Lamian: a statistical framework for differential pseudotime analysis in multiple single-cell RNA-seq

true true true true true true

Overview

Pseudotime analysis based on single-cell RNA-seq (scRNA-seq) data has been widely used to study dynamic gene regulatory programs along continuous biological processes such as cell differentiation, immune responses, and disease development. Existing pseudotime analysis methods primarily address the issue of reconstructing cellular pseudotemporal trajectories and inferring gene expression changes along the reconstructed trajectory in one biological sample. As scRNA-seq studies are increasingly performed on multiple patient samples, comparing gene expression dynamics across samples has been emerging as a new demand for which a systematic analytical solution is lacking.

We develop a systematic computational and statistical framework, Lamian, for multi-sample pseudotime analysis. Given scRNA-seq data from multiple biological samples with covariates (e.g., age, sex, sample type, disease status, etc.), this framework allows one to (1) construct cellular pseudotemporal trajectory, evaluate the uncertainty of the trajectory branching structure, (2) evaluate changes in branching structure associated with sample covariates, (3) identify changes in cell abundance and gene expression along the pseudotime, and (4) characterize how sample covariates modifies the pseudotemporal dynamics of cell abundance and gene expression. Importantly, when identifying cell abundance or gene expression changes associated with pseudotime and sample covariates, Lamian accounts for variability across biological samples which other existing pseudotime methods do not consider. As a result, Lamian is able to more appropriately control the false discovery rate (FDR) when analyzing multi-sample data, a property not offered by any other methods.

For more details, see our paper describing the **Lamian** package:

- A statistical framework for differential pseudotime analysis with multiple single-cell RNA-seq samples. Wenpin Hou, Zhicheng Ji, Zeyu Chen, E John Wherry, Stephanie C Hicks*, Hongkai Ji*. bioRxiv 2021.07.10.451910; doi: https://doi.org/10.1101/2021.07.10.451910

Download

Please follow the details here https://github.com/Winnie09/Lamian to download **Lamian** package and then load it using the following commands.

```
# load in Lamian
options(warn=-1)
suppressMessages(library(Lamian))
```

Module 1: tree variability

The module 1 of Lamian is designed for detecting the stability of branches in a pseudotime tree structure. We automatically enumerate all branches from the pseudotime tree structure, and then test for their detection rate through 10,000 bootstraps. After each cell-bootstrapping, we reconstruct the tree structure and re-identify all branches. We apply both Jaccard statistics and overlap coefficient as the statistics for evaluating whether

any branch in a bootstrap setting matches with one of the original branches. A branch's detection rate is defined as the percentage of bootstrap settings that a branch finds it match. Module 1 also reports and tests for samples' proportion in each branch.

Data preparation

We will need a gene by cell expression matrix, the low-dimension representation of the cells, and the cell annotation (which sample each cell belongs to). Here, we use example data **hca_bm_saver**, **hca_bm_pca**, **hca_bm_cellanno** to demonstrate their data structures. Users can read in their own data of interest in this step.

```
data(man_tree_data)
```

man_tree_data is a list containing gene by cell expression matrix, low-dimension representation (PCA) of the cells, and the cell annotation where the first column are the cell names, the second column are the sample names, and the row names are the cell names.

```
str(man_tree_data)
```

```
## List of 3
##  $ pca       : num [1:2000, 1:50] -19.6 -10.5 -13.2 -12.5 27.3 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2000] "BM2:50:male_205626" "BM4:29:male_164021" "BM3:39:male_104439" "BM3:39:male_
##   .. ..$ : chr [1:50] "PC_1" "PC_2" "PC_3" "PC_4" ...
##  $ expression: num [1:11, 1:2000] 0.0172 0.2461 1.0014 0.1017 0.0072 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:11] "CDK1" "CD14" "CDCA4" "CD7" ...
##   .. ..$ : chr [1:2000] "BM2:50:male_205626" "BM4:29:male_164021" "BM3:39:male_104439" "BM3:39:male_
##  $ cellanno  :'data.frame':  2000 obs. of  2 variables:
##   ..$ cell  : chr [1:2000] "BM2:50:male_205626" "BM4:29:male_164021" "BM3:39:male_104439" "BM3:39:ma
##   ..$ sample: chr [1:2000] "BM2" "BM4" "BM3" "BM3" ...
```

