TSCAN: Tools for Single-Cell ANalysis

Zhicheng Ji

Johns Hopkins University,
Baltimore, Maryland, USA
zji4@jhu.edu

Hongkai Ji

Johns Hopkins University,
Baltimore, Maryland, USA
hji@jhsph.edu

April 28, 2026

Contents

1 Introductions

Single-cell high-throughput transcriptomics is a powerful approach to investigate the heterogeneity of gene expression activities on cell level, which is otherwise hard to detect by bulk transcriptomics experiments. Many of the single-cell transcriptomics researches focus on the differentiation of various cell types and hope to find how the expression of genes and key regulatory factors changes over the differentiation process [1,2]. For individual cells during the differentiation process, each cell may experience quite varying schedule of transcriptional changes even though the overall trajectory of changes is highly identical. To depict such variation between individual cells, it is essential to reconstruct the differentiation lineage and order each single cell to the correct cellular states on the lineage.

Several computational methods specifically tailored for single-cell experiments have been proposed to reconstruct the trend of cellular transitions during differentiation. For example, SPADE [3] uses an unsupervised spanning-tree progression approach to organize cells in a hierarchy of related phenotypes when applied

to flow cytometry data. Such approach can reveal complex cellular hierarchies of differentiation and rare cell states. Similarly, Monocle [4] also takes advantages of unsupervised minimum spanning tree to order whole-transcriptome profiles of single cells along 'pseudotime', a hypothesized time course which can quantitatively measure the real biological process of differentiation. This pseudotime course is then used to study how gene expressions change over the differentiation process.

If we assume the validity of the pseudotemporal ordering approach, several problems still remain to be solved. First, it is obvious that there could be numerous ways to construct such pseudotime course for a given single-cell transcriptome profile. Thus how to evaluate these pseudotime courses in an unbiased way is a critical issue before any follow-up analysis. Second, a completely unsupervised learning strategy (e.g. without using any prior biological information of certain marker genes) may lead to quite unreliable results. In Monocle's case, since the MST (minimum spanning tree) algorithm cannot automatically designate a starting or ending point of the tree, it is possible that the MST will falsely switch the starting and ending point of the true biological process and lead to the opposite results. As reliable prior biological information do exist for most biological system in which single-cell experiments are conducted, we hypothesize that these information could be incorporated into constructing the pseudotime course to generate more reliable results.

To address the first problem, we propose pseudotemporal ordering score (POS), a quantitative measure of the reliability of numerous possible pseudotime course. We also propose a new pseudotime course construction method, travelling salesman problem algorithm (TSP), which aims to find a pseudotime course to minimize the total distance of linking the transcriptome profiles of all single cells. Using POS, we argue that TSP is more reliable than the MST algorithm in several testing datasets.

A new software, TSCAN, is proposed to faciliate the consturction of pseudotemporal cell ordering using TSP algorithm on single-cell RNA-seq data. TSCAN first uses principal component analysis to perform dimension reduction on the preprocessed gene expression data. Then the travelling salesman problem (TSP) algorithm is applied to construct a pseudotemporal ordering of cells so that the total distance along the path is nearly the shortest one. To address the problem that TSP does not have a designated starting/ending point, users can provide prior biological information of gene expression to select the optimal starting point of the ordering. After the pseudotemporal ordering and pseudotime are constructed, TSCAN can search for differentially expressed genes.

TSCAN comes with a user-friendly GUI written in shiny which provides powerful and convenient functions to tune pseudotemporal ordering. For example, users can trim unwanted cells or branches after an initial construction of pseudotime ordering. Users can use multiple marker genes to tune the starting point of the TSP path. Users can also easily calculate POS for different cell orderings using TSCAN GUI by uploading cell ordering and sub-population information. Many new features can be added to TSCAN in the future. Most importantly, TSCAN will be able to analyze single-cell experiments data other than gene ex-

pression data and construct pseudotemporal ordering correspondingly. TSCAN will also be able to automatically determine the change points of the differentially expressed genes and order the genes according to these change points. Genes with similar change points and change patterns may provide interesting biological findings about the biological processes in the single-cell experiments. Now we use the single-cell RNA-seq data for BMDC cells before and after 6h of LPS stimulation [5] to demonstrate how TSCAN can be used to construct pseudotemporal cell ordering using TSP.

