The *Rnits* package for normalization and inference of differential expression in time series microarray data.

Dipen P. Sangurdekar *

October 13, 2014

Contents

1 Introduction

Gene expression studies interrogating various specific aspects of cellular physiology – cell cycle regulation, dynamic response to perturbations or natural circadian rhythms – involve samples being collected over a time series. A number of methods have been developed to infer differentially expressed genes from temporal transcriptional data. One class of methods involves extending static differential expression methods like ANOVA ([1]) and empirical Bayes ([2, 3]) to temporal data sets. Another approach is to use functional data analysis methods to model the time series data as linear combinations of basis functions (splines) ([4, 5, 6, 7, 8, 9]). Differential expression is then inferred by hypothesis testing on the basis coefficients, empirical Bayes analysis ([10, 11]) or by hypothesis testing between two groups ([5, 9, 12, 8, 13])

The method proposed by [8] and [9] utilizes a model driven approach to infer differential expression. The null hypothesis is that for a given gene, the difference in profiles between two treatments (or their deviation from the population average for the gene) is random. They model this scenario by fitting a single average curve, constructed from natural cubic spline basis curves, onto the two profiles. Under the alternative hypothesis, the two profiles arise from distinct underlying biological processes and this is modeled by using different models for each profile. If the gene is significantly DE, the quality of fit (as measured by the sum of squares of the regression residuals) under the alternative model is better than

*dipen.sangurdekar@gmail.com

under the null model. To get the significance of the fit statistic, the method instead uses a bootstrapping based approach to simulate expression profiles under the null model and empirically estimates the frequency of the simulated fit statistic being higher than the observed one. A critical parameter in this modeling approach is the dimensionality of the spline basis p. [8] propose a method to pick an optimal model that minimizes the generalized cross-validation error on the top eigenvectors inferred from the data. In the current work, we extend the previous work by proposing an alternative approach to model selection for inference in complex temporal data. We identify distinct optimal models for clustered subsets of genes sharing complex patterns based on the statistical power to detect genes with differential trajectories. We implement this approach in a statistical package written in *R*, *Rnits*, to assist the research community in the end-to-end genomic analysis of time course experiments, from raw data import normalization, inference of DE and visualization of results.

2 Method

2.1 Data preprocessing and normalization

Raw or pre-processed expression data can be imported into *Rnits* either as a data matrix, an RGList object created by the package *limma*, an AffyBatch object (package *affy*) or an eSet object downloaded from NCBI's Gene Expression Omnibus by package *GEOquery*. For all data formats, *Rnits* offers probe filtering and normalization options that are native to the respective packages. Processed data, along with probe and sample phenotype data, is stored as an *Rnits* class object, which inherits all methods from the *ExpressionSet* object in *R* that is commonly used for storing expression data and metadata.This allows expression and metadata retrieval from the object using the standard methods defined for *ExpressionSet* objects. In addition to the standard metadata requirements for the *ExpressionSet* object, building *Rnits* objects requires columns labeled "Sample" and "Time" in the phenotype data matrix, and a column labeled "GeneName" for the probe data matrix (for gene level summarization). One of the time series experiments may be labeled as 'control' for the purpose of clustering data. Inference can be done at the probe level or at the gene level if multiple probes represent a single gene. For gene level analysis, probe data is collapsed into a gene level summary using the robust Tukey Biweight estimator, which penalizes outlier values. For analysis, the distinct time series experiments must have been sampled at the same points.

2.2 B-spline basis model based hypothesis testing

The analysis approach is to model the time course data under the null hypothesis that individual gene expression trajectories from distinct sets is a realization of the same underlying basis trajectory modeled as a B-spline curve. Under the alternative hypothesis, each series is modeled as a distinct underlying basis. [8] et al describe a method of selecting the optimal B-spline basis model $\mathcal{L}$. For each data set, the top eigenvectors are computed and the set of models $\overline{\mathcal{L}}$ are used to compute the generalized cross-validation error using the eigenvectors as predicted variables. For each data set, the model that minimizes the cross-validated error across all top components is chosen as optimal and the largest optimal model among all data sets is chosen (*Rnits.gcv*). We extend this work to develop a parallel approach for selecting the optimal model based on the power of inferring differentially expressed genes. *Rnits.power*. A series of candidate models $\overline{\mathcal{L}}$ are evaluated on the entire data set and optimal models are selected based on the distribution of the p-values of fit.

