t24 <- sc_DimensionalityReductionObj(SO_t24, method="ICA",dim=2)
SO_t24 <- sc_GraphBuilderObj(SO_t24, graph.algorithm="MST",
        graph.using.cells.clustering=FALSE)

SO_t48 <- sc_InitializingSincellObject(EMlog_t48)
SO_t48 <- sc_DimensionalityReductionObj(SO_t48, method="ICA",dim=2)
SO_t48 <- sc_GraphBuilderObj(SO_t48, graph.algorithm="MST",
        graph.using.cells.clustering=FALSE)

SO_t72 <- sc_InitializingSincellObject(EMlog_t72)
SO_t72 <- sc_DimensionalityReductionObj(SO_t72, method="ICA",dim=2)
SO_t72 <- sc_GraphBuilderObj(SO_t72, graph.algorithm="MST",
        graph.using.cells.clustering=FALSE)
```

Figure 6 represents the cell-state hierarchies calculated separately for each myoblast differentiation time:

```
# Plotting parameters
vertex.size=5;
edge.color="black";
edge.width=2;
vertex.label.cex=0.2;
vertex.label.dist=1
vertex.color=colorSCsByTime
vertex.label="";
layout.graph=layout.kamada.kawai;

par(bty="o",xaxs="i",yaxs="i",cex.axis=mycex-0.2,cex.main=mycex,cex.lab=mycex,
        las=1,mar=c(5.3,5.3,2.9,1.6),oma=c(1,1,2,10))
zones=matrix(c(1:4),ncol=2,byrow=FALSE)
graphics::layout(zones)
```

```
plot.igraph(SO_t0[["cellstateHierarchy"]],main="0 hours",
        vertex.color=ColorT0,vertex.label=vertex.label,
        vertex.size=vertex.size,edge.color=edge.color,
        edge.width=edge.width,vertex.color=vertex.color,
        vertex.label.cex=vertex.label.cex,layout=layout.graph)

plot.igraph(SO_t24[["cellstateHierarchy"]],main="24 hours",
        vertex.color=ColorT24,vertex.label=vertex.label,
        vertex.size=vertex.size,edge.color=edge.color,
        edge.width=edge.width,vertex.color=vertex.color,
        vertex.label.cex=vertex.label.cex,layout=layout.graph)

plot.igraph(SO_t48[["cellstateHierarchy"]],main="48 hours",
        vertex.color=ColorT48,vertex.label=vertex.label,
        vertex.size=vertex.size,edge.color=edge.color,
        edge.width=edge.width,vertex.color=vertex.color,
        vertex.label.cex=vertex.label.cex,layout=layout.graph)

plot.igraph(SO_t72[["cellstateHierarchy"]],main="72 hours",
        vertex.color=ColorT72,vertex.label=vertex.label,
        vertex.size=vertex.size,edge.color=edge.color,
        edge.width=edge.width,vertex.color=vertex.color,
        vertex.label.cex=vertex.label.cex,layout=layout.graph)
```

6.1 Statistical support of cell-state hierarchies by gene resampling

One strategy to provide statistical support to a connected graph representing a cell-state hierarchy, is the one implemented in *Sincell*'s function *sc_StatisticalSupportByGeneSubsampling()*. This function performs "s" times a random subsampling of a given number "n" of genes on the original gene expression matrix. Then, for each sampling, a new connected graph of cells is generated using the same parameters as for the hierarchy being tested. In each subsampling, the similarity between the resulting connected graph and the original one is assessed as the Spearman rank correlation between the two graphs of the shortest distance for all pairs of cells. The distribution of Spearman rank correlation values of all subsamplings might be interpreted as the distribution of similarities between hierarchies that would be obtained from small changes in the data. A distribution with a high median and small variance would indicate a well-supported cell-state hierarchy. On the

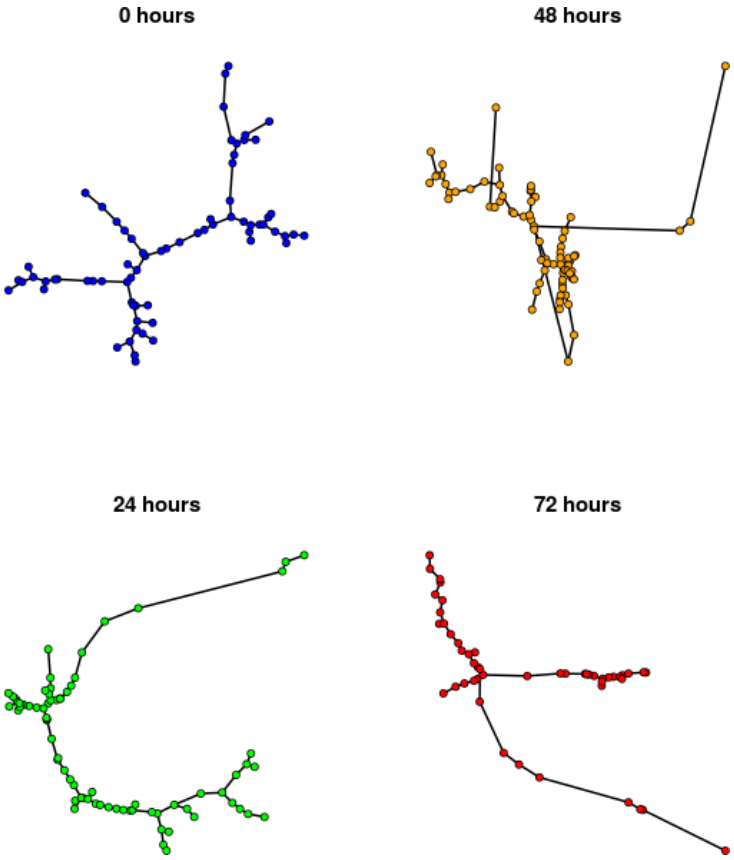

Figure 6: **Cell-state hierarchies of differentiating myoblast samples at 4 time points (0,24,48 and 72h) obtained through a dimensionality reduction with Independent Component Analysis (ICA, 2 dimensions) and a Minimum Spanning Tree (MST)**

contrary, a distribution with a low median of similarities and/or a wide variance would indicate a hierarchy very sensitive to changes in the data, and therefore not highly statistically supported.

To provide statistical support by gene subsampling to the cell-state hierarchies obtained in the differentiating myoblasts libraries at 0, 24, 48 and 72 hours, we may run:

