

User's Guide for Dynomics

Package: **Dynomics**

Version: 0.1

Feb. 2009

Zahra Montazeri, David R. Bickel, Ottawa Institute of System Biology, Department of BMI, University of Ottawa

License: Mozilla Public License 1.1. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>.

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

Contents:

1. Introduction
2. Before running Dynomics
3. Usage
4. Arguments
5. Output
6. Requirements
7. Example
8. Reference

1. Introduction

Dynomics is a tool to select a regulating gene among the set of potentially regulating genes for any gene of interest. This program analyzes causal relationship between genes by applying Bayes's theorem to well known kinetic models. The main function is **downstream** which inputs for this function are gene expression over time for regulating and regulated gene. Selected regulating gene according to the first order, second order, and average models (see D.R.Bickel, *et al.*, 2009 for details) along with posterior probabilities are returned by **Dynomics**.

2. Before running Dynomics

In order to produce output, R and the Biobase package of Bioconductor must be installed. Place all files that end with '.r' or '.s' in the same directory. Launch R, then set the working directory to the directory with those '.r' and '.s' files. Files "**data.r**" and "**data.s**" are used to transfer data to **XprnSet** format (see example). Lines beginning with the sign **#** are annotation lines to explain parameters.

One needs to apply the following actions to analyze data by **Dynomics** (if applicable):

- **Changing format:** Function **XprnSet** change the data from matrix to **XprnSet**.
- **Missing data:** If some part of data are missing, they should first be filled (imputed) by using function **imputation** and then analyzed by function **downstream**.
- **Negative data:** When the gene expression or ratio is given, all data are positive and one may use the function **downstream** directly since **downstream** acts on objects of class **XprnSet**. On the other hand, the given data could be negative if they have been changed by log, in which case **xprnSet** represents them (**xprnSet** uses for negative data). The function **positive.downstream** changes an object's class from **xprnSet** to **XprnSet**.

3. Usage

```
downstream(data.process,downs,time.name)
```

4. Arguments

<code>data.process</code>	Measured gene expression intensities of possible regulator genes in XprnSet format
<code>downs</code>	Measured gene expression intensities of interest genes in XprnSet format
<code>time.name</code>	Name of the column represented time in pdata (see below), default is “time”

5. Output

`downstream` returns a data frame of the following columns:

<code>Marginal prob-1</code>	Marginal posterior probability for the first order model over all genes
<code>Marginal prob-2</code>	Marginal Posterior probability for the second order model over all genes
<code>Regulated-Name-1th order</code>	Name of regulator gene selected by the first order model
<code>prob-1th order</code>	Probability that the highest-probability gene is regulated by the interest gene according to the first order model
<code>Regulated-Name-2th order</code>	Name of regulator gene selected by the second order model
<code>prob-2th order</code>	Probability that the highest probability gene is regulated by the interest gene according to the second order model
<code>Regulated-Name-union</code>	Name of regulator gene selected by the average model
<code>prob-union</code>	Probability the highest probability gene is regulated by the interest gene according to the average model

6. Requirements

“Dynomics” is written in Biobase and the R programming language. It can be run on Linux, Mac OS X or Microsoft Windows operating systems with R and Biobase installed. R and Biobase are freely available at <http://cran.us.r-project.org/> and <http://www.bioconductor.org/>, respectively. Furthermore, the following functions are included

- **imputation** This function replaces missing values by the average of the values of their immediate neighbours, but first filters out all profiles that have missing values for whole time-course length.

```
imputation(data.process)
```

– Arguments:

<code>data.process</code>	Measured gene expression intensities of possible regulator genes in XprnSet format with missing data
---------------------------	---

– output:

Imputed expression intensities for regulator genes as an object of class **XprnSet**

- **positive.downstream** This function changes **xprnSet** to **XprnSet**.

```
positive.downstream(data.process,base)
```

– Arguments:

<code>data.process</code>	Logarithm of measured gene expression intensities in xprnSet format
<code>base</code>	Base of logarithm taken to generate <code>data.process</code>

- output:
Measured gene expression intensities in `XprnSet` format
- **XprnSet** This function changes the format of gene expression intensities to `XprnSet`
`XprnSet(pdata,data)`
 - Arguments:
 - `pdata` phenotypic data in `data.frame` format
 - `data` data (gene expression) in `Matrix` format (capitalization indicates nonnegative)
 - output:
Measured gene expression intensities in `XprnSet` format
- **xprnSet** This function changes the format of logarithm of measured gene expression intensities to `xprnSet` format
`xprnSet(pdata,data)`
 - Arguments:
 - `pdata` phenotypic data in `data.frame` format
 - `data` data (logarithm of gene expression) in `matrix` format
 - output:
Measured logarithm of gene expression in `xprnSet` format
- **Details**
One of the argument which is used to make `XprnSet` or `xprnSet` is `pdata`. Time points in `pdata` need to be in *character*, *factor*, or *numeric* format and should be in order with equally spaced differences. Furthermore, the whole time course for each replicate should be together, e.g., for time (1,2,3) with 2 replicates set `pdata = data.frame(c(1,2,3,1,2,3))`

7. Example

```
setwd("C:/Documents and Settings/My Documents/directory.name")
#path of the directory that contains the source files

source("Dynamics.r")
t<-seq(1,10,by=1)
x<-matrix(0,nrow=10000,ncol=10)
for(i in 1:10000){
x[i,]<-rnorm(10,mean=runif(1,min=0.2,max=1), sd=0.4)+sin(t)
}
colnames(x)<-paste("t",1:10, sep="")
rownames(x)<-paste("X",1:10000, sep="")
pdata.x<-seq(1,10,by=1)
pdata.x<-as.data.frame(pdata.x)
rownames(pdata.x)<-colnames(x)
colnames(pdata.x)<-"time"
regulate.x<-x[c(1,4,6,8,12),]
ex.x<-xprnSet(pdata.x,x)
ex.regulate.x<-xprnSet(pdata.x,regulate.x)
ex.x<-positive.downstream(ex.x,2)
ex.regulate.x<-positive.downstream(ex.regulate.x,2)
result<-downstream(ex.x,ex.regulate.x)
```

In this example, we generated a set of microarray data with 10000 genes with 10 time points in `xprnSet` format, then change it to `XprnSet` by using `positive.downstream`. Here, `ex.x` is a set of potentially regulating gene expressions and `ex.regulate.x` is a set of interest genes. After running `downstream`, we had the following results in 15 minutes

```
> result
      Marginal prob-1      Marginal prob-2 Regulated-Name-1th order      prob-1th order
1 0.96947698933206 0.0305230106679409      X101 0.00673076147350804
2 0.922828022945725 0.0771719770542747      X6157 0.0493083536335618
3 0.926531724827964 0.0734682751720356      X4601 0.00915601940537825
4 0.978197690501962 0.0218023094980377      X4292 0.0139415376398571
5 0.982631762227543 0.0173682377724572      X4606 0.0334673559500337
      Regulated-Name-2th order      prob-2th order Regulated-Name-union      prob-union
1      X1961 0.00765317803007236      X101 0.00425749508744453
2      X8062 0.0186823327936379      X6157 0.0269145620349890
3      X1566 0.0640998718361330      X4601 0.00541063187505122
4      X8485 0.0195618653976766      X4292 0.00871203291755115
5      X7020 0.00117002187594033      X4606 0.0202147984112503
>
```

8. Reference

D. R. Bickel, Z. Montazeri, P.-C. Hsieh, M. Beatty, S. J. Lawit, and N. J. Bate, "Gene network reconstruction from transcriptional dynamics under kinetic model uncertainty: A case for the second derivative," *Bioinformatics* doi:10.1093/bioinformatics/btp028 (2009).