2.3 Clustering

In the first step of this approach, the genes (or probes) may be clustered. Clustering allows models of varying complexity to be selected to represent the diverse expression patterns observed within the data. Each gene/probe is assigned to one of K distinct clusters using k-means clustering on gene centered data (either all the series or based on a single "control" series). The total number of clusters may be provided by the user as an argument or is determined empirically as twice the number of eigenvectors required to explain 70% of variation in the time series labeled as control or the entire data set. If any of the resulting clusters have less than 500 genes, clustering is iteratively repeated with one less initial cluster centroid.

2.4 Candidate model evaluation

The size r of the individual time series experiment determines the set of candidate B-spline models that can be investigated. Each model $\mathcal{L}(c,p,r)$ is defined by its degree of curvature *c* and the degrees of freedom *p* i.e. the number of basis splines used. The rank constrains the curvature and the number of basis splines as follows. For a given matrix with rank r, the set of candidate models $\overline{\mathcal{L}}$ were:

$$\overline{\mathcal{L}}(c,p,r)\begin{cases}3 \leq p \leq min(8, r-2)\\ 2 \leq c \leq min(5, p-1)\end{cases} \tag{1}$$

where each model was constrained to have a maximum degree of curvature of 5 and maximum number of basis splines at 8. Whenever sufficient degrees of freedom were available to allow manual placement of knots ($p - c - 1 > 0$), knots were chosen based on which time points had the maximum inflection. For each cluster k (or for the entire data), each model $\mathcal{L} \in \overline{\mathcal{L}}$ or the optimal model obtained by generalized cross-validation was evaluated as follows:

- The gene expression data matrix for series *s* can be represented as $\mathbf{Y}_{m_k n}$ where *m* represents the number of probes (or genes) in cluster k, and *n* equals size of the data series. The candidate model $\mathcal{L}$ is represented by its B-spline basis $\mathbf{X}^{\mathcal{L}}_{pn}$. For models with larger basis splines, knot placement is guided by regions of higher curvature.
- Under the null model of no differential expression between the different time series, the distinct profiles are assumed to be generated from the same underlying curve, and the expression matrix for all series $\mathbf{Y}_{mn}$ is fit as follows:

$$\mathbf{Y}_{mn} = \widehat{\beta_{mn}}\mathbf{X}^{\mathcal{L}}_{pn} + \mathbf{E}_{mn} = \widehat{\mathbf{Y}}_{null} + \mathbf{E}_{null} \tag{2}$$

- Under the alternate hypothesis, for each individual series $s \in (1, S)$, we fit a distinct curve and the individual data matrix is fit as follows

$$\mathbf{Y}_{mr} = \widehat{\beta_{mr}}\mathbf{X}^{\mathcal{L}}_{pr} + \mathbf{E}_{mr} = \widehat{\mathbf{Y}}_{alt} + \mathbf{E}_{alt} \tag{3}$$

- The two models are evaluated using a ratio statistic $F^{\mathcal{L}}_k$

$$\mathcal{F}^{\mathcal{L}}_k = \frac{\sum^r \mathbf{E}^2_{null} - \sum^s \sum^r \mathbf{E}^2_{alt}}{\sum^s \sum^r \mathbf{E}^2_{alt} + s_0} \tag{4}$$

  where $\sum^r \mathbf{E}^2_{null}$ is the vector of sum of square of residuals under the null hypothesis for all genes in cluster k and $\sum^s \sum^r \mathbf{E}^2_{alt}$ is the sum of squared residuals for each gene in cluster k added over all series s. We add a variance stabilizing factor to the denominator s_0 computed as described by [14].