Infer tree structure

```
set.seed(12345)
res = infer_tree_structure(pca = man_tree_data[['pca']],
                           expression = man_tree_data[['expression']],
                           cellanno = man_tree_data[['cellanno']],
                           origin.marker = c('CD34'),
                           number.cluster = 5,
                           xlab='Principal component 1',
                           ylab = 'Principal component 2')
```

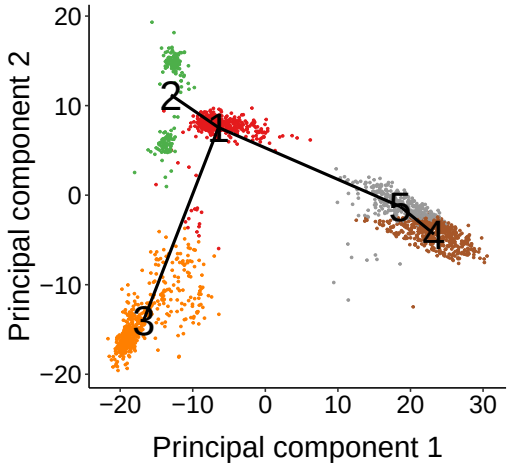
As we can see from the above inputs, there are five clusters, and the tree structure inferred based on these clusters consists of three branches: 2 –> 3 –> 1, 2 –> 4, and 2 –> 5. The result object **res** is a list containing information about the tree structure, branches, cell clusters, pseudotime ordering the cells, etc..

```
names(res)
```

```
##  [1] "pcareduceres" "MSTtree"      "clusterid"    "clucenter"    "pseudotime"
##  [6] "branch"       "js.cut"       "oc.cut"       "pca"          "order"
## [11] "allsample"
```

Plot the tree structure

```
Lamian::plotmclust(res, cell_point_size = 0.5,
                   x.lab = 'Principal component 1',
                   y.lab = 'Principal component 2')
```



Evaluate the uncertainty of tree branches

We can call the function **evaluate_uncertainty()** to evaluate the tree topology uncertainty. We suggested that users set **n.permute = 100** or more to ensure enough randomness to construct the null distribution, but here we set **n.permute = 5** in order to provide a simplified example.

```
result <- evaluate_uncertainty(res, n.permute=5)
names(result)
```

```
## [1] "detection.rate"       "sample.cellcomp.mean" "sample.cellcomp.sd"
```

Since for the simplicity of the example we set **n.permute = 5** only, the branch proportions might not really make sense. When users set **n.permute = 100** or more in their real applications, results will make much more sense. The result is a list of three elements. The first selement is the detection rate of each branch.

```
result[[1]]
```

```
##            detection.rate
## 1:2                     1
## c(1, 3)                 1
## c(1, 5, 4)              1
```

The second element is the sample proportion mean information.

```
result[[2]]
```

```
##                  BM2       BM4       BM3       BM8       BM1       BM6
## 1:2        0.2268874 0.1625525 0.2043402 0.3333333 0.2367380 0.3333333
## c(1, 3)    0.5046939 0.2329026 0.5247018 0.3333333 0.2583391 0.3333333
## c(1, 5, 4) 0.2684187 0.6045449 0.2709579 0.3333333 0.5049229 0.3333333
##                  BM5       BM7
## 1:2        0.1880292 0.3333333
## c(1, 3)    0.1880292 0.3333333
## c(1, 5, 4) 0.6239415 0.3333333
```

The thrid element is the sample proportion sd (standard deviation) information.

```
result[[3]]
```

```
##                    BM2         BM4        BM3 BM8         BM1 BM6         BM5
## 1:2        0.006148083 0.004220924 0.01119082   0 0.011753888   0 0.003269304
## c(1, 3)    0.008780549 0.006676212 0.02276736   0 0.002057573   0 0.003269304
## c(1, 5, 4) 0.002632466 0.010897136 0.01157654   0 0.009696315   0 0.006538609
##            BM7
## 1:2          0
## c(1, 3)      0
## c(1, 5, 4)   0
```

