A probabilistic dynamical model for quantitative inference of the regulatory mechanism of transcription

Guido Sanguinetti, Magnus Rattray and Neil D. Lawrence

October 24, 2013

Abstract

Quantitative estimation of the regulatory relationship between transcription factors and genes is a fundamental stepping stone when trying to develop models of cellular processes. This task, however, is difficult for a number of reasons: transcription factors‘ expression levels are often low and noisy, and many transcription factors are post-transcriptionally regulated. It is therefore useful to infer the activity of the transcription factors from the expression levels of their target genes.

Contents

1 Installing the chipDyno package

The recommanded way to install the *Bioconductor* package *chipDyno* is to use the `biocLite` function available in the *Bioconductor* website. This way of installation should ensure that all the dependencies are met.

```
> source ("www.//bioconductor.org/chipDyno.R")
> biocLite("chipDyno")
```

2 Loading the package and getting help

The first step in any *chipDyno* analysis is to load the package. This package can be loaded when the *R* console is ready. At the *R* console type the following command:

```
> library("chipDyno")
```

Command `help` can be used to get help on any function. For example, to get help on the `chipDynoLikeStatGrad` type the following (both of them have the similar output):

```
> help(chipDynoLikeStatGrad)
> ?chipDynoLikeStatGrad
```

3 Input data and preparations

If the data is in Affymetrix CEL files then it may requered to do some preprocessing. This CEL files can be extracted using the bioconductor package *puma* [Pearson et al., 2009]. *puma* usually stores the point expressione in '*_exprs.csv' file and defferent levels of unicertainities in '*_se.csv' files as Comma-separated values.

Load the point expression levels data. Here the data was stored in comma-seperated value (csv) file.

```
> data_file =
+   "/home/muhammad/Dropbox/CElegans/cluster/3exp_15dp_After_meeting/eset_cElegans_exprs.csv"
> data_file

[1] "/home/muhammad/Dropbox/CElegans/cluster/3exp_15dp_After_meeting/eset_cElegans_exprs.csv"

> data_dictionary <- read.table(data_file, header = TRUE, sep = ",", quote = "",
+                       dec = ".", , na.strings = "NA", colClasses = NA, nrows = -1, skip = 0,
+                       check.names = TRUE, fill = TRUE, strip.white = FALSE,
+                       blank.lines.skip = TRUE, comment.char = "#",
+                       allowEscapes = FALSE, flush = FALSE)
> data_dictionary[is.na(data_dictionary)] <- 0
> probeId = data_dictionary[,1]
> as.matrix(probeId[1:7],,1)

     [,1]
[1,] "171720_x_at"
[2,] "171721_x_at"
[3,] "171722_x_at"
[4,] "171723_x_at"
[5,] "171724_x_at"
[6,] "171725_x_at"
[7,] "171726_x_at"
```

In our experimental data set there was 3 replication of the same experiment with 5 experimental time points with different environmental conditions. So there was (3x5=) 15 time points. We will to reorganize this datapoint maintaining the time series for different experiment. For our this example we will consider the data obtained from the second experiment.

```
> data = data_dictionary[,2:ncol(data_dictionary)]
> data_sample1 <- cbind(data[,10],data[,1],data[,4],data[,7],data[,13])
> data_sample2 <- cbind(data[,11],data[,2],data[,5],data[,8],data[,14])
> data_sample3 <- cbind(data[,12],data[,3],data[,6],data[,9],data[,15])
> data = data_sample2
> data_sample1[1:7, 1:5]
```

```
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,]  7.538062  8.335555  8.171527  7.874888  7.784602
[2,]  9.985015 10.186255 10.390509 10.329048  9.536829
[3,]  9.138663 10.718470 11.301698 10.594997  9.836753
[4,] 13.793286 14.250379 14.365042 14.088584 13.461373
[5,]  9.929965 10.710961 10.794212 10.714703 10.011631
[6,] 13.052323 14.345205 14.552540 14.175887 13.203751
[7,]  5.356036  6.658377  6.501826  5.939661  5.797593
```

Load the the standard errors-

```
> vars_file =
+   "/home/muhammad/Dropbox/CElegans/cluster/3exp_15dp_After_meeting/eset_cElegans_se.csv"
> vars_dictionary <- read.table(vars_file, header = TRUE, sep = ",", quote = "",
+                    dec = ".", , na.strings = "NA", colClasses = NA, nrows = -1, skip = 0,
+                    check.names = TRUE, fill = TRUE, strip.white = FALSE,
+                    blank.lines.skip = TRUE, comment.char = "#",
+                    allowEscapes = FALSE, flush = FALSE)
> vars_dictionary[is.na(vars_dictionary)] <- 0
> vars = vars_dictionary[,2:ncol(vars_dictionary)]
> vars_sample1 <- cbind(vars[,10],vars[,1],vars[,4],vars[,7],vars[,13])
> vars_sample2 <- cbind(vars[,11],vars[,2],vars[,5],vars[,8],vars[,14])
> vars_sample3 <- cbind(vars[,12],vars[,3],vars[,6],vars[,9],vars[,15])
> vars = vars_sample2 # Choose the sample
```