To compute p-values for the ratio statistic, we use the bootstrap approach as described by [8]. First the residuals under the alternate model E_{alt} are standardized to correct for heteroskedasticity for each series. This penalizes the models with larger basis splines by increasing the effective size of their residuals.

$$\mathbf{E}'_{alt} = \frac{\mathbf{E}_{alt}}{\sqrt{1 - X(X^T X)^{-1} X^T}} \tag{5}$$

For each bootstrap iteration b, a synthetic null data set is constructed by adding the bootstrapped studentized residuals to the null fit.

$$\mathbf{Y}^*_{null} = \widehat{\mathbf{Y}}_{null} + \mathbf{E}'^*_{alt} \tag{6}$$

Eqn (2)-(4) are repeated for each iteration and ratio statistics $\mathcal{F}^{\mathcal{L}*}_k$ under the null hypothesis of no differential expression are computed. Empirical p-values are computed for the observed ratio statistic based on the distribution of the simulated ones under the null hypothesis.

$$\mathcal{P}^{\mathcal{L}}_k = \frac{\#(\mathcal{F}^{\mathcal{L}*}_k > \mathcal{F}^{\mathcal{L}}_k)}{B} \tag{7}$$

where B are the number of bootstrap iterations.

2.5 Model Selection

For each cluster k, p-values are similarly calculated for each model $\mathcal{L} \in \overline{\mathcal{L}}$. The significance of each model fit is a function of the size of the studentized residuals under the null and alternative hypotheses. While more complex models (larger number of basis functions) may yield smaller residuals during the alternative fit, these may get heavily penalized during correction for heteroskedasticity, leading to low statistical power. At the other extreme, an insufficiently complex model may not effectively capture the complexity of the expression trajectories that may lead to larger residuals and lack of power. To alleviate this effect, we select the model $\mathcal{L}$ which results in the maximum power, as determined by the number of genes inferred at an estimated False Discovery Rate $< 5\%$

3 Expression profiling in Yeast in a chemostat perturbation experiment

We applied the *Bspline* methods to time course expression data tracking the glucose perturbation responses of a wild type yeast strain grown at steady state in a chemostat with galactose as the carbon source ([15]). Two glucose concentrations (effective concentration of 0.2 g/l and 2 g/l) were used as pulses and responses were tracked at 10, 15, 20, 30, 45, 90, 120 and 150 minutes post each treatment. We obtained the published microarray data from NCBI Gene Expression Omnibus (GEO accession GSE4158) and analyzed it using *Rnits*.

3.1 Data preparation

First, we download the data from GEO and format it.

```
# Download NCBI GEO data with GEOquery
library(knitr)
rm(list = ls())
library(GEOquery)
```

```
## Loading required package:  Biobase
## Loading required package:  BiocGenerics
## Loading required package:  parallel
##
## Attaching package:  'BiocGenerics'
##
## The following objects are masked from 'package:parallel':
##
##    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##    clusterExport, clusterMap, parApply, parCapply, parLapply,
##    parLapplyLB, parRapply, parSapply, parSapplyLB
##
## The following object is masked from 'package:stats':
##
##    xtabs
##
## The following objects are masked from 'package:base':
##
##    Filter, Find, Map, Position, Reduce, anyDuplicated, append,
##    as.data.frame, as.vector, cbind, colnames, do.call, duplicated,
##    eval, evalq, get, intersect, is.unsorted, lapply, mapply, match,
##    mget, order, paste, pmax, pmax.int, pmin, pmin.int, rank, rbind,
##    rep.int, rownames, sapply, setdiff, sort, table, tapply, union,
##    unique, unlist, unsplit
##
```

```
## Welcome to Bioconductor
##
##    Vignettes contain introductory material; view with
##    'browseVignettes()'.  To cite Bioconductor, see
##    'citation("Biobase")', and for packages 'citation("pkgname")'.
##
## Setting options('download.file.method.GEOquery'='curl')
```

```
library(stringr)
library(Rnits)
```

```
## Loading required package:  ggplot2
## Loading required package:  limma
##
## Attaching package:  'limma'
##
## The following object is masked from 'package:BiocGenerics':
##
##    plotMA
```

```
gds <- getGEO("GSE4158", AnnotGPL = FALSE)[[1]]
```

```
## ftp://ftp.ncbi.nlm.nih.gov/geo/series/GSE4nnn/GSE4158/matrix/
## Found 1 file(s)
## GSE4158_series_matrix.txt.gz
## File stored at:
## /tmp/RtmpfsKhwu/GPL3415.soft
```

```
class(gds)
```

```
gds
```

```
# Extract non-replicate samples
pdata <- pData(gds)
filt <- pdata$characteristics_ch2 %in% names(which(table(pdata$characteristics_ch2) ==
    2))
gds <- gds[, filt]
pdata <- pData(gds)
time <- as.numeric(str_extract(pdata$characteristics_ch2, "\\d+"))
```

```
sample <- rep("2g/l", length(time))
sample[grep("0.2g/l", gds[["title"]])] <- "0.2g/l"

# Format phenotype data with time and sample information
gds[["Time"]] <- time
gds[["Sample"]] <- sample
dat <- gds
fData(dat)["Gene Symbol"] <- fData(dat)$ORF
```