Module 2: Evaluate differential topoloty

Module 2 of Lamian first identifies variation in tree topology across samples and then assesses if there are differential topological changes associated with sample covariates. For each sample, Lamian calculates the proportion of cells in each tree branch, referred to as *branch cell proportion*. Because a zero or low proportion can reflect absence or depletion of a branch, changes in tree topology can be described using branch cell proportion changes. With multiple samples, Lamian characterizes the cross-sample variation of each branch by estimating the variance of the branch cell proportion across samples. Furthermore, regression models can be fit to test whether the branch cell proportion is associated with sample covariates. This allows one to identify tree topology changes between different conditions, for example in a case-control cohort.

```
data = result[[2]]
rownames(data) <-
  c('HSC->lymphocyte', 'HSC->myeloid', 'HSC->erythroid')
design = data.frame(sample = paste0('BM', 1, 8), sex = c(0, rep(1, 4), rep(0, 3)))
branchPropTest(data = data, design = design)
```

```
## HSC->lymphocyte    HSC->myeloid  HSC->erythroid
##       0.5129678       0.9790714       0.7383605
```

Module 3: Trajectory differential tests about gene expression

XDE test

Data preparation

In the following, we will use an example dataset **expdata** of 100 genes and 1000 cells to demonstration the workflow. In practice, we can input any other interesting dataset.

```
data(expdata)
```

The inputs should contain: (a) **expr**: a gene by cell expression matrix. Values are library-size-normalized log-transformed gene expression matrix. They can be either imputed or non-imputed. Zero-expression genes should have been filtered out.

```
str(expdata$expr)
```

```
##  num [1:120, 1:1000] 1.973 1.163 0.526 1.637 0.34 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:120] "TPM4" "HLA-DMB" "KHSRP" "ACER3" ...
##   ..$ : chr [1:1000] "BM4:26:female_219004" "BM1:26:female_219241" "BM4:29:male_179571" "BM3:26:fema
```

(b) **cellanno**: a dataframe where the first column are cell names and second column are sample names.

```
head(expdata$cellanno)
```

```
##                   Cell Sample
## 1 BM4:26:female_219004    BM4
```

```
## 2 BM1:26:female_219241    BM1
## 3   BM4:29:male_179571    BM4
## 4  BM3:26:female_52851    BM3
## 5  BM8:52:female_54061    BM8
## 6 BM8:52:female_117933    BM8
```

(c) **pseudotime**: a numeric vector of pseudotime, and the names of this vector are the corresponding cell names.

```
str(expdata$pseudotime)
```

```
##  Named int [1:1000] 1 2 3 4 5 6 7 8 9 10 ...
##  - attr(*, "names")= chr [1:1000] "BM4:26:female_219004" "BM1:26:female_219241" "BM4:29:male_179571"
```

(d) **design**: a matrix. Number of rows should be the same as the number of unique samples. Rownames are sample names. First column is the intercept (all 1), second column is the covariate realization valuels for each of the samples.

```
print(expdata$design)
```

```
##     intercept group
## BM1         1     1
## BM2         1     1
## BM3         1     0
## BM4         1     0
## BM5         1     1
## BM6         1     1
## BM7         1     0
## BM8         1     0
```

The function `lamian.test()` is designed to perform multiple tests. To perform the Sample Covariate Test, we need to set **test.type = 'variable'**.

Perform XDE test

We can perform *XDE test* using four inputs **expr**, **cellanno**, **pseudotime**, and **design**. The default permutation is 100, i.e. **permuiter = 100**, but in this small example we set it as **permuiter = 5** only to saver the running time. We recommend setting **permuiter = 100** or higher in real applications. Here, we set **testvar = 2** to test the second column of the **design** matrix as the sample covariate while adjusting for other columns (except the intercept).

```
Res <- lamian_test(expr = expdata$expr,
                   cellanno = expdata$cellanno,
                   pseudotime = expdata$pseudotime,
                   design = expdata$design,
                   test.type = 'variable',
                   testvar = 2,
                   permuiter = 5,
                   ncores = 1)
```

Downstream analysis 1: visualize and cluster XDE genes based on their multiple-sample temporal patterns across sample covariates