2 Preprocess Gene Expression Profiles

The function preprocess can be used to filter out genes which are not expressed in most cells or not differentially expressed.

```
library(TSCAN)
```

```
## Loading required package: SingleCellExperiment
## Loading required package: SummarizedExperiment
## Loading required package: MatrixGenerics
## Loading required package: matrixStats
##
## Attaching package: 'MatrixGenerics'
## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars
## Loading required package: GenomicRanges
## Loading required package: stats4
## Loading required package: BiocGenerics
## Loading required package: generics
##
## Attaching package: 'generics'
```

```
## The following objects are masked from 'package:base':
##
##     as.difftime, as.factor, as.ordered, intersect, is.element, setdiff,
##     setequal, union
##
## Attaching package:  'BiocGenerics'
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
## The following objects are masked from 'package:base':
##
##     Filter, Find, Map, Position, Reduce, anyDuplicated, aperm, append,
##     as.data.frame, basename, cbind, colnames, dirname, do.call,
##     duplicated, eval, evalq, get, grep, grepl, is.unsorted, lapply,
##     mapply, match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     rank, rbind, rownames, sapply, saveRDS, table, tapply, unique,
##     unsplit, which.max, which.min
## Loading required package:  S4Vectors
##
## Attaching package:  'S4Vectors'
## The following object is masked from 'package:utils':
##
##     findMatches
## The following objects are masked from 'package:base':
##
##     I, expand.grid, unname
## Loading required package:  IRanges
## Loading required package:  Seqinfo
## Loading required package:  Biobase
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'.  To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
##
## Attaching package:  'Biobase'
## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians
## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians
## Loading required package:  TrajectoryUtils
data(lpsdata)
procdata <- preprocess(lpsdata)
```

By default, preprocess will first take log2 on the original dataset after adding a pseudocount of 1. Then the function will rule out genes which have expression values of less than 1 in at least half of all cells. Genes whose coefficient of covariance of the expression values is less than 1 are also filtered out.

3 Constructing Pseudotime

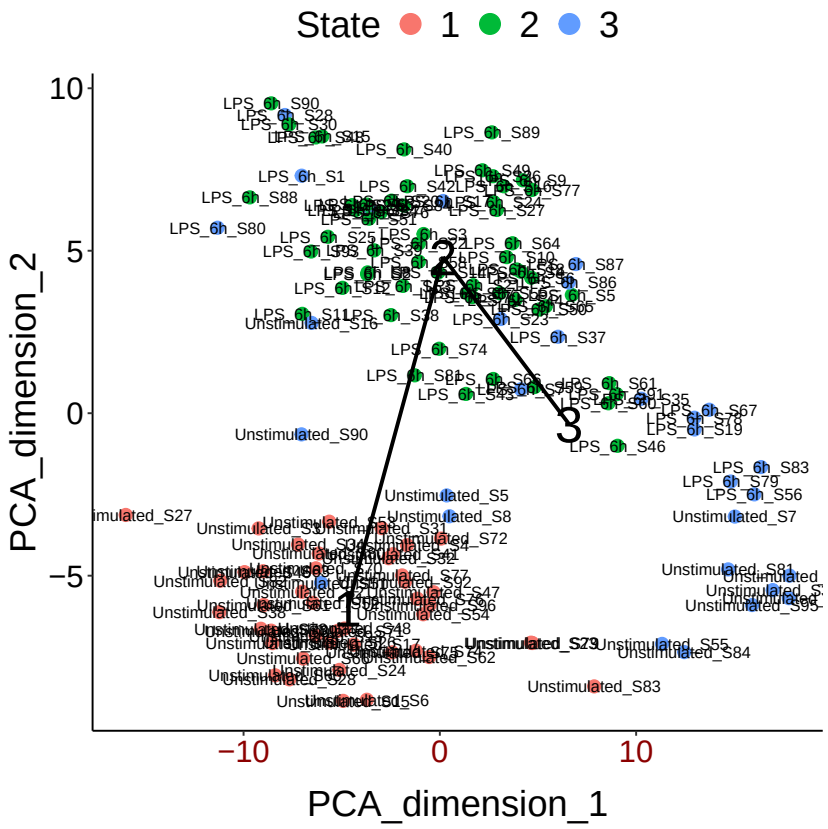
The function exprmclust will perform dimension reduction using principal component analysis and model-based clustering.

```
lpsmclust <- exprmclust(procdata)
```