3.2 Running Rnits

3.2.1 Build Rnits object from *ExpressionSet* or *data matrix*

Next we build the *Rnits* object

```
# Build rnits data object from formatted data (samples can be in any order) and
# between-array normalization.
rnitsobj <- build.Rnits(dat[, sample(ncol(dat))], logscale = TRUE, normmethod = "Between")
```

```
## Between Array normalization
## Data set with 2 time series
## Dataset with  10752  features and  16  samples
```

```
rnitsobj
```

```
## Rnits (storageMode: lockedEnvironment)
## assayData: 10752 features, 16 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: GSM95010 GSM95011 ... GSM94998 (16 total)
##   varLabels: title geo_accession ... Sample (38 total)
##   varMetadata: labelDescription
## featureData
##   featureNames: 1 2 ... 10752 (10752 total)
##   fvarLabels: ID METACOLUMN ... GeneName (12 total)
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation:
```

```
# Alternatively, we can also build the object from just a data matrix, by
# supplying the 'probedata' and 'phenodata' tables
datdf <- exprs(dat)
rownames(datdf) <- fData(dat)$ID
probedata <- data.frame(ProbeID = fData(dat)$ID, GeneName = fData(dat)$ORF)
phenodata <- pData(dat)
rnitsobj <- build.Rnits(datdf, probedata = probedata, phenodata = phenodata, logscale = TRUE,
    normmethod = "Between")
```

```
## Log scale is TRUE
## Between Array normalization
## Data set with 2 time series
## Dataset with  10752  features and  16  samples
```

```
# Extract normalized expression values
lr <- getLR(rnitsobj)
```

```
head(lr)
```

```
##      GSM95010    GSM95011    GSM95012  GSM95013    GSM95002  GSM95004
## 1 -0.99269970 -0.84741081 -0.22867060 0.5941574 -1.18590131 0.5137818
## 2  0.12448779  0.76758977  1.43044492 0.9482546 -0.04944478 0.4212989
## 3  0.05362765 -0.69975822 -0.70887206 0.2560923  0.62460336 0.3294558
## 4 -0.59017581 -0.47129159 -0.08102022 0.7885572 -1.03973077 0.8060511
## 5 -1.05032467  0.04912668  0.09804767 0.4690431 -1.59731454 0.4243162
## 6  0.84482623  1.10519264  1.25240820 1.6704265 -0.22307705 0.3077783
##      GSM95006    GSM95008    GSM94988   GSM94989    GSM94991     GSM94992
## 1 -0.31852729 -0.19303436 -0.20592568 -0.0797288 -0.05466009  0.005671936
## 2  0.31219698  0.13219700  0.62636186  0.6992206  0.78594481  0.684897162
## 3 -0.09733628 -0.39948994 -0.26945402 -0.7516256 -1.01418075 -1.066315054
## 4  0.57356735  0.43055165 -0.07012699 -0.3610244 -0.97771053 -0.841308620
## 5  0.38912313  0.09812534  0.25315615  0.0561665  0.17523029  0.273787476
## 6  0.24570856 -0.19826540  0.85110526  0.7232386  1.04992619  1.053762306
##     GSM94993  GSM94994   GSM94996  GSM94998
## 1  0.3418725 -2.487601  0.1680312 0.2977183
## 2  0.8925079        NA  1.1047615 0.7014654
## 3 -1.0909498        NA  0.1363547 0.7593684
## 4 -0.1664416        NA -0.2371354        NA
## 5  0.1791928        NA  0.3288013 0.5037860
## 6  1.0160011 -1.292664  0.8986091 0.7248260
```

