Introduction to LOXR: An R Package for Visualizing the Output of LOX

Erin Michet

July 13, 2012

1 What is LOX?

LOX is bioinformatics software written in Perl and C++. It uses Markov Chain Monte Carlo to estimate Levels Of gene eXpression from high-throughput expressed sequence data sets. Unlike other types of available software, LOX can handle data from multiple treatments or samples. You can download LOX from the Townsend Laboratory website.

For more information, see the *Bioinformatics* article about LOX, published June 2010.

2 What is LOXR?

LOXR is an R package for parsing and plotting the output of LOX. For these graphical capabilities, it relies upon the R package ggplot2. You can also use LOXR to implement LOXR from the R console. LOXR has functions for downloading, untarring, installing, and running LOX.

You can download LOXR from the Townsend Laboratory website or from Bioconductor.

3 Implementing the LOX analysis from the R console with LOXR:

3.1 Getting set up

To use LOXR to run the LOX analysis from the R console, you should begin by downloading, untarring, and installing the latest LOX distribution. This is an example workflow, commented out:

```
# loxdir <- '~/Desktop/LOXR_out/'
# commandline = paste('mkdir ',loxdir,sep='')
# system(commandline)
```

```
# LOX.download(loxdir)
# LOX.unpack(loxtarball,loxdir)
# LOX.install(loxfile)
```

Next, you should use the Perl script included in the untarred `LOX` folder to properly format your input file. As yet, this process cannot be done using `LOXR` functions, but do not skip this step—it is essential!

4 Running the analysis

Once you have properly formatted your input file, you can run the Markov chain Monte Carlo from the R console with the function `LOX.runanalysis`. This function takes two required arguments: the full path to the executable `LOX` file, and the full path to your properly-formatted input file. `LOX.runanalysis` can also take several optional arguments that set Markov chain Monte Carlo parameters such as chain sample size and length of burn-in. These options are described at length in the manual, or can be viewed using the function `LOX.help`. The outputs of this function are your `LOX` output files.

```
# LOX.help(loxexecutable);
# loxinputfile <- paste(loxfile,'/example/LOXinputfile.txt',sep='');
# LOX.run_analysis(loxexecutable,loxinputfile);
```

Be forewarned: this analysis can take some time. Using the default arguments, it takes about 40 minutes to analyze the example input file provided in the untarred `LOX` folder.

If you want to reproduce this analysis in your own R console, you can use the `LOXR` function `LOX.demo`, which performs this entire workflow on the example file included in your `LOX` download.

5 Getting Started with `LOXR`: Parsing Output

To parse output from `LOX`, the first thing you should is to run the function `LOX.parse.output`. You supply this function with the absolute path name of the folder containing your `LOX` output files, and `LOX.parse.output` automatically reads in the relevant files as data frames:

```
require("ggplot2")

## Loading required package: ggplot2

library(LOXR)
path <- system.file("doc", "example/", package = "LOXR")
LOX.parse.output(path)
```

```
## TREATMENTS:
## Reading file LOXinputfile.txt.lox
## Identified 2 treatments:  MM  SC
## Identified 5525 genes
## Saving levels of expression as LOX.treatments
##
## METHODS:
## Reading file LOXinputfile.txt.met
## Identified 2 methods:  oligodT  MTP
## Identified 5525 genes
## Saving levels of expression as LOX.methods
```

In the chunk of code above, we supplied `LOX.parse.output` with the absolute path to a folder of example output files included in the `LOXR` package. Then `LOX.parse.output` read in the output files `LOXinputfile.txt.lox` and `LOXinputfile.txt.met`. From `LOXinputfile.txt.lox`, it made a data frame `LOX.treatments` showing relative levels of expression under different treatments. From the file `LOXinputfile.txt.met`, it made a data frame `LOX.methods` showing relative levels of expression under different methods.

6 Making plots

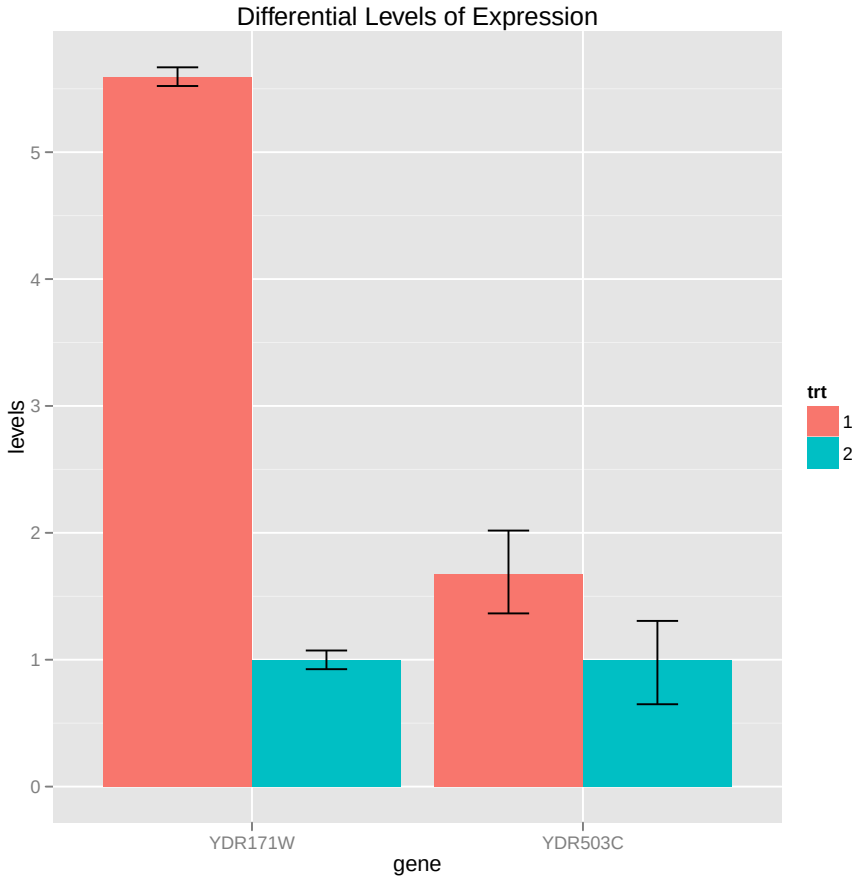
`LOXR` can take those data frames and make elegant data visualizations. These plotting capibilities rely upon the graphics package `ggplot2`.

