

Illustrations on Reproducing Results

DESCRIPTION

The source code comprises of 3 files; cpputils.cpp, core.R and plotting.R. cpputils.cpp contains some utility functions while core.R and plotting.R is comprised of the main functions needed to reproduce the results.

COMPILING AND INSTALLING

The source files and data files are zipped together. To compile: 1. Rcpp::sourceCpp('cpputils.cpp') 2. source('core.R') 3. source('plotting.R')

Unzip the the file containing the source code and data into your working directory. After unzipping, there is now 'BMC_source_and_data' directory inside your working directory.

In this example, my working directory is set to "~/Documents".

Run the code below to compile all the 3 source files.

```
if(!require("dyndimred", quietly = TRUE))
{
  install.packages("dyndimred", repos = "http://cran.us.r-project.org")
  library(dyndimred)
}
Rcpp::sourceCpp('./BMC_source_and_data/DetectTrajectory_src/cpputils.cpp')
source('./BMC_source_and_data/DetectTrajectory_src/core.R')
source('./BMC_source_and_data/DetectTrajectory_src/plotting.R')
```

RUNNING AND EXAMPLES

The function 'reproduce.results' reproduces the results, including plots, presented in the paper. The function takes 2 inputs, a path to all the data used in the paper(provided in .rds format), and a path for all the output plots. The function returns a list with pvalues for each statistic and for every dataset used in the paper. The list also contains corresponding number of clusters, at which each pvalue was obtained. All the data used in the manuscript is provided in the 'data' directory, provided together with the source code. All the plots generated from running the function are in the 'results' directory.

Run the code below to reproduce the results

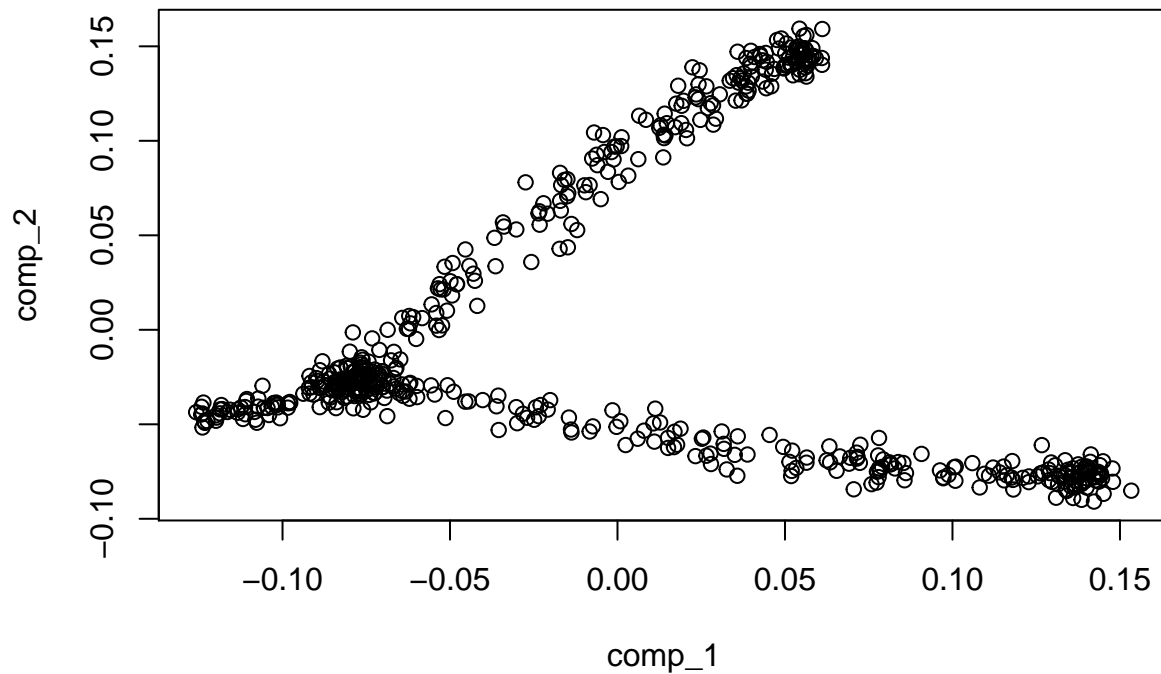
```
res = reproduce.results("./BMC_source_and_data/data", "./BMC_source_and_data/results")
res

## $median_pvalues
## $median_pvalues$noisy_simulated
##      d1      d2      lp
## 0.8146667 0.6733333 0.6640000
##
## $median_pvalues$bifurcating_simulated
##      d1      d2      lp
## 0.01266667 0.00800000 0.02933333
##
## $median_pvalues$`cell-cycle_buettner`
##      d1      d2      lp
## 0.5553333 0.3373333 0.1600000
##
## $median_pvalues$`mesoderm-development_loh`
##      d1      d2      lp
## 0.16333333 0.16666667 0.02666667
```

```
##
## $median_pvalues$multifurcating_7
##      d1      d2      lp
## 0.048 0.050 0.146
##
## $median_pvalues$disconnected_10
##      d1      d2      lp
## 0.002 0.002 0.002
##
## $median_pvalues$diverging_converging_6
##          d1          d2          lp
## 0.02933333 0.02933333 0.06000000
##
## $median_pvalues$noisy_simulated2
##          d1          d2          lp
## 0.8980000 0.7686667 0.8840000
##
##
## $median_clusters
## $median_clusters$noisy_simulated
## d1 d2 lp
## 16 8 17
##
## $median_clusters$bifurcating_simulated
## d1 d2 lp
## 29 24 15
##
## $median_clusters$`cell-cycle_buettner`
## d1 d2 lp
## 12 9 19
##
## $median_clusters$`mesoderm-development_loh`
## d1 d2 lp
## 20 24 11
```