```
# Impute missing values using K-nn imputation
lr <- getLR(rnitsobj, impute = TRUE)
```

```
## Warning in knnimp(x, k, maxmiss = rowmax, maxp = maxp):  1635 rows with more than 50 % entries
missing;
##  mean imputation used for these rows
```

```
## Cluster size 9117 broken into 6018 3099
## Cluster size 6018 broken into 4290 1728
## Cluster size 4290 broken into 428 3862
## Done cluster 428
## Cluster size 3862 broken into 1966 1896
## Cluster size 1966 broken into 1191 775
## Done cluster 1191
## Done cluster 775
## Done cluster 1966
## Cluster size 1896 broken into 1298 598
## Done cluster 1298
## Done cluster 598
## Done cluster 1896
## Done cluster 3862
## Done cluster 4290
## Cluster size 1728 broken into 1542 186
## Cluster size 1542 broken into 976 566
## Done cluster 976
## Done cluster 566
## Done cluster 1542
## Done cluster 186
## Done cluster 1728
## Done cluster 6018
## Cluster size 3099 broken into 1001 2098
```

```
## Done cluster 1001
## Cluster size 2098 broken into 522 1576
## Done cluster 522
## Cluster size 1576 broken into 941 635
## Done cluster 941
## Done cluster 635
## Done cluster 1576
## Done cluster 2098
## Done cluster 3099
```

```
head(lr)
```

```
##      GSM95010    GSM95011    GSM95012  GSM95013    GSM95002  GSM95004
## 1 -0.99269970 -0.84741081 -0.22867060 0.5941574 -1.18590131 0.5137818
## 2  0.12448779  0.76758977  1.43044492 0.9482546 -0.04944478 0.4212989
## 3  0.05362765 -0.69975822 -0.70887206 0.2560923  0.62460336 0.3294558
## 4 -0.59017581 -0.47129159 -0.08102022 0.7885572 -1.03973077 0.8060511
## 5 -1.05032467  0.04912668  0.09804767 0.4690431 -1.59731454 0.4243162
## 6  0.84482623  1.10519264  1.25240820 1.6704265 -0.22307705 0.3077783
##      GSM95006    GSM95008    GSM94988   GSM94989    GSM94991     GSM94992
## 1 -0.31852729 -0.19303436 -0.20592568 -0.0797288 -0.05466009  0.005671936
## 2  0.31219698  0.13219700  0.62636186  0.6992206  0.78594481  0.684897162
## 3 -0.09733628 -0.39948994 -0.26945402 -0.7516256 -1.01418075 -1.066315054
## 4  0.57356735  0.43055165 -0.07012699 -0.3610244 -0.97771053 -0.841308620
## 5  0.38912313  0.09812534  0.25315615  0.0561665  0.17523029  0.273787476
## 6  0.24570856 -0.19826540  0.85110526  0.7232386  1.04992619  1.053762306
##     GSM94993   GSM94994   GSM94996  GSM94998
## 1  0.3418725 -2.4876007  0.1680312 0.2977183
## 2  0.8925079  0.8512363  1.1047615 0.7014654
## 3 -1.0909498 -0.2650156  0.1363547 0.7593684
## 4 -0.1664416 -0.1232633 -0.2371354 0.0988995
## 5  0.1791928  0.6795649  0.3288013 0.5037860
## 6  1.0160011 -1.2926636  0.8986091 0.7248260
```