```
# For the sake of time we set here num_it=100.
# For a higher significance, we recommend setting  num_it=1000
SO_t0<-sc_StatisticalSupportByGeneSubsampling(SO_t0, num_it=100)

## The summary of the distribution of Spearman rank correlations
## between the original hierarchy and the hierarchies obtained
## from 100 resamplings of 287 genes in the initial expression matrix is:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.552   0.805   0.883   0.855   0.922   0.956
```

```
SO_t24<-sc_StatisticalSupportByGeneSubsampling(SO_t24, num_it=100)

## The summary of the distribution of Spearman rank correlations
## between the original hierarchy and the hierarchies obtained
## from 100 resamplings of 287 genes in the initial expression matrix is:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.638   0.856   0.886   0.872   0.911   0.964

SO_t48<-sc_StatisticalSupportByGeneSubsampling(SO_t48, num_it=100)

## The summary of the distribution of Spearman rank correlations
## between the original hierarchy and the hierarchies obtained
## from 100 resamplings of 287 genes in the initial expression matrix is:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.773   0.906   0.921   0.917   0.937   0.971

SO_t72<-sc_StatisticalSupportByGeneSubsampling(SO_t72, num_it=100)

## The summary of the distribution of Spearman rank correlations
## between the original hierarchy and the hierarchies obtained
## from 100 resamplings of 287 genes in the initial expression matrix is:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.848   0.907   0.928   0.925   0.945   0.974
```

The distribution of Spearman rank correlation values of all subsampings is stored as a vector in SincellObject[["StatisticalSupportbyGeneSubsampling"]] and a summary is printed to the standard output.

We may now plot together the distribution corresponding to the hierarchies from the different time points (Figure 7):

```
mycex<-1
par(bty="o",xaxs="i",yaxs="i",cex.axis=mycex-0.2,cex.main=mycex,cex.lab=mycex,
        las=1,mar=c(5.3,5.3,2.9,1.6),oma=c(1,1,2,6))

plot(density(SO_t0[["StatisticalSupportbyGeneSubsampling"]]),col=ColorT0,lwd=4,
        main="Similarities of hierarchies \nupon gene subsamping",xlim=c(0.5,1),
        ylim=c(0,20),ylab="Density",xlab="Spearman rank correlation")
lines(density(SO_t24[["StatisticalSupportbyGeneSubsampling"]]),col=ColorT24,lwd=4)
lines(density(SO_t48[["StatisticalSupportbyGeneSubsampling"]]),col=ColorT48,lwd=4)
lines(density(SO_t72[["StatisticalSupportbyGeneSubsampling"]]),col=ColorT72,lwd=4)

par(fig = c(0, 1, 0, 1), oma = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
        new = TRUE, xpd = TRUE)
plot(0, 0, type = "n", bty = "n", xaxt = "n", yaxt = "n")
```