Main function is `detect.traj`. It takes as input a matrix `x`, with rows as observation and columns as features; minimum number of clusters `min.k`; maximum number of clusters `max.k`; and number of permutations, `perm`. Output is an object of class 'pvals', which is a list of pvalues for each of the 3 statistics over all cluster numbers. `Plot_pvals` takes an object of class 'pvals' and plots box plots of pvalues for all the 3 statistics.

```
library(dyndimred)
data1 = readRDS("./BMC_source_and_data/data/bifurcating_simulated.rds")
x = dyndimred::dimred_mds(data1$expression, ndim=2)
plot(x)
```

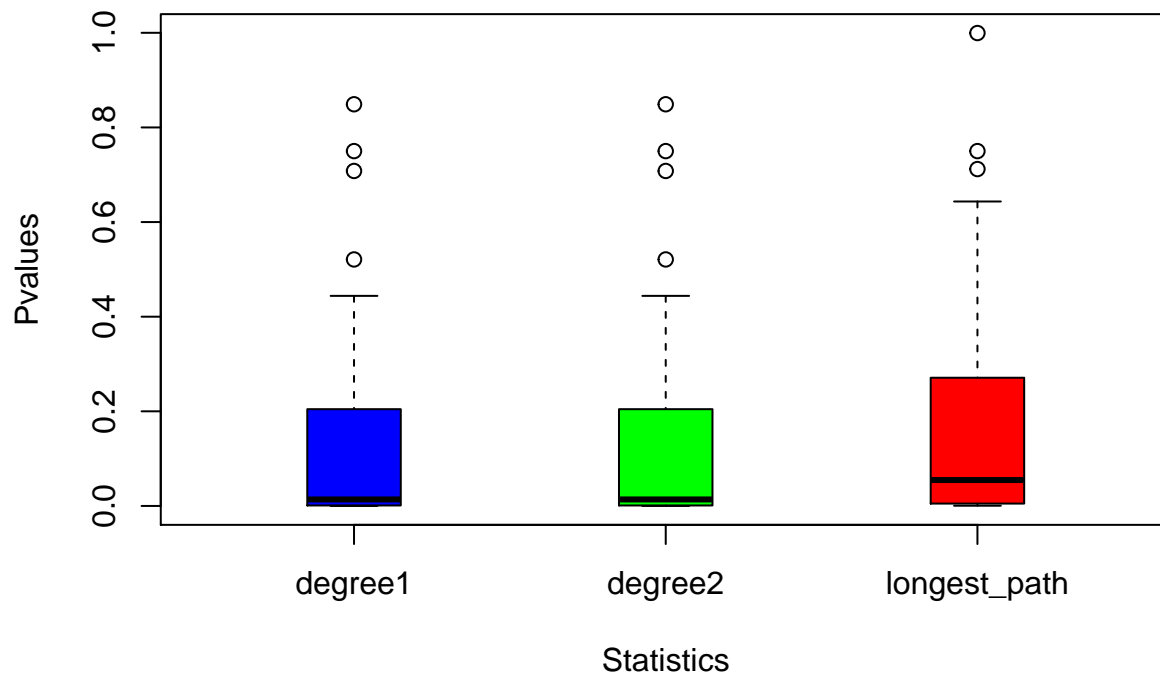


```
res <-detect.traj(x, min.k=5, max.k=30, perm=2000)
```

```
res
```