3.2.2 Fitting the model

Next fit the model using gene-level summarization and by clustering all genes.

```
# Fit model using gene-level summarization
rnitsobj <- fit(rnitsobj, gene.level = TRUE, clusterallsamples = TRUE)
```

```
## Collapsing probes into gene level log-ratios
## 3066  probes had no gene names. Removing them from further processing.
## Found  6342  unique genes.
## ############################
## Analyzing Models
## Computed number of clusters is 12
## Repeating clustering to eliminate small clusters
## Repeating clustering to eliminate small clusters
## Repeating clustering to eliminate small clusters
## Repeating clustering to eliminate small clusters
## Repeating clustering to eliminate small clusters
## Repeating clustering to eliminate small clusters
## Repeating clustering to eliminate small clusters
## Number of clusters is: 5
```

```
## ############################
## Cluster 1 of size 1000
## Cluster 2 of size 504
## Cluster 3 of size 1708
## Cluster 4 of size 2602
## Cluster 5 of size 528
## Estimated proportion of null genes is 49.42 %
```

The model may be fit using no clustering of samples, or by precomputing the optimal model using the method of [8].

```
rnitsobj_nocl <- fit(rnitsobj, gene.level = TRUE, clusterallsamples = FALSE)

opt_model <- calculateGCV(rnitsobj)
rnitsobj_optmodel <- fit(rnitsobj, gene.level = TRUE, model = opt_model)
```

3.2.3 Get top genes and other fit summary statistics

```
# Get pvalues from fitted model
pval <- getPval(rnitsobj)
head(pval)
```

```
##     YAL001C     YAL003W     YAL005C     YAL008W     YAL010C     YAL012W
## 0.333647541 0.013530000 0.008665105 0.218993082 0.435983859 0.047720000
```

```
# Get ratio statistics from fitted model
stat <- getStat(rnitsobj)
head(stat)
```

```
##     YAL001C     YAL003W     YAL005C     YAL008W     YAL010C     YAL012W
## 0.019826266 1.134269251 0.570045135 0.053763321 0.004774746 0.624189085
```

```
# If clustering was used, check the cluster gene distribution
table(getCID(rnitsobj))
```

```
##
##    1    2    3    4    5
## 1000  504 1708 2602  528
```

```
# P-values, ratio statistics and cluster ID's can be retrieved for all genes
# together
fitdata <- getFitModel(rnitsobj)
head(fitdata)
```

```
##         Ratio.statistic     p.value clusterID
## YAL001C     0.019826266 0.333647541         3
## YAL003W     1.134269251 0.013530000         1
## YAL005C     0.570045135 0.008665105         3
## YAL008W     0.053763321 0.218993082         4
## YAL010C     0.004774746 0.435983859         4
## YAL012W     0.624189085 0.047720000         1
```

```
# View summary of top genes
summary(rnitsobj, top = 10)
```

```
##         Rank    Name Statistic p.value FDR.percent ClusterID Model.degree
## YLR108C    1 YLR108C    12.910       0           0         1            5
## YGL157W    2 YGL157W     9.327       0           0         1            5
## YDR277C    3 YDR277C     9.177       0           0         1            5
```

```
## YLR044C    4 YLR044C     8.783       0           0         1            5
## YLR377C    5 YLR377C     7.996       0           0         2            2
## YOL126C    6 YOL126C     6.536       0           0         2            2
## YNL117W    7 YNL117W     5.954       0           0         2            2
## YJL045W    8 YJL045W     5.081       0           0         2            2
## YLR411W    9 YLR411W     5.005       0           0         2            2
## YJR048W   10 YJR048W     2.562       0           0         4            2
##         Model.df
## YLR108C        6
## YGL157W        6
## YDR277C        6
## YLR044C        6
## YLR377C        3
## YOL126C        3
## YNL117W        3
## YJL045W        3
## YLR411W        3
## YJR048W        3
```

```
# Extract data of top genes (5% FDR)
td <- topData(rnitsobj, fdr = 5)
```

```
## 1194
```