Use plotmclust function to visualize the clustering based on the data after dimension reduction.

```
plotmclust(lpsmclust)
```

```
## Warning:  `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## i The deprecated feature was likely used in the TSCAN package.
##   Please report the issue to the authors.
## This warning is displayed once per session.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning
was
## generated.
## Warning:  Using `size` aesthetic for lines was deprecated in ggplot2
3.4.0.
## i Please use `linewidth` instead.
## i The deprecated feature was likely used in the TSCAN package.
##   Please report the issue to the authors.
## This warning is displayed once per session.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning
was
## generated.
```



Use the TSCANorder function to obtain the TSCAN ordering.

```
lpsorder <- TSCANorder(lpsmclust)
lpsorder
```

```
##   [1] "Unstimulated_S60" "Unstimulated_S38" "Unstimulated_S28"
##   [4] "Unstimulated_S93" "Unstimulated_S66" "Unstimulated_S61"
##   [7] "Unstimulated_S49" "Unstimulated_S39" "Unstimulated_S27"
##  [10] "Unstimulated_S46" "Unstimulated_S78" "Unstimulated_S15"
##  [13] "Unstimulated_S82" "Unstimulated_S6"  "Unstimulated_S63"
##  [16] "Unstimulated_S71" "Unstimulated_S52" "Unstimulated_S2"
##  [19] "Unstimulated_S26" "Unstimulated_S24" "Unstimulated_S48"
##  [22] "Unstimulated_S3"  "Unstimulated_S80" "Unstimulated_S75"
##  [25] "Unstimulated_S34" "Unstimulated_S70" "Unstimulated_S17"
##  [28] "Unstimulated_S76" "Unstimulated_S54" "Unstimulated_S92"
##  [31] "Unstimulated_S74" "Unstimulated_S62" "Unstimulated_S53"
```

```
##  [34] "Unstimulated_S77" "Unstimulated_S31" "Unstimulated_S32"
##  [37] "Unstimulated_S47" "Unstimulated_S96" "Unstimulated_S72"
##  [40] "Unstimulated_S41" "Unstimulated_S23" "Unstimulated_S4"
##  [43] "Unstimulated_S79" "Unstimulated_S83" "LPS_6h_S11"
##  [46] "LPS_6h_S81"       "LPS_6h_S12"       "LPS_6h_S88"
##  [49] "LPS_6h_S93"       "LPS_6h_S38"       "LPS_6h_S2"
##  [52] "LPS_6h_S53"       "LPS_6h_S8"        "LPS_6h_S25"
##  [55] "LPS_6h_S51"       "LPS_6h_S63"       "LPS_6h_S39"
##  [58] "LPS_6h_S68"       "LPS_6h_S29"       "LPS_6h_S48"
##  [61] "LPS_6h_S15"       "LPS_6h_S30"       "LPS_6h_S76"
##  [64] "LPS_6h_S90"       "LPS_6h_S20"       "LPS_6h_S42"
##  [67] "LPS_6h_S14"       "LPS_6h_S40"       "LPS_6h_S84"
##  [70] "LPS_6h_S3"        "LPS_6h_S74"       "LPS_6h_S58"
##  [73] "LPS_6h_S22"       "LPS_6h_S70"       "LPS_6h_S21"
##  [76] "LPS_6h_S43"       "LPS_6h_S24"       "LPS_6h_S55"
##  [79] "LPS_6h_S66"       "LPS_6h_S49"       "LPS_6h_S89"
##  [82] "LPS_6h_S16"       "LPS_6h_S57"       "LPS_6h_S26"
##  [85] "LPS_6h_S4"        "LPS_6h_S44"       "LPS_6h_S9"
##  [88] "LPS_6h_S27"       "LPS_6h_S18"       "LPS_6h_S10"
##  [91] "LPS_6h_S64"       "LPS_6h_S5"        "LPS_6h_S77"
##  [94] "LPS_6h_S59"       "LPS_6h_S6"        "LPS_6h_S65"
##  [97] "LPS_6h_S50"       "LPS_6h_S61"       "LPS_6h_S91"
## [100] "LPS_6h_S60"       "LPS_6h_S46"       "LPS_6h_S80"
## [103] "LPS_6h_S28"       "LPS_6h_S1"        "Unstimulated_S16"
## [106] "Unstimulated_S90" "LPS_6h_S34"       "LPS_6h_S17"
## [109] "Unstimulated_S5"  "LPS_6h_S7"        "LPS_6h_S23"
## [112] "Unstimulated_S51" "Unstimulated_S8"  "LPS_6h_S37"
## [115] "LPS_6h_S86"       "LPS_6h_S87"       "LPS_6h_S56"
## [118] "LPS_6h_S83"       "Unstimulated_S84" "LPS_6h_S79"
## [121] "Unstimulated_S55" "Unstimulated_S81" "LPS_6h_S35"
## [124] "Unstimulated_S95" "Unstimulated_S18" "Unstimulated_S85"
## [127] "Unstimulated_S37" "LPS_6h_S78"       "LPS_6h_S67"
## [130] "LPS_6h_S19"       "Unstimulated_S7"
```