```
legend("right",title="Time\npoint", c("0h","24h","48h","72h"),
       fill= c(ColorT0,ColorT24,ColorT48,ColorT72),
       inset=c(0.03,0),  bty="n")
```

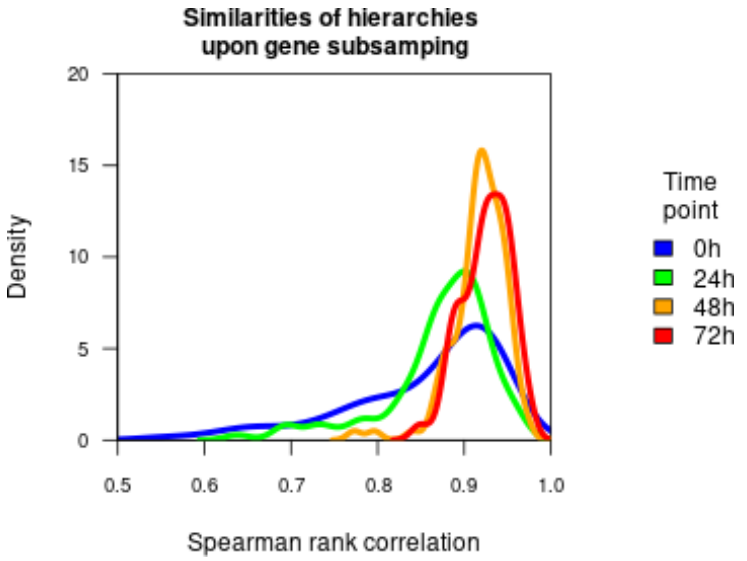

Figure 7: Similarities of hierarchies upon random gene subsampling
The figure represents the similarities between a given cell-state hierarchy and the hierarchies obtained when random sets of genes are subsampled. Four distributions are plotted corresponding to the hierarchies obtained from the libraries at different time points: 0,24,48 and 72 hours, as indicated in the legend. A high median and a low variance are indicative of a cell-state hierarchy robust to variations in the data. See text for details.

A summary of the distributions of similarities obtained upon subsampling can be printed as follows:

```
summary(SO_t0[["StatisticalSupportbyGeneSubsampling"]])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.552   0.805   0.883   0.855   0.922   0.956
```

```
summary(SO_t24[["StatisticalSupportbyGeneSubsampling"]])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.638   0.856   0.886   0.872   0.911   0.964
```

```
summary(SO_t48[["StatisticalSupportbyGeneSubsampling"]])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.773   0.906   0.921   0.917   0.937   0.971
```

```
summary(SO_t72[["StatisticalSupportbyGeneSubsampling"]])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.848   0.907   0.928   0.925   0.945   0.974
```

```
var(SO_t0[["StatisticalSupportbyGeneSubsampling"]])
```

```
## [1] 0.00807
```

```
var(SO_t24[["StatisticalSupportbyGeneSubsampling"]])
```

```
## [1] 0.00394
```

```
var(SO_t48[["StatisticalSupportbyGeneSubsampling"]])
```

```
## [1] 0.001
```

```
var(SO_t72[["StatisticalSupportbyGeneSubsampling"]])
```

```
## [1] 0.000762
```

Late time points (48 and 72h) lead to hierarchies with a high median while early time points (0 and 24h) had a lower median and a higher variance. Results suggest that at the initial stages of differentiation homogeneity of cell states is rather high, leading to a hierarchy sensitive to the set of genes in which it is assessed and therefore less statistically supported. However, late time point showed a hierarchy very robust to gene subsampling, reflecting a marked heterogeneity in cell-states characteristic of more mature differentiation stages.

6.2 Statistical support of cell-state hierarchies by random cell substitution with in silico-generated cell replicates

Gene expression levels detected by single-cell RNA seq are subject to stochastic factors both technical and biological. This means that, if it were possible to profile the same cell in the same cell-state multiple times (or, more realistically, a population of individual cells in a highly homogeneous state), the detected expression levels of a gene would randomly fluctuate within a distribution of values. In the ideal scenario where that distribution was known for each gene, individual cell replicates could be produced in silico, leading to variations in gene expression levels similar to what would be obtained from in vivo replicates. The generation of in silico replicates would then permit to test the reproducibility of the cell-state hierarchy upon random replacement of a fraction of the original cells with them.

6.2.1 Generation of in silico cell replicates

The stochastic distribution of the expression levels of a gene can be characterized by a measure of dispersion, e.g. the variance or the coefficient of variation. It is known that the expected variation is dependent on the mean expression values of the gene, so that the higher the mean, the higher the variance and the lower the coefficient of variation (Anders and Huber 2010; Brennecke et al 2013). Based on this, we can simulate a stochastic fluctuation of the expression of a gene by perturbing the observed level in a given cell with an error term whose magnitude is consistent with the mean-variance relationship observed in the data. By doing that in all genes from an individual cell Ci, we can produce an in silico replicate of it.

Sincell's function *sc_InSilicoCellsReplicatesObj()* implements this strategy as follows: first, the mean and variance of all genes in the original gene expression matrix is assessed. Genes are assigned to classes according to the deciles of mean they belong to. Next, for a given gene g, a variance v is randomly chosen from the set of variances within the class of the gene. Then, a random value drawn from a uniform distribution U(0,v) of mean zero and variance v is added to the expression value of a gene g in a cell c. By perturbing in this way all genes in a reference cell c we obtain an in silico replicate c'. Redoing the process N times, N stochastic replicates are generated for each original cell. If method= "cv2.deciles" in function *sc_InSilicoCellsReplicatesObj()*, a coefficient of variation cv2 is randomly chosen from the set of coefficient of variation values within the class of the gene. Then, the parameter v for the uniform distribution is assessed by v=cv2*(mean**2).

To generate 100 in-silico replicates of each individual cell in the differentiating myoblasts libraries at 0, 24, 48 and 72 hours, we may run:

```
# For the sake of time we set here multiplier=100.
# For a higher significance, we recommend setting multiplier=1000
SO_t0 <- sc_InSilicoCellsReplicatesObj(SO_t0,
        method="variance.deciles", multiplier=100)
SO_t24 <- sc_InSilicoCellsReplicatesObj(SO_t24,
        method="variance.deciles", multiplier=100)
SO_t48 <- sc_InSilicoCellsReplicatesObj(SO_t48,
        method="variance.deciles", multiplier=100)
SO_t72 <- sc_InSilicoCellsReplicatesObj(SO_t72,
        method="variance.deciles", multiplier=100)
```

The cell replicates are concatenated by columns to the original gene expression matrix and saved in the SincellObject[["InSilicoCellsReplicates"]].

Stochasticity in gene expression at the single-cell level has also been described as following a lognormal distribution log(x) ~N(m,v) (Bengtsson et al 2005; Raj et al. 2006). More recently, Shalek et al 2014 described gene expression variability in single-cell RNA-seq through a log normal distribution with a third parameter alpha describing the proportion of cells where transcript expression was detected above a given threshold level. Authors found that the majority of genes in their study (91%) showed distributions well described by the three-parameter model ($p < 0.01$, goodness of fit test; Shalek et al 2014). By setting parameter method="lognormal-3parameters" within *Sincell*'s function *sc_InSilicoCellsReplicatesObj()*, the function uses this "three parameter" model estimation to generate random perturbations of gene expression levels and produce in silico cell replicates.

Other works have found that, for most of the genes, the variability observed among their expression levels across individual cells was better described by a negative binomial (NB) distribution rather than a lognormal distribution (Gr?n et al., 2014). These authors used NB distribution to model not only technical noise but also true biological gene expression noise. Their assumption is that endogenous mRNA abundance follows a NB as supported by a physical model of bursting expression (Raj et al., 2006). A negative binomial noise model was also adopted in (Zeisel et al., 2015). As pointed out in these works, NB is frequently used to model overdispersed count data and has been previously used for bulk RNA-seq data (Anders and Huber, 2010; Robinson et al., 2010). We recommend this approach only if normalized count data is used (i.e. not length-normalized RPKM/FPKM). By setting parameter method="negative.binomial" within *Sincell*'s function *sc_InSilicoCellsReplicatesObj()*, *Sincell* can follow an NB distribution parameterized on the observed gene expression levels to generate random perturbations and produce in silico cell replicates accordingly.

6.2.2 Random cell substitution with in silico-generated cell replicates

Once cell-replicates have been generated, a second strategy to provide statistical support to a connected graph is provided by *Sincell*'s function *sc_StatisticalSupportByReplacementWithInSil*
This function performs "s" times a random replacement of a given number "n" cells on the original gene expression matrix with a randomly selected set of in-silico replicates. Then, for each set of substitutions "s", a new connected graph of cells is calculated using the same parameters as for the hierarchy being tested. In each "s", the similarity between the resulting connected graph and the original one is assessed as the Spearman rank correlation between the two graphs of the shortest distance for all pairs of cells. The distribution of Spearman rank correlation values of all replacements might be interpreted as the distribution of similarities between hierarchies that would be obtained from

stochastic perturbations of a proportion of cells. A distribution with a high median and small variance would indicate a well-supported cell-state hierarchy. On the contrary, a distribution with a low median of similarities and/or a wide variance would indicate a hierarchy very sensitive to changes in the data, and therefore not much statistically supported.

To provide statistical support by gene substitution with in silico-generated cell replicates to the cell-state hierarchies obtained in the differentiating myoblasts libraries at 0, 24, 48 and 72 hours, we may run:

```
SO_t0<-sc_StatisticalSupportByReplacementWithInSilicoCellsReplicates(SO_t0,
        num_it=100,fraction.cells.to.replace=1)

## The summary of the distribution of spearman rank correlations
##  between the original hierarchy and the hierarchies obtained from
##  substitution of in silico generated replicates of order own
## in the initial expression matrix is:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.572   0.794   0.846   0.833   0.895   0.960

SO_t24<-sc_StatisticalSupportByReplacementWithInSilicoCellsReplicates(SO_t24,
        num_it=100,fraction.cells.to.replace=1)

## The summary of the distribution of spearman rank correlations
##  between the original hierarchy and the hierarchies obtained from
##  substitution of in silico generated replicates of order own
## in the initial expression matrix is:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.730   0.860   0.886   0.881   0.908   0.963

SO_t48<-sc_StatisticalSupportByReplacementWithInSilicoCellsReplicates(SO_t48,
        num_it=100,fraction.cells.to.replace=1)

## The summary of the distribution of spearman rank correlations
##  between the original hierarchy and the hierarchies obtained from
##  substitution of in silico generated replicates of order own
## in the initial expression matrix is:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.834   0.896   0.915   0.913   0.930   0.969