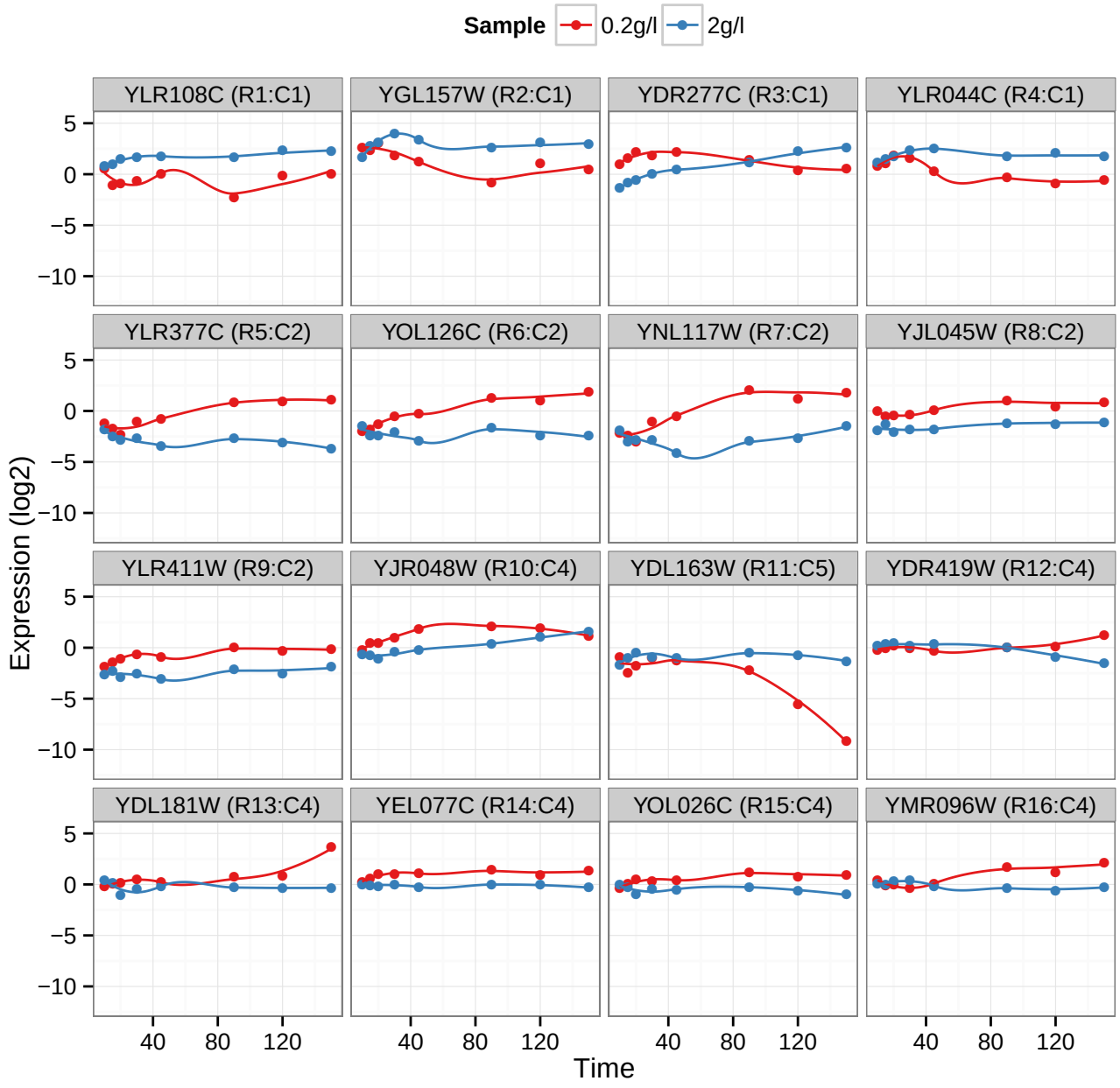
```
head(td)
```

```
##          0.2g/l_10 0.2g/l_15 0.2g/l_20  0.2g/l_30   0.2g/l_45  0.2g/l_90
## YLR108C  0.5685404 -1.103856 -0.920593 -0.6556341  0.03927246 -2.2515196
## YGL157W  2.6492136  2.318524  3.000039  1.8748751  1.25321768 -0.8536410
## YDR277C  0.9715954  1.605899  2.144856  1.8467730  2.19654505  1.4071820
## YLR044C  0.8418638  1.074910  1.873320  1.5850722  0.26911068 -0.2902454
## YLR377C -1.2391829 -1.767067 -2.287780 -1.0063690 -0.82232518  0.8824731
## YOL126C -1.9560429 -1.823695 -1.256064 -0.5267378 -0.30743812  1.3158447
##         0.2g/l_120  0.2g/l_150    2g/l_10    2g/l_15    2g/l_20     2g/l_30
## YLR108C -0.1540562  0.07452528  0.8272629  0.9679145  1.4876804  1.65439917
## YGL157W  1.0488212  0.50091541  1.6802568  2.7494918  3.1017524  4.00086971
## YDR277C  0.3663740  0.51117759 -1.3601205 -0.8216271 -0.5638514  0.04898597
## YLR044C -0.8900869 -0.58703064  1.1211933  1.5203426  1.7523680  2.33330558
## YLR377C  0.9316106  1.07983128 -1.8115118 -2.5088787 -2.8089979 -2.70192631
## YOL126C  1.0227485  1.84276981 -1.4861348 -2.3854609 -2.4288680 -2.07939884
##            2g/l_45   2g/l_90  2g/l_120  2g/l_150
## YLR108C  1.7804393  1.669185  2.322039  2.263248
## YGL157W  3.3594363  2.626783  3.101510  2.974053
## YDR277C  0.4385779  1.141847  2.291573  2.599364
## YLR044C  2.5023404  1.734326  2.062824  1.783147
## YLR377C -3.4228364 -2.697300 -3.086772 -3.658583
## YOL126C -2.9615985 -1.614670 -2.408455 -2.435676
```