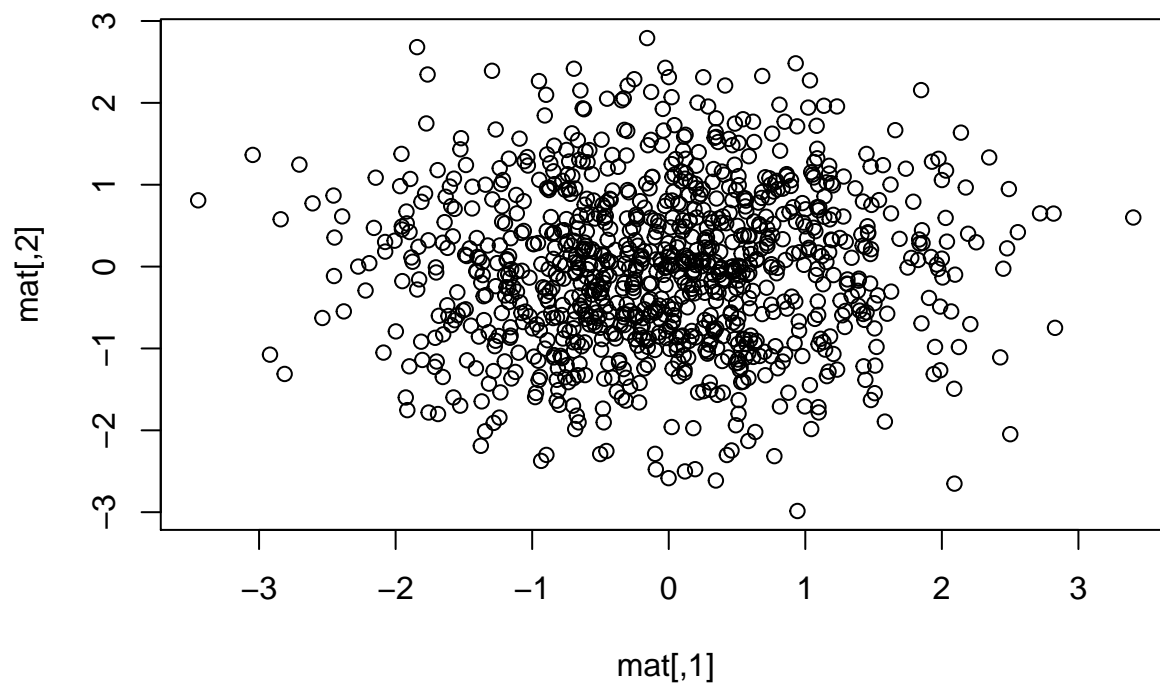
```
## $num_clustering
## [1] 26
##
## $d1_pvalues
## [1] 0.7500 0.5210 0.8490 0.7080 0.1890 0.4440 0.3370 0.2045 0.1160 0.0660
## [11] 0.0430 0.0275 0.0170 0.0105 0.0050 0.0015 0.0010 0.0005 0.0010 0.0085
## [21] 0.0005 0.0000 0.0000 0.0015 0.0105 0.0005
##
## $d2_pvalues
## [1] 0.7500 0.5210 0.8490 0.7080 0.1890 0.4440 0.3370 0.2045 0.1160 0.0660
## [11] 0.0430 0.0275 0.0170 0.0105 0.0050 0.0015 0.0010 0.0005 0.0010 0.0085
## [21] 0.0005 0.0000 0.0000 0.0015 0.0000 0.0005
##
## $lp_pvalues
## [1] 0.7500 0.5210 0.9995 0.6435 0.1890 0.7120 0.2710 0.4295 0.0850 0.0475
## [11] 0.0270 0.0805 0.1475 0.2630 0.0620 0.0050 0.0180 0.0460 0.0010 0.0030
## [21] 0.0005 0.0040 0.0015 0.0080 0.0065 0.0005
##
## $d1_pvalue
## [1] 0.01375
##
## $d2_pvalue
## [1] 0.01375
##
## $lp_pvalue
## [1] 0.05475
##
## attr("class")
## [1] "pvals"
```

```
plot_pvals(res)
```