Load the annotation file. From this annotation file we can find the gene names for corresponding probe Set ID

```
> annotation_file = "/home/muhammad/Dropbox/CElegans/annotation.txt"
> anno_dictionary <- read.table(annotation_file, header = FALSE, sep = "\t", quote = "",
+                    dec = ".", , na.strings = "NA", colClasses = NA, nrows = -1, skip = 0,
+                    check.names = TRUE, fill = TRUE, strip.white = FALSE,
+                    blank.lines.skip = TRUE, comment.char = "#",
+                    allowEscapes = FALSE, flush = FALSE)
> probeId2 = anno_dictionary[,1]
> geneName = anno_dictionary[,2]
> probeId=as.matrix(probeId,,1) # Probe set ID from experimental data
> probeId2=as.matrix(probeId2,,1) # Probe set ID from annotation file
> geneName=as.matrix(geneName,,1)
> geneName[1:7]
```

```
[1] "Y48E1B.14" "T01G9.2"   "F56B3.11"  "vit-4"     "cdc-25.1"  "col-160"   "cyp-23A1"
```

Remove the genes which expression level data at any time point is missing-

```
> index=array(0, dim<-c(nrow(data),1))
> for (i in 1: nrow(data)){
+   vec <- probeId[i]==probeId2
+         index[i]=colSums(vec)
+ }
> probeId=probeId[which(index!=0),];
> data=data[which(index!=0),];
> vars=vars[which(index!=0),];
```

```
> connectivity_file =
+   "/home/muhammad/Dropbox/CElegans/Connectivity/connectivity_matHS_LC.txt"
> connectivity <- read.table(connectivity_file, header = FALSE, sep = ",", dec = ".",
+                  na.strings = "NA", colClasses = NA, nrows = -1, skip = 0,
+                  check.names = TRUE, fill = TRUE, strip.white = FALSE,
+                  blank.lines.skip = TRUE, comment.char = "#",
+                  allowEscapes = FALSE, flush = FALSE)
> probeId3 = as.matrix(connectivity[,1],,1) # Probe ID from connectivity files
> dataChip = connectivity[,2:ncol(connectivity)]
> dataChip[is.na(dataChip)] <- 0
```

Collect the common gene expression data-

```
> index=array(0, dim<-c(nrow(data),1))
> for (i in 1: nrow(data)){
+   vec <- geneName[i]==probeId3
+         index[i]=sum(vec)
+ }
> data=data[which(index!=0),];
> probeId=as.matrix(probeId[which(index!=0)]);
> geneName=as.matrix(geneName[which(index!=0)],,1);
> vars=vars[which(index!=0),];
```

Collect the common gene expression on dataChip-

```
> index=array(0, dim<-c(nrow(dataChip),1))
> for (i in 1: nrow(dataChip)){
+         vec <- probeId3[i]==geneName
+         index[i]=sum(vec)
+ }
> dataChip=dataChip[which(index!=0),];
> probeId3=as.matrix(probeId3[which(index!=0)],,1);
```

avoid the duplicate entry

```
> tempIndex= duplicated(geneName)
> index=array(0, dim<-c(length(tempIndex),1))
> for (i in 1 : length(tempIndex)){
+         if (tempIndex[i]=="FALSE"){
+                 index[i]=1
+         }
+ }
> data=data[which(index!=0),];
> probeId=as.matrix(probeId[which(index!=0)]);
> geneName=as.matrix(geneName[which(index!=0)],,1);
> vars=vars[which(index!=0),];
> ##### Rearrange the dataChip data based on data entry
>
> dataChipNew = list()
> geneName2 = list()
> for (i in 1 : length(geneName)){
+         id = which (geneName[i]==probeId3)
+         dataChipNew = rbind(dataChipNew,dataChip[id,])
+         geneName2 = rbind(geneName2,probeId3[id,])
+ }
> #Create the binary matrix
>
> #save.image("test20130501.RData")
>
```

```
> #X= mat.or.vec(nrow(dataChip),ncol(dataChip))
> X= array(0, dim=c(nrow(dataChip),ncol(dataChip)))
> I <- c(which(dataChip>0)) ### ???????????????/ #TODO
> X[I] = 1
> ####
```

4 ??????????

Add the following to your package 'DESCRIPTION' file:

```
Suggests: BiocStyle
```

and this code chunk to the preamble of each vignette:

```
<<style, eval=TRUE, echo=FALSE, results=tex>>=
BiocStyle::latex()
@
```

Arguments to `latex` are passed to `options`. The default sets `width=90`, which is optimal for the default font size 11pt; for smaller/larger font size please consider increasing/decreasing this value.