We will know which are XDE from the **statistics** data frame in the result object **Res**.

```
## get differential dynamic genes statistics
stat <- Res$statistics
head(stat)
```

```
##          fdr.overall pval.overall z.overall log.pval.overall  fdr.meanDiff
## TPM4    1.483541e-14 1.149744e-14  6.063011        -30.49998  2.195804e-73
## HLA-DMB 0.000000e+00 0.000000e+00 11.711591     -38185.21499  1.872958e-06
## KHSRP   0.000000e+00 0.000000e+00 30.739535      -1028.81158 1.227456e-247
## ACER3   2.930225e-31 2.075576e-31  8.340832        -69.24017  2.836927e-07
## TMEM14A 2.505465e-76 1.294490e-76 10.176168       -173.12891  2.330766e-68
## FAM168B 1.245583e-45 7.888694e-46 10.434159       -102.24405  7.822035e-01
##         pval.meanDiff z.meanDiff log.pval.meanDiff fdr.trendDiff pval.trendDiff
## TPM4     8.206542e-74  5.6868469     -1.666769e+02 3.771912e-197  4.191014e-198
## HLA-DMB  1.040532e-06  3.6893786     -1.227760e+01  1.562986e-16   6.630850e-17
## KHSRP   2.851667e-248 21.2651175     -5.683838e+02  5.449635e-13   2.587200e-13
## ACER3    1.461447e-07  4.0331280     -1.413239e+01  8.793795e-04   4.885442e-04
## TMEM14A  8.946375e-69 12.3965644     -1.551325e+02  2.784120e-01   1.771713e-01
## FAM168B  7.031931e-01 -0.7264751     -8.947868e-04  0.000000e+00   0.000000e+00
##         z.trendDiff log.pval.trendDiff
## TPM4      11.345342       -452.8694679
## HLA-DMB    3.297219        -35.6427756
## KHSRP      3.182796        -27.3735920
## ACER3      2.868519         -6.0360748
## TMEM14A    1.053268         -0.1301529
## FAM168B   18.365121       -739.9322942
```

```
stat <- stat[order(stat[, 1],-stat[, 3]),]
## identify XDE genes with FDR.overall < 0.05 cutoff
diffgene <-
  rownames(stat[stat[, grep('^fdr.*overall$', colnames(stat))] < 0.05, ])
str(diffgene)
```

```
##  chr [1:99] "ACOT8" "KHSRP" "YTHDC2" "ZBTB2" "CNIH4" "GCFC2" "SPATC1L" ...
```

We will get the population-level estimates for all the XDE genes by applying function `getPopulationFit()`.

```
## population fit
Res$populationFit <-
  getPopulationFit(Res, gene = diffgene, type = 'variable')
```

We can apply `getCovariateGroupDiff()` to calculate the group difference regarding this sample covariate, and then cluster the XDE genes based on the group difference. By setting $\mathbf{k = 4}$, we will get two clusters for meanSig XDE genes, and the other two clusters for XDE genes of other types.

```
## clustering
Res$covariateGroupDiff <-
  getCovariateGroupDiff(testobj = Res, gene = diffgene)
Res$cluster <-
  clusterGene(Res, gene = diffgene, type = 'variable', k = 5)
table(Res$cluster)
```

```
##
##  1  2  3  4  5  6  7  8  9
##  4  6 12  6 11  5 15 23 17
```

We can also plot original expression values, model fitted expression values, and model-fitted group difference in three seperate heatmaps.

```
colnames(Res$populationFit[[1]]) <-
  colnames(Res$populationFit[[2]]) <- colnames(Res$expr)
plotXDEHm(
  Res,
```

```
  cellWidthTotal = 180,
  cellHeightTotal = 350,
  subsampleCell = FALSE,
  sep = ':.*'
)
```

We can plot the cluster mean and difference.

```
## plotClusterMeanAndDiff
plotClusterMeanAndDiff(Res)
```

We can also plot the cluster difference seperately by calling `plotClusterDiff(testobj=Res, gene = diffgene)`.

Downstream analysis 2: Gene ontology (GO) enrichment analysis

We can further check out whether the XDE genes in each cluster have any enriched genome functions by applying Gene Ontology (GO) analysis `library(topGO); goRes <- GOEnrich(testobj = Res, type = 'variable')`. Please note that there are no enriched GO terms in any of the clusters in this simplified example since we only subset 1,000 genes. In practice, if there are significant GO terms in the clusters, we can generate a heatmap of the top GO terms using the command `plotGOEnrich(goRes = goRes)`.

TDE test

Data preparation

The inputs for Constant Time Test are the same as those for Sample Covariate Test (see above section), except that the **design** matrix can have only one intercept column. If there are more than one columns in **design**, only the first column will be considered.