SO_t72<-sc_StatisticalSupportByReplacementWithInSilicoCellsReplicates(SO_t72,
        num_it=100,fraction.cells.to.replace=1)

## The summary of the distribution of spearman rank correlations
##  between the original hierarchy and the hierarchies obtained from
##  substitution of in silico generated replicates of order own
## in the initial expression matrix is:
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.874   0.906   0.924   0.922   0.939   0.967
```

The distribution of Spearman rank correlation values of all subsampings is stored as a vector in the SincellObject[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]] and a summary is printed to the standard output.

We may now plot together the distribution corresponding to the hierarchies from the different time points (Figure 8):

```
mycex<-1
par(bty="o",xaxs="i",yaxs="i",cex.axis=mycex-0.2,cex.main=mycex,cex.lab=mycex,
        las=1,mar=c(5.3,5.3,2.9,1.6),oma=c(1,1,2,6))

plot(density(SO_t0[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]]),
        col=ColorT0,lwd=4,
        main="Similarities of hierarchies upon substitution\nwith in silico cell replicates
        xlim=c(0.5,1),ylim=c(0,20),ylab="Density",xlab="Spearman rank correlation")
lines(density(SO_t24[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]]),
        col=ColorT24,lwd=4)
lines(density(SO_t48[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]]),
        col=ColorT48,lwd=4)
lines(density(SO_t72[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]]),
        col=ColorT72,lwd=4)

par(fig = c(0, 1, 0, 1), oma = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
        new = TRUE, xpd = TRUE)
plot(0, 0, type = "n", bty = "n", xaxt = "n", yaxt = "n")
legend("right",title="Time\npoint", c("0h","24h","48h","72h"),
        fill= c(ColorT0,ColorT24,ColorT48,ColorT72),
        inset=c(0.03,0),  bty="n")
```

A summary of the distributions of similarities obtained upon subsampling can be printed as follows:

```
summary(SO_t0[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.572   0.794   0.846   0.833   0.895   0.960
```

```
summary(SO_t24[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.730   0.860   0.886   0.881   0.908   0.963
```

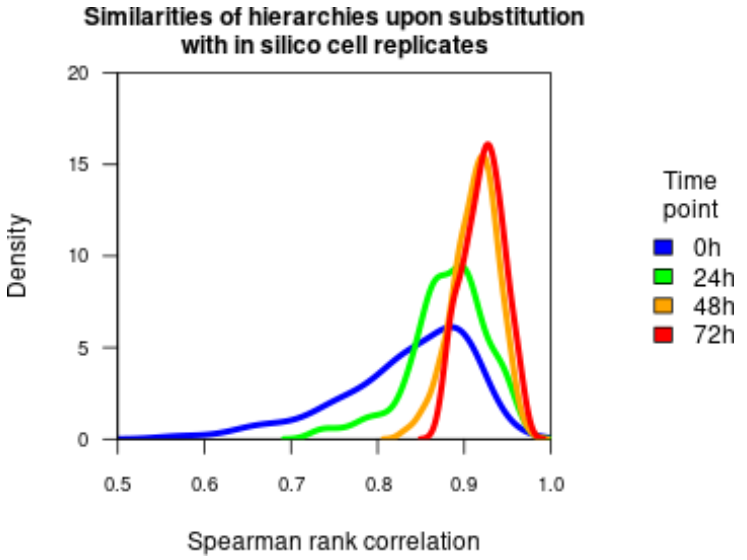

Figure 8: **Similarities of hierarchies upon random cell substitution with in silico-generated cell replicates**
The figure represents the similarities between a given cell-state hierarchy and the hierarchies obtained when a random fraction of individual cells (here 100%) are substituted by a randomly chosen in silico replicate of them. Four distributions are plotted corresponding to the hierarchies obtained from the libraries at different time points: 0, 24, 48 and 72 hours, as indicated in the legend. A high median and a low variance are indicative of a cell-state hierarchy robust to variations in the data. See text for details.

```
summary(SO_t48[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]])

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.834   0.896   0.915   0.913   0.930   0.969

summary(SO_t72[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]])

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.874   0.906   0.924   0.922   0.939   0.967

var(SO_t0[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]])

## [1] 0.00571

var(SO_t24[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]])

## [1] 0.00206

var(SO_t48[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]])