Applying the function to random data with no significant trajectory.

```
mat = cbind(rnorm(1000), rnorm(1000))
plot(mat)
```

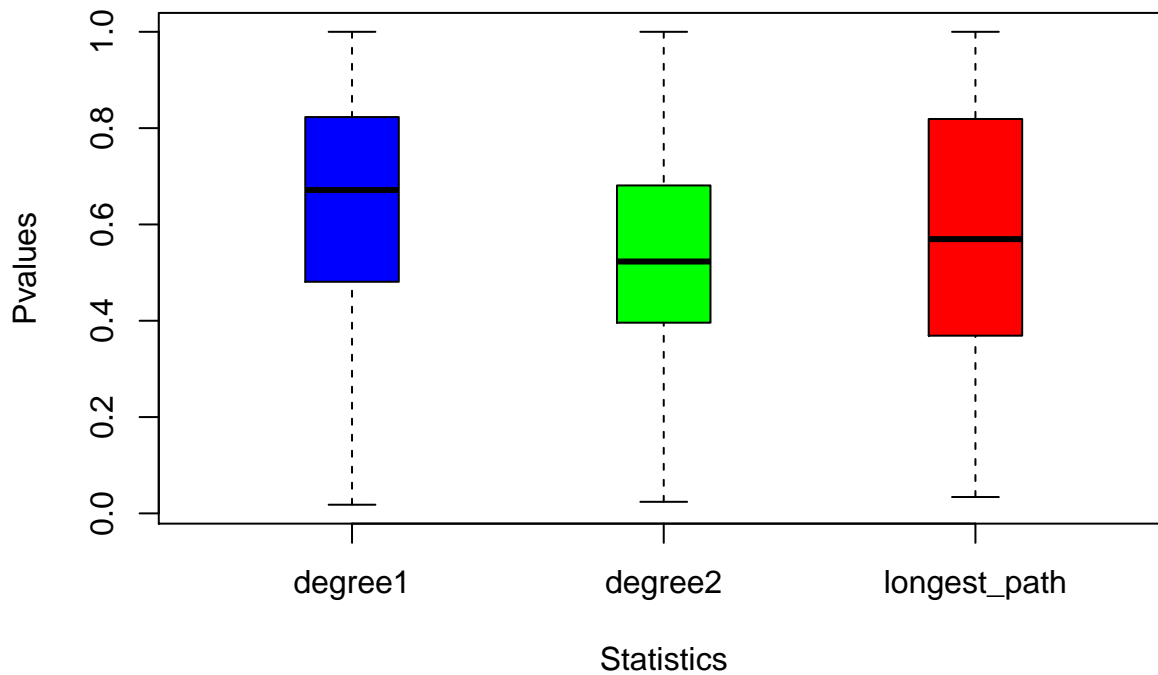


```
res <- detect.traj(mat, 5, 30, 1000)
res
```

```
## $num_clustering
## [1] 26
```

```
##
## $d1_pvalues
## [1] 0.500 1.000 0.638 0.234 1.000 0.270 0.018 0.481 0.270 0.723 0.529 0.891
## [13] 0.767 0.963 0.439 0.823 0.741 0.621 0.870 0.973 0.715 0.637 0.477 0.764
## [25] 0.705 0.627
##
## $d2_pvalues
## [1] 0.500 1.000 0.654 0.245 0.613 0.460 0.024 0.408 0.212 0.313 0.466 0.849
## [13] 0.386 0.945 0.380 0.772 0.396 0.760 0.647 0.970 0.428 0.598 0.467 0.546
## [25] 0.681 0.580
##
## $lp_pvalues
## [1] 0.500 1.000 0.638 0.212 1.000 0.662 0.034 0.607 0.232 0.369 0.480 0.972
## [13] 0.756 0.432 0.928 0.723 0.181 0.532 0.999 0.747 0.819 0.896 0.114 0.423
## [25] 0.514 0.279
##
## $d1_pvalue
## [1] 0.6715
##
## $d2_pvalue
## [1] 0.523
##
## $lp_pvalue
## [1] 0.5695
##
## attr("class")
## [1] "pvals"
```

```
plot_pvals(res)
```



Function `compute.distribution` computes the null distributions of the 3 statistics. It takes as input a matrix `x`, number of clusters `k`, and number of permutations `perm`. `Plot_distr` plots the distributions of the statistics

```
distbns = compute.distributions(x, k=15, perm=1000)
plot_distr(distbns)
```