4 Testing Differentially Expressed Genes

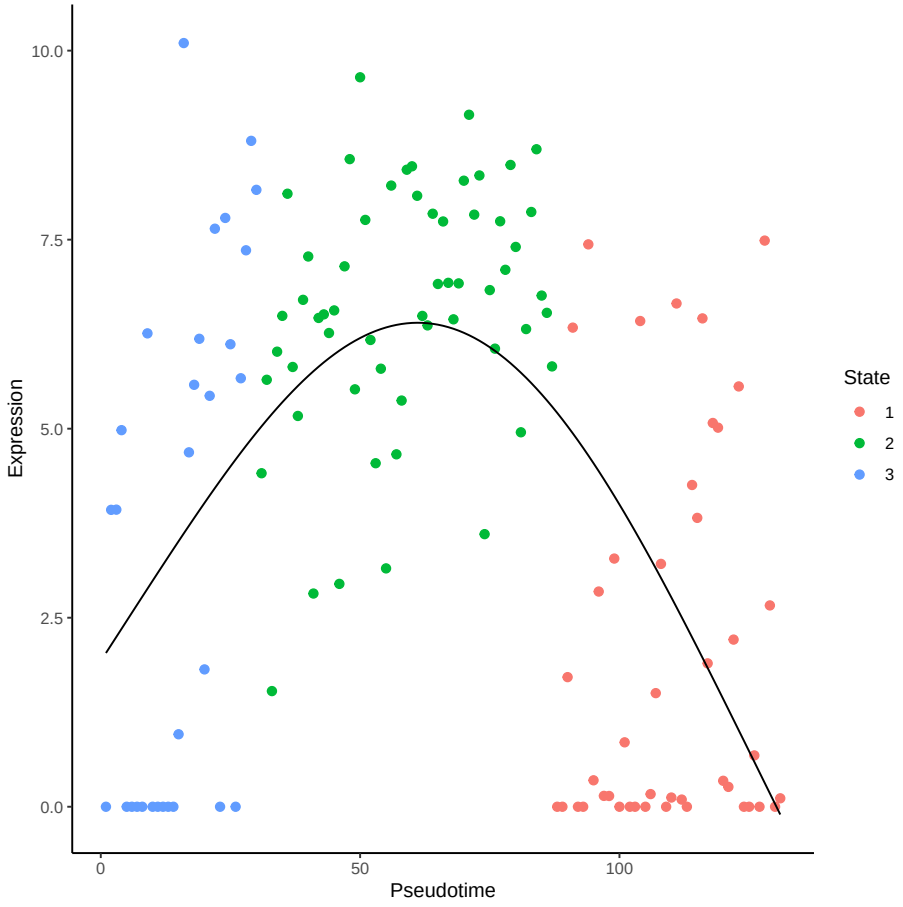
Use difftest function to detect differentially expressed genes given a constructed pseudotemporal ordering. The function will compare fitting a generalized additive model (gam) and fitting a constant. If gam fits significantly better than a constant fit than the gene is supposed to be differentially expressed. Original p-values as well as q-values after adjusting for multiple testing will be the output.

```
diffval <- difftest(procdata,lpsorder)
#Selected differentially expressed genes under qvlue cutoff of 0.05
head(row.names(diffval)[diffval$qval < 0.05])

## [1] "STX6"     "MRPL28"   "CUTA"     "AI413582" "SNRPC"    "MTCH1"
```