## [1] 0.000663

var(SO_t72[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]])
```

```
## [1] 0.000517
```

Consistent with results from the gene resampling procedure, time point 72 lead to the hierarchy with the highest median while time point 0 had the lowest median and the highest variance. Results again suggest that at 0 hours, homogeneity of cell states is high while at time point 72 hours, a higher degree of heterogeneity is present. Actually, a gradient along time can be observed (from 0 to 24, 48 and 72h) suggesting that heterogeneity in cell-states increased as a function of time.

6.3 Note on the spike-in molecules to deconvolute technical and biological noise.

The use of spike-in molecules is recommended in single-cell RNA-seq to infer the amount of variability in the ex-pression levels of one gene that is expected to arise from technical factors (Brennecke et al., 2013; Grün et al., 2014; Islam et al., 2014). More recently, Ding et al. went a step further by using spike-ins to explicitly remove technical noise and compute de-noised gene expression levels (R software GRM, http://wanglab.ucsd.edu/star/GRM/ , Ding et al., 2015). When spike-ins are available, we recommend performing first technical denoise of the expression matrix before using *Sincell*. In such a way, *Sincell*'s computations of cell-state hierarchies would mainly rely on biological variation.

Notwithstanding, biological variation will contain two main components. First, the heterogeneity explained by truly different cell-states in a differentiation/activation process, which is the component we aim to capture in a hierarchy. And second, the intrinsic biological noise that is expected to arise even among cells in the same differentiation/activation state. Intrinsic biological noise originates from the characteristics of gene expression mechanisms such as bursts of transcription (Raj et al., 2006), the stochasticity of signal transmission (Rand et al., 2012), bimodality (Shalek et al., 2013), cell cycle effects (McDavid et al., 2014; Buettner et al., 2015) and random monoallelic expression (Deng et al., 2014).

Therefore, even if technical noise has been removed, it is important to test whether the hierarchy is mainly determined by real cell-state heterogeneity or, on the contrary, it is an artifact of the intrinsic biological noise. To this end, the reproducibility of the cell-state hierarchy can be evaluated upon random perturbations of the de-noised expression levels. *Sincell*'s strategies described above based on either i) random gene subsampling or ii) random cell substitution

with in silico-generated cell replicates, would in that case test whether the hierarchy is robust to stochastic perturbations of the data mimicking the effect of intrinsic biological noise.

Additionally, the user could eventually wish to test the reproducibility of the hierarchy assessed on the original expression data (i.e. not performing first a technical de-noising step) upon random perturbations exclusively based on technical noise. Estimates of technical noise for each gene can be obtained by modeling the dependence of the coefficient of variation (cv2) of spike-in molecules as a function of their average expression. For instance, in Brennecke et al. 2013, for each technical gene i (e.g. the spike-ins), the sample mean (m) and sample variance of its normalized counts are estimated. Then, the observed squared coefficients of variation (cv2) are fitted against the sample mean (m) with a generalized linear model of the gamma family with identity link and parameterization cv2=a1/m+ alpha0. Applying the fitted formula to the sample mean expression levels of a gene provides an estimate of cv2 arising from technical noise, leading to a vector so-called here cv2.estimated. By setting parameter method="negative.binomial", and dispersion.statistic=cv2.estimated within *Sincell*'s function *sc_InSilicoCellsReplicatesObj()*, *Sincell* permits to incorporate a technical cv2 estimate per gene in the assessment of in silico cell replicates based on normalized counts (i.e. following the previously described negative binomial distribution whose dispersion is parameterized using the estimated technical cv2).

Alternatively, in the absence of spike-in molecules, by setting parameter method="negative.binomial", and dispersion.statistic="cv2.fitted.to.data" within *Sincell*'s function *sc_InSilicoCellsReplicatesObj()* *Sincell* implements the fit described in Brennecke et al. 2013 using the cv2 and m values of all genes in the input expression matrix to provide a surrogate of technical noise estimates. However, this alternative should not be used if the user has previously followed our recommendation in Section 1 of using such an approach to identify highly variable genes in order to decrease the size of the input matrix (http://pklab.med.harvard.edu/scw2014/subpop_tutorial.html; Section "Identifying highly variable genes").

7 Functional association tests to help interpreting cell-state hierarchies

Once a cell-state hierarchy has been assessed and its statistical support checked, the next step is interpreting the hierarchy in functional terms. In Section 4.3 and Section 5.3 we have shown different graphical representations that can help interpreting the hierarchies in terms of the features of the samples/libraries (e.g. differentiation time) or the expression levels of markers of interest (Figures **??**,

4, and 5). In this section, we propose an analytical approach to test whether the cell-state hierarchy associates with a given functional gene set, that is: whether the relative similarities among the individual cells in the hierarchy are driven by the expression levels of a subset of genes with a common functional feature.

Sincell's function *sc_AssociationOfCellsHierarchyWithAGeneSet()* implements an algorithmic strategy to evaluate that association. First, the function calculates a new cell-state hierarchy where only the expression levels of the genes in a given functional gene set are considered. Second, it calculates the similarity of that hierarchy and the reference hierarchy (the one assessed on the initial gene expression matrix). The similarity between the two hierarchies is computed as the Spearman rank correlation between the two graphs of the shortest distance for all pairs of cells. Third, the function provides an empirical p-value of the observed similarity between the two hierarchies. This empirical p-value is derived from a distribution of similarities resulting from random samplings of gene sets of the same size.