6.1 Barplots with `LOXR`

To make a barplot of levels of expression for one or several genes, use `LOX.barplot`. You supply the data frame (`LOXinputfile.txt.lox` or `LOXinputfile.txt.met`) and a vector containing some gene IDs; `LOX.barplot` displays the relative levels of gene expression for these genes across the different methods or treatments.

Here is an example of a `LOXR` barplot:

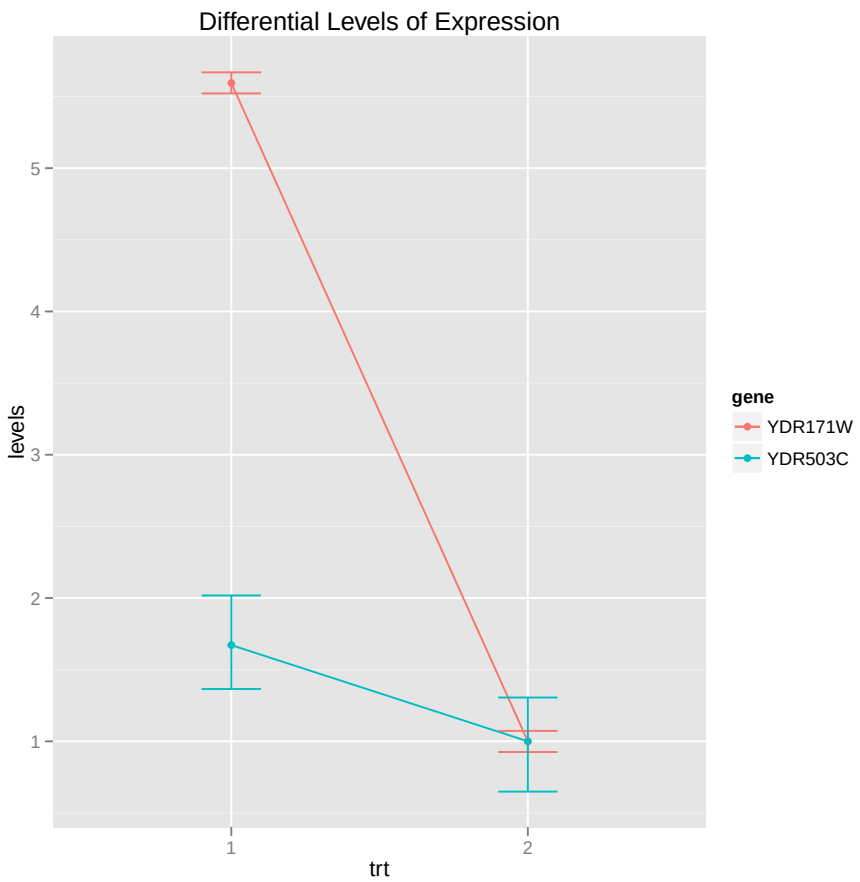
```r
LOX.barplot(LOX.treatments, c("YDR171W", "YDR503C"))
```



6.2 Time series with LOXR

To display this information as a time series instead of a barplot, use `LOX.timeseries`:

```
LOX.timeseries(LOX.treatments, c("YDR171W", "YDR503C"))
```



7 Identifying genes associated with significant p-values

When analyzing your data with `LOX`, you have the option to output tables of p-values describing the probability that level of expression was greater under one treatment or method than under another. `LOXR` can parse these p-values quickly and easily to help you identify genes for which there are interesting results:

```
# returns a vector of IDs for which P(MTP>oligodT) <= 0.001:
LOX.parse.pvalues(path, "MTP", 0.001)

## Scanning output file with extension *\MTP.pvalue$
##  Saved p-values for MTP in MTP.pvalue
##  Condition: P(MTP>oligodT) <= 0.001
```

```
##   Number of significant p-values:  939
##   Saved their gene IDs in:  1_significants
```