dynGENIE3: documentation

Author: Vân Anh Huynh-Thu, vahuynh@uliege.be

This is the documentation for the R implementation of dynGENIE3. This implementation is a research prototype and is provided "as is". No warranties or guarantees of any kind are given.

The dynGENIE3 method is described in the following paper:
Huynh-Thu V. A. and Geurts P. (2017) dynGENIE3: dynamical GENIE3 for the inference of gene networks from time series expression data. *Scientific Reports* 8:3384.

This R implementation wraps a C code. The source code is provided in the two files "dynGENIE3.R" and "dynGENIE3.c".

1 Installation

1. You will have to install R:
   http://www.r-project.org

2. Compile the C code by typing the following command (you must be in the directory where "dynGENIE3.c" is located):

   ```
   R CMD SHLIB dynGENIE3.c
   ```

   This will create two files dynGENIE3.o and dynGENIE3.so

3. You must also install the following R packages, all available from CRAN:

   - *reshape2*
     https://cran.r-project.org/web/packages/reshape2/index.html
   - *doRNG*
     https://cran.r-project.org/web/packages/doRNG/index.html
   - *doParallel*
     https://cran.r-project.org/web/packages/doParallel/index.html

   This installation can be done from R with these commands:

   ```
   install.packages("reshape2")
   install.packages("doRNG")
   install.packages("doParallel")
   ```

2 Load the dynGENIE3 package

```
source("dynGENIE3.R")
```

This command will load the source file.

3 Run dynGENIE3

dynGENIE3 is meant to be run on time series data (with or without steady-state data).

Load the data

Some files (in directory 'example_data') containing gene expression data are provided for this tutorial. There are 3 time series and one steady-state expression dataset. The function `read.expr.matrix()` can be used to load this data:

```
# Load time series data
TS1 <- read.expr.matrix("example_data/time_series_1.txt", ...
  form="rows.are.samples")
TS2 <- read.expr.matrix("example_data/time_series_2.txt", ...
  form="rows.are.samples")
TS3 <- read.expr.matrix("example_data/time_series_3.txt", ...
  form="rows.are.samples")

# Re-format data
time.points <- list(TS1[1,], TS2[1,], TS3[1,])
TS.data <- list(TS1[2:nrow(TS1),], TS2[2:nrow(TS2),], TS3[2:nrow(TS3),])
```

The function `read.expr.matrix` reads the expression matrix from the file. This command will automatically recognize if gene names and/or sample names are provided in the first row and/or column. The `form` parameter is important to set correctly. It tells if every row in the expression file corresponds to a gene (‘‘`rows.are.genes`’’), or if every row corresponds to a sample (‘‘`rows.are.samples`’’). Additional function parameters and information are explained in the source file.

Run dynGENIE3 with its default parameters

The following input arguments are mandatory to run the function `dynGENIE3()`:

- `TS.data`: a list of matrices containing time series expression data. Each matrix (genes x time points) corresponds to a time series experiment. Each row of a matrix is a gene, each column is a time point.
- `time.points`: a list of vectors containing the time points. The k-th vector must correspond to the k-th time series of `TS.data`.

```
# Run dynGENIE3
res <- dynGENIE3(TS.data,time.points)
```

The function `dynGENIE3()` returns a list object (here `res`) containing the following items:

- `res$weight.matrix`: the weighted adjacency matrix of the gene network. In this weight matrix, the element (i, j) (row i, column j) gives the weight of the link from regulatory gene i to target gene j, with high scores corresponding to more likely regulatory links.
- `res$alphas`: the gene decay rates. By default, the decay rate α_i of gene i is estimated from its time series data, by assuming an exponential decay $e^{-\alpha_i t}$ between the highest and lowest observed expression values of gene i.

Set the values of the decay rates

The gene decay rates can be specified with the input argument `alpha`. `alpha` can be either a single positive number (in that case, the decay rates of all the genes are assumed to have the same value) or a vector of positive numbers (specifying the value of the decay rate of each gene).

```
# Gene degradation rates
decay_rates <- 0.02
# Or alternatively:
gene.names <- rownames(TS.data[[1]])
decay_rates <- rep(0.02, length(gene.names))
decay_rates <- setNames(decay_rates, gene.names)

# Run dynGENIE3
res2 <- dynGENIE3(TS.data,time.points,alpha=decay_rates)
```

In this case, the returned vector `res2$alphas` is the same as `decay_rates`.

Run dynGENIE3 on time series data and steady-state data

`dynGENIE3()` can be used to learn networks jointly from time series and steady-state data. The steady-state data must be contained in a matrix where the element (i, n) is the expression of the i-th gene in the n-th steady-state condition.

```
# Load some steady-state data
SS.data <- read.expr.matrix("example_data/SS_data.txt", ...
  form="rows.are.samples")

res3 = dynGENIE3(TS.data,time.points,SS.data=SS.data)
```

Restrict the candidate regulators to a subset of genes

You can specify that only a subset of the genes are to be used as candidate regulators, by passing an array of gene indices through the `regulators` parameter:

```
res4 <- dynGENIE3(TS.data,time.points,regulators=3:5)
```

Here, for example, only genes from 3 to 5 (included) will be used as candidate regulators. This can be useful when you know which genes are transcription

factors. If you have a list of gene names to be used as candidate regulators, you can also use it like this:

```
input.genes <- c("GATA5","XRCC2","OSR2","RAD51")
res5 <- dynGENIE3(TS.data,time.points,regulators=input.genes)
```

Change the tree-based method and its settings

```
# Use the Extra-Trees as tree-based method
tree.method <- "ET"

# Number of randomly chosen candidate regulators at each node of a tree
K <- 7

# Number of trees per ensemble
ntrees <- 50

# Run the method with these settings
res6 <- dynGENIE3(TS.data,time.points, ...
  tree.method=tree.method, K=K, ntrees=ntrees)
```

Obtain more information

Additional function parameters and information are explained in the source file.

4 Write the predictions

Get the predicted ranking of all the regulatory links

```
link.list <- get.link.list(res$weight.matrix)
head(link.list)
```

The output will look like this:

```
##  regulatory.gene target.gene    weight
## 1           XRCC2        TBX3 0.6066478
## 2            TBX3       XRCC2 0.5961631
## 3           CREB5        CD93 0.4408768
## 4            CD19       RAD51 0.4031206
## 5            CD93       CREB5 0.3886690
## 6           GATA5       CREB5 0.2937685
```

Each line corresponds to a regulatory link. The first column shows the regulator, the second column shows the target gene, and the last column indicates the score of the link.
Note that the ranking that is obtained will be slightly different from one run to another. This is due to the intrinsic randomness of the Random Forest and Extra-Trees method. The variance of the ranking can be decreased by increasing the number of trees per ensemble.

Get the first 5 links only

```
link.list <- get.link.list(res$weight.matrix, report.max=5)
```

Get the links with a score above 0.1

```
link.list <- get.link.list(res$weight.matrix, threshold=0.1)
```

Important note on the interpretation of the scores: The weights of the links returned by `dynGENIE3()` **do not have any statistical meaning** and only provide a way to rank the regulatory links. There is therefore no standard threshold value, and caution must be taken when choosing one.

Obtain more information

Information about function parameters are explained in the source file.