```
# help(sc_AssociationOfCellsHierarchyWithAGeneSet())
```

Sincell's function *sc_AssociationOfCellsHierarchyWithAGeneSet()* is particularly suited to evaluate gene set collections from the Molecular Signatures Database (MSigDB) of the Broad Institute http://www.broadinstitute.org/gsea/msigdb/collections.jsp). Here, we illustrate the use of this function for evaluating the association of the previously obtained hierarchy for time point 72 hours (Section 6) with the gene sets derived from the Reactome pathway database available at MSigDB (http://www.broadinstitute.org/gsea/msigdb/download_file.jsp?filePath=/resources/msigdb/4.0/c2.cp.reactome.v4.0.symbols.gmt).

We first download gene set c2.cp.reactome.v4.0.symbols.gmt from MSigDB

```
# geneset.list <- lapply(strsplit(readLines("c2.cp.reactome.v4.0.symbols.gmt"),"\t"),
# as.character)
# geneset.list  is provided here for illustrative purposes:
data(geneset.list)
head(geneset.list[[1]])
```

```
## [1] "REACTOME_GLYCOGEN_BREAKDOWN_GLYCOGENOLYSIS"
## [2] "http://www.broadinstitute.org/gsea/msigdb/cards/REACTOME_GLYCOGEN_BREAKDOWN_GLYCOGE
## [3] "AGL"
## [4] "GYG1"
## [5] "PGM1"
## [6] "PHKA1"
```

```
# We establish a minimum gene set size overlap with the genes in
# the expression matrix
```

```
minimum.geneset.size=30
myAssociationOfCellsHierarchyWithAGeneSet<-list()
for (i in 1:length(geneset.list)){
  if(sum(rownames(SO_t72[["expressionmatrix"]])
        %in% geneset.list[[i]])>=minimum.geneset.size){

    SO_t72<-sc_AssociationOfCellsHierarchyWithAGeneSet(SO_t72,geneset.list[[i]],
        minimum.geneset.size=minimum.geneset.size,p.value.assessment=TRUE,
        spearman.rank.threshold=0.5,num_it=1000)
    myAssociationOfCellsHierarchyWithAGeneSet[[
        as.character(i)]]<-list()
    myAssociationOfCellsHierarchyWithAGeneSet[[
        as.character(i)]][["AssociationOfCellsHierarchyWithAGeneSet"]]<-
                SO[["AssociationOfCellsHierarchyWithAGeneSet"]]
    myAssociationOfCellsHierarchyWithAGeneSet[[
        as.character(i)]][["AssociationOfCellsHierarchyWithAGeneSet.pvalue"]]<-
        SO[["AssociationOfCellsHierarchyWithAGeneSet.pvalue"]]
  }
}

## The spearman rank correlation between the original hierarchy
## and the hierarchy obtained when using only the 30 genes
## common with gene list REACTOME_IMMUNE_SYSTEM is
## r= 0.657973032847333
##  with an empirical p-value= 4.49e-01
## drawn from 1000 random subsamplings of equal gene set size= 30

length(names(myAssociationOfCellsHierarchyWithAGeneSet))

## [1] 8
```

The previous code checks the association of the hierarchy SO_t72 with all Reactome pathways. If the association with a pathway has a Spearman rank correlation higher than 0.5, an empirical p.value is calculated and a summary of the results is printed. If no gene set passes the Spearman correlation threshold, no messages are generated. The content of all tests is stored in the list "myAssociationOfCellsHierarchyWithAGeneSet". In this particular case, only the pathway REACTOME_IMMUNE_SYSTEM showed a high correlation, though the empirical p-value was not significant.

To test association with Gene Ontology terms. The following MSigDB collections can be tested: - GO Biological process: c5.bp.v4.0.symbols.gmt - GO Molecular function: c5.mf.v4.0.symbols.gmt - GO Cellular component: c5.cc.v4.0.symbols.gmt

8 References

Amir,E.D. et al. (2013) viSNE enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia. Nat. Biotechnol., 31, 545?552.

Anders S, Huber W (2010) Differential expression analysis for sequence count data. Genome Biology 11: R106.

Bengtsson,M. et al. (2005) Gene expression profiling in single cells from the pancreatic islets of Langerhans re-veals lognormal distribution of mRNA levels. Genome Res., 15, 1388?1392.

Brennecke,P. et al. (2013) Accounting for technical noise in single-cell RNA-seq experiments. Nat. Methods, 10, 1093?1095.

Buettner,F. et al. (2015) Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. Nat. Biotechnol., 33, 155?160.

Deng,Q. et al. (2014) Single-Cell RNA-Seq Reveals Dynamic, Random Monoallelic Gene Ex-pression in Mammalian Cells. Science, 343, 193?196.

Ding,B. et al. (2015) Normalization and noise reduction for single cell RNA-seq experiments. Bioinformatics, btv122.

Gr?n,D. et al. (2014) Validation of noise models for single-cell transcriptomics. Nat. Methods, 11, 637?640.

Islam,S. et al. (2014) Quantitative single-cell RNA-seq with unique molecular identifiers. Nat. Methods, 11, 163?166.

McDavid,A. et al. (2014) Modeling Bi-modality Improves Characterization of Cell Cycle on Gene Expression in Single Cells. PLoS Comput Biol, 10, e1003696.

Raj, A., Peskin, C. S., Tranchina, D., Vargas, D. Y., Tyagi, S. (2006) Stochastic mRNA Synthesis in Mammalian Cells. PLoS Biol 4, e309.

Rand,U. et al. (2012) Multi-layered stochasticity and paracrine signal propagation shape the type-I interferon response. Mol. Syst. Biol., 8.

Robinson,M.D. et al. (2010) edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics, 26, 139?140.

Salipante,S.J. and Hall,B.G. (2011) Inadequacies of Minimum Spanning Trees in Molecular Epidemiology. J. Clin. Microbiol., 49, 3568?3575.

Shalek,A.K. et al. (2013) Single-cell transcriptomics reveals bimodality in expression and splic-ing in immune cells. Nature.

Trapnell,C. et al. (2014) The dynamics and regulators of cell fate decisions are revealed by pseudotemporal or-dering of single cells. Nat. Biotechnol., 32, 381?386.

Zeisel,A. et al. (2015) Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. Science, 347, 1138?1142.

9 Session Info