Perform TDE test

Similar to the XDE test, we can perform *TDE test* using four inputs **expr**, **cellanno**, **pseudotime**, and **design**. The only difference is that **design** can be a one-column matrix or dataframe whose values are 1s. If there are multiple columns, only the first column will be actually used. The default permutation is 100, i.e. **permuiter = 100**, but in this small example we set it as **permuiter = 5** again to saver the running time. We recommend setting **permuiter = 100** or higher in real applications.

```
Res <- lamian_test(
  expr = expdata$expr,
  cellanno = expdata$cellanno,
  pseudotime = expdata$pseudotime,
  design = expdata$design,
  test.type = 'time',
  permuiter = 5
)
names(Res)
```

```
##  [1] "statistics"  "parameter"   "llr.overall" "knotnum"     "pseudotime"
##  [6] "design"      "cellanno"    "expr"        "test.type"   "test.method"
```

Downstream analysis 1: visualize and cluster TDE genes based on their multiple-sample temporal patterns

The result object is a list containing multiple elements. The first element is a dataframe of statistics.

```
head(Res$statistics)
```

```
##         fdr.overall pval.overall z.overall log.pval.overall
## TPM4              0            0 131.13446       -68090.279
## HLA-DMB           0            0  34.44377       -27734.379
## KHSRP             0            0  37.44955        -1577.405
## ACER3             0            0  38.20505       -12077.123
## TMEM14A           0            0  46.95296        -2466.988
## FAM168B           0            0  86.30528        -9151.278
```

We can further determine the TDE genes as the genes with **fdr.overall** < 0.05.

```
diffgene <- rownames(Res$statistics)[Res$statistics[, 1] < 0.05]
str(diffgene)
```

```
##  chr [1:93] "TPM4" "HLA-DMB" "KHSRP" "ACER3" "TMEM14A" "FAM168B" "SETD9" ...
```

We can estimate the population-level model-fitted patterns for all TDE genes.

```
Res$populationFit <-
  getPopulationFit(Res, gene = diffgene, type = 'time')
```

Then we can further cluster these genes.

```
Res$cluster <-
  clusterGene(Res, gene = diffgene, type = 'time', k = 3)
```

We can visualize the temporal patterns and compare original and fitted expression

```
plotTDEHm(
  Res,
  subsampleCell  = FALSE,
  showCluster = TRUE,
  type = 'time',
  cellWidthTotal = 200,
  cellHeightTotal = 200
)
```

To plot cluster mean patterns.

```
plotClusterMean(testobj = Res,
                cluster = Res$cluster,
                type = 'time')
```

Downstream analysis 2: GO analysis for TDE genes

Similar to **XDE test**, we can further proceed to identify the enriched GO terms for each gene cluster using commands `library(topGO); goRes <- GOEnrich(testobj = Res, type = 'time', sep = ':.*')`. Please note that there are no enriched GO terms in any of the clusters in this simplified example since we only subset 1,000 genes. In practice, if there are enriched GO terms, we can plot them using commands `plotGOEnrich(goRes = goRes)`.

Module 4: trajectory differential test based on cell composition

TCD test

Data preparation

The inputs are the same as those for TDE test (see above), except that we don not need gene expression matrix.

Perform cell proportion test

```
Res <-
  cellPropTest(
    cellanno = expdata$cellanno,
    pseudotime = expdata$pseudotime,
    design = expdata$design[, 1, drop = F],
    ncores = 4,
    test.type = 'Time'
  )
```

The dataframe **statistics** in the **Res** object contains the test result: a p-value **pval.overall** and a effect size z-score **z.overall**.

```
head(Res$statistics)
```

```
##         fdr.overall pval.overall z.overall log.pval.overall
## TPM4              0            0 131.13446       -68090.279
## HLA-DMB           0            0  34.44377       -27734.379
## KHSRP             0            0  37.44955        -1577.405
## ACER3             0            0  38.20505       -12077.123
## TMEM14A           0            0  46.95296        -2466.988
## FAM168B           0            0  86.30528        -9151.278
```

XCD test

The inputs are the same for XDE test (see above), except that we do not need gene expression matrix.

```
Res <-
  cellPropTest(
    cellanno = expdata$cellanno,
    pseudotime = expdata$pseudotime,
    design = expdata$design,
    ncores = 4,
    test.type = 'Variable',
    testvar = 2
  )
```

The dataframe **statistics** in the **Res** object contains the test result: a p-value **pval.overall** and a effect size z-score **z.overall**.

```
head(Res$statistics)
```

```
##         fdr.overall pval.overall z.overall log.pval.overall
## TPM4              0            0 131.13446       -68090.279
## HLA-DMB           0            0  34.44377       -27734.379
## KHSRP             0            0  37.44955        -1577.405
## ACER3             0            0  38.20505       -12077.123
## TMEM14A           0            0  46.95296        -2466.988
```

```
## FAM168B           0            0  86.30528       -9151.278
```