The style features headers containing the title of the vignette as defined by `\title`. If you wish to override it, e.g. in the case of long titles, add the following line to the preamble of your vignette:

```
\renewcommand{\thetitle}{Short header title}
```

BiocStyle automatically attaches the following LaTeX packages: `color`, `fancyhdr`, `geometry`, `helvet`, `hyperref`, `sectsty`, `Sweave` and `titling`, so there is no need to reload them with `\usepackage`.

5 Markup commands

BiocStyle introduces the following additional markup styling commands useful in typical *Bioconductor* vignettes.

Software:
- `\R{}` and `\Bioconductor{}` to reference *R* software and the *Bioconductor* project.
- `\software{GATK}` to reference third-party software, e.g., *GATK*.

Packages:
- `\Biocpkg{IRanges}` for *Bioconductor* software packages, including a link to the release version landing page, *IRanges*.
- `\Biocannopkg{org.Hs.eg.db}` for *Bioconductor* annotation packages, including a link to the release version landing page, *org.Hs.eg.db*.
- `\Biocexptpkg{parathyroidSE}` for *Bioconductor* experiment data packages, including a link to the release version landing page, *parathyroidSE*.
- `\CRANpkg{data.table}` for *R* packages available on CRAN, including a link to the FHCRC CRAN mirror landing page, *data.table*.
- `\Rpackage{MyPkg}` for *R* packages that are *not* available on *Bioconductor* or CRAN, *MyPkg*.

Code:
- `\Rfunction{findOverlaps}` for functions `findOverlaps`.
- `\Robject{olaps}` for variables `olaps`.
- `\Rclass{GRanges}` when refering to a formal class *GRanges*.
- `\Rcode{log(x)}` for *R* code, `log(x)`.

Communication:
- `\comment{comment to the user}` communicates a *comment to the user*.

- `\warning{common pitfalls}` signals *warning: common pitfalls*.
- `\fixme{incomplete functionality}` provides an indication of *fixme: incomplete functionality*.

General:
- `\email{user@domain.com}` to provide a linked email address, `user@domain.com`.
- `\file{script.R}` for file names and file paths 'script.R'.

6 Sectioning: this is a section

Use `\tableofcontents` for a hyperlinked table of contents, `\section`, `\subsection`, `\subsubsection` for structuring your vignette.

6.1 This is a subsection

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

6.1.1 This is a subsubsection

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

7 Including a figure

Besides the usual LaTeX capabilities (`figure` environment and `includegraphics` command), 'Bioconductor.sty' defines a shortcut `\incfig{filename}{width}{shorttitle}{extendedcaption}`, which expects four arguments:

filename The name of the figure file, also used as the label by which the float can be refered to by `\ref{}`. Some *Sweave* and *knitr* options place figures in a sub-folder; the sub-folder should be included in the file name.

width Figure width.

shorttitle A short description, used in the list of figures and printed in bold as the first part of the caption.

extendedcaption Continuation of the figure caption.

Thus

```
<<figureexample,fig=TRUE,include=FALSE,width=4.2,height=4.6>>=
v = seq(0, 60i, length=1000)
plot(abs(v)*exp(v), type="l", col="Royalblue")
@
\incfig{LatexStyle-figureexample}{0.25\textwidth}{A curve.}
    {The code that creates this figure is shown in the code chunk.}
as shown in Figure~\ref{LatexStyle-figureexample}.
```

results in

```
> v = seq(0, 60i, length=1000)
> plot(abs(v)*exp(v), type="l", col="Royalblue")
```

as shown in Figure **??**. (Use `\ref{figure/figureexample}` with *knitr*).

8 Session info

Here is the output of `sessionInfo` on the system on which this document was compiled:

```
> toLatex(sessionInfo())
```

- R version 3.0.2 (2013-09-25), `x86_64-pc-linux-gnu`

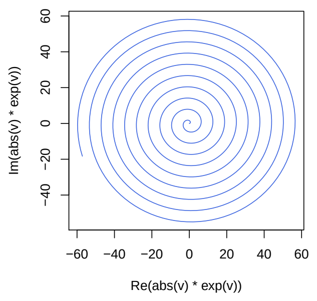

Figure 1: **A curve.** The code that creates this figure is shown in the code chunk.

- Locale: `LC_CTYPE=en_GB.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_GB.UTF-8`, `LC_COLLATE=en_GB.UTF-8`, `LC_MONETARY=en_GB.UTF-8`, `LC_MESSAGES=en_GB.UTF-8`, `LC_PAPER=en_GB.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_GB.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Loaded via a namespace (and not attached): BiocStyle 0.0.19, tools 3.0.2

References

[Pearson et al., 2009] Pearson, R., Liu, X., Sanguinetti, G., Milo, M., Lawrence, N., and Rattray, M. (2009). puma: a bioconductor package for propagating uncertainty in microarray analysis. *BMC Bioinformatics*. 10:211.