Use singlegeneplot function to plot the expression value of a single gene against a given pseudotime. Notice that here orderonly should be set to FALSE.

```
STAT2expr <- log2(lpsdata["STAT2",]+1)
singlegeneplot(STAT2expr, TSCANorder(lpsmclust,flip=TRUE,orderonly=FALSE))
```



5 Comparing Different Pseudotemporal Ordering

Given sub-population information of the single-cell information, function orderscore can calculate the pseudotemporal ordering score (POS) of multiple pseudotemporl ordering.
First prepare the ordering information. We want to compare TSP ordering with or without marker gene information.

```
subpopulation <- data.frame(cell = colnames(procdata), sub = ifelse(grepl("Unstimulated",col
#Comparing ordering with or without marker gene information
order1 <- TSCANorder(lpsmclust)
order2 <- TSCANorder(lpsmclust, c(1,2,3))
orders <- list(order1,order2)
```

Run orderscore function to get the comparison result:

```
orderscore(subpopulation, orders)

## [1] 0.5841631 0.5841631
```

From the previous example we can see that when prior information of MYOG gene expression is available, POS will dramatically increase compared to the ordering without MYOG gene expression information.

6 Reference

[1] Bendall, S. C., Simonds, E. F., Qiu, P., El-ad, D. A., Krutzik, P. O., Finck, R., ... & Nolan, G. P. (2011). Single-cell mass cytometry of differential immune and drug responses across a human hematopoietic continuum. Science, 332(6030), 687-696.
[2] Tang, F., Barbacioru, C., Bao, S., Lee, C., Nordman, E., Wang, X., ... & Surani, M. A. (2010). Tracing the derivation of embryonic stem cells from the inner cell mass by single-cell RNA-Seq analysis. Cell stem cell, 6(5), 468-478.
[3] Qiu, P., Simonds, E. F., Bendall, S. C., Gibbs Jr, K. D., Bruggner, R. V., Linderman, M. D., ... & Plevritis, S. K. (2011). Extracting a cellular hierarchy from high-dimensional cytometry data with SPADE. Nature biotechnology, 29(10), 886-891.
[4] Trapnell, C., Cacchiarelli, D., Grimsby, J., Pokharel, P., Li, S., Morse, M., ... & Rinn, J. L. (2014). The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. Nature biotechnology.
[5] Shalek, A. K., Satija, R., Shuga, J., Trombetta, J. J., Gennert, D., Lu, D., ... & Regev, A. (2014). Single-cell RNA-seq reveals dynamic paracrine control of cellular variation. Nature.