Session Info

```
sessionInfo()
```

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS  10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] Lamian_0.99.0
##
## loaded via a namespace (and not attached):
##   [1] colorspace_2.0-1     rjson_0.2.20         ellipsis_0.3.2
##   [4] mclust_5.4.6         rprojroot_2.0.2      circlize_0.4.13
##   [7] GlobalOptions_0.1.2  fs_1.4.2             clue_0.3-59
##  [10] farver_2.1.0         remotes_2.1.1        topGO_2.42.0
##  [13] bit64_0.9-7          AnnotationDbi_1.52.0 fansi_0.4.2
##  [16] splines_4.0.2        knitr_1.33           pkgload_1.1.0
##  [19] Cairo_1.5-12.2       cluster_2.1.0        GO.db_3.12.1
##  [22] png_0.1-7            pheatmap_1.0.12      graph_1.67.1
##  [25] shiny_1.5.0          compiler_4.0.2       assertthat_0.2.1
##  [28] Matrix_1.2-18        fastmap_1.0.1        cli_2.5.0
##  [31] later_1.1.0.1        htmltools_0.5.1.1    prettyunits_1.1.1
##  [34] tools_4.0.2          igraph_1.2.5         gtable_0.3.0
##  [37] glue_1.4.2           reshape2_1.4.4       dplyr_1.0.2
##  [40] Rcpp_1.0.5           scattermore_0.7      Biobase_2.49.0
##  [43] TSCAN_1.7.0          vctrs_0.3.8          rhdf5filters_1.2.1
##  [46] gdata_2.18.0         nlme_3.1-148         xfun_0.22
##  [49] stringr_1.4.0        ps_1.4.0             testthat_3.0.0
##  [52] mime_0.9             lifecycle_1.0.0      gtools_3.8.2
##  [55] devtools_2.3.0       org.Hs.eg.db_3.12.0  scales_1.1.1
##  [58] promises_1.1.1       SparseM_1.78         rhdf5_2.34.0
##  [61] RColorBrewer_1.1-2   ComplexHeatmap_2.6.2 yaml_2.2.1
##  [64] memoise_1.1.0        gridExtra_2.3        ggplot2_3.3.3
##  [67] fastICA_1.2-2        stringi_1.5.3        RSQLite_2.2.0
##  [70] S4Vectors_0.27.12    desc_1.2.0           caTools_1.18.0
##  [73] BiocGenerics_0.35.4  pkgbuild_1.1.0       shape_1.4.6
##  [76] rlang_0.4.11         pkgconfig_2.0.3      matrixStats_0.57.0
##  [79] bitops_1.0-6         evaluate_0.14        lattice_0.20-41
##  [82] purrr_0.3.4          Rhdf5lib_1.12.1      labeling_0.4.2
```

```
##  [85] bit_4.0.4           processx_3.4.4       tidyselect_1.1.0
##  [88] plyr_1.8.6          magrittr_2.0.1       R6_2.5.0
##  [91] IRanges_2.23.10     gplots_3.0.4         generics_0.1.0
##  [94] combinat_0.0-8      DBI_1.1.0            pillar_1.6.1
##  [97] withr_2.4.2         mgcv_1.8-33          tibble_3.1.2
## [100] crayon_1.4.1        KernSmooth_2.23-17   utf8_1.2.1
## [103] rmarkdown_2.7       viridis_0.5.1        GetoptLong_1.0.5
## [106] usethis_1.6.1       grid_4.0.2           blob_1.2.1
## [109] callr_3.5.1         matrixcalc_1.0-3     digest_0.6.27
## [112] xtable_1.8-4        httpuv_1.5.4         stats4_4.0.2
## [115] munsell_0.5.0       viridisLite_0.4.0    sessioninfo_1.1.1
```

Citation

If the **Lamian** package is useful in your work, please cite the following paper:

- A statistical framework for differential pseudotime analysis with multiple single-cell RNA-seq samples. Wenpin Hou, Zhicheng Ji, Zeyu Chen, E John Wherry, Stephanie C Hicks*, Hongkai Ji*. bioRxiv 2021.07.10.451910; doi: https://doi.org/10.1101/2021.07.10.451910

Maintaince or issue reports

Should you encounter any bugs or have any suggestions, please feel free to contact Wenpin Hou whou10@jhu .edu, or open an issue on the Github page https://github.com/Winnie09/Lamian/issues.