```
sessionInfo()
```

```
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.2 LTS
##
## Matrix products: default
## BLAS:   /home/biocbuild/bbs-3.13-bioc/R/lib/libRblas.so
## LAPACK: /home/biocbuild/bbs-3.13-bioc/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_GB              LC_COLLATE=C
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
##  [1] splines   stats4    parallel  stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] biomaRt_2.48.0      monocle_2.20.0      DDRTree_0.1.5
##  [4] irlba_2.3.3         VGAM_1.1-5          ggplot2_3.3.3
##  [7] Biobase_2.52.0      BiocGenerics_0.38.0 Matrix_1.3-3
## [10] sincell_1.24.0      igraph_1.2.6        knitr_1.33
##
## loaded via a namespace (and not attached):
##   [1] Rtsne_0.15             colorspace_2.0-1       ellipsis_0.3.2
##   [4] XVector_0.32.0         proxy_0.4-25           ggrepel_0.9.1
##   [7] bit64_4.0.5            AnnotationDbi_1.54.0   fansi_0.4.2
##  [10] xml2_1.3.2             codetools_0.2-18       docopt_0.7.1
```

```
##  [13] cachem_1.0.5           spam_2.6-0             entropy_1.3.0
##  [16] cluster_2.1.2          dbplyr_2.1.1           png_0.1-7
##  [19] pheatmap_1.0.12        BiocManager_1.30.15    compiler_4.1.0
##  [22] httr_1.4.2             assertthat_0.2.1       fastmap_1.1.0
##  [25] limma_3.48.0           htmltools_0.5.1.1      prettyunits_1.1.1
##  [28] tools_4.1.0            dotCall64_1.0-1        gtable_0.3.0
##  [31] glue_1.4.2             GenomeInfoDbData_1.2.6 RANN_2.6.1
##  [34] reshape2_1.4.4         dplyr_1.0.6            rappdirs_0.3.3
##  [37] maps_3.3.0             Rcpp_1.0.6             slam_0.1-48
##  [40] vctrs_0.3.8            Biostrings_2.60.0      iterators_1.0.13
##  [43] xfun_0.23              stringr_1.4.0          lifecycle_1.0.0
##  [46] statmod_1.4.36         XML_3.99-0.6           MASS_7.3-54
##  [49] zlibbioc_1.38.0        scales_1.1.1           TSP_1.1-10
##  [52] BiocStyle_2.20.0       hms_1.1.0              RColorBrewer_1.1-2
##  [55] fields_12.3            curl_4.3.1             yaml_2.2.1
##  [58] memoise_2.0.0          gridExtra_2.3          fastICA_1.2-2
##  [61] stringi_1.6.2          RSQLite_2.2.7          highr_0.9
##  [64] S4Vectors_0.30.0       foreach_1.5.1          filelock_1.0.2
##  [67] densityClust_0.3       GenomeInfoDb_1.28.0    rlang_0.4.11
##  [70] pkgconfig_2.0.3        matrixStats_0.58.0     bitops_1.0-7
##  [73] qlcMatrix_0.9.7        evaluate_0.14          lattice_0.20-44
##  [76] purrr_0.3.4            bit_4.0.4              tidyselect_1.1.1
##  [79] plyr_1.8.6             magrittr_2.0.1         R6_2.5.0
##  [82] magick_2.7.2           IRanges_2.26.0         generics_0.1.0
##  [85] combinat_0.0-8         DBI_1.1.1              pillar_1.6.1
##  [88] withr_2.4.2            KEGGREST_1.32.0        scatterplot3d_0.3-41
##  [91] RCurl_1.98-1.3         tibble_3.1.2           crayon_1.4.1
##  [94] utf8_1.2.1             BiocFileCache_2.0.0    rmarkdown_2.8
##  [97] viridis_0.6.1          progress_1.2.2         grid_4.1.0
## [100] blob_1.2.1             FNN_1.1.3              HSMMSingleCell_1.11.0
## [103] sparsesvd_0.2          digest_0.6.27          munsell_0.5.0
## [106] viridisLite_0.4.0
```