Next we plot the results for the top 16 differentially expressed genes by FDR

```
# Plot top genes trajectories
plotResults(rnitsobj, top = 16)
```

```
## Next: 1 of 1
```




```
sessionInfo()
```

```
## R version 3.1.1 Patched (2014-09-25 r66681)
## Platform: x86_64-unknown-linux-gnu (64-bit)
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
```

```
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] Rnits_1.0.0         limma_3.22.0        ggplot2_1.0.0
## [4] stringr_0.6.2       GEOquery_2.32.0     Biobase_2.26.0
## [7] BiocGenerics_0.12.0 knitr_1.7
##
## loaded via a namespace (and not attached):
##  [1] BiocInstaller_1.16.0  BiocStyle_1.4.0       MASS_7.3-35
##  [4] RColorBrewer_1.0-5    RCurl_1.95-4.3        Rcpp_0.11.3
##  [7] XML_3.98-1.1          affy_1.44.0           affyio_1.34.0
## [10] boot_1.3-13           codetools_0.2-9       colorspace_1.2-4
## [13] digest_0.6.4          evaluate_0.5.5        formatR_1.0
## [16] grid_3.1.1            gtable_0.1.2          highr_0.3
## [19] impute_1.40.0         labeling_0.3          munsell_0.4.2
## [22] plyr_1.8.1            preprocessCore_1.28.0 proto_0.3-10
## [25] qvalue_1.40.0         reshape2_1.4          scales_0.2.4
## [28] splines_3.1.1         tools_3.1.1           zlibbioc_1.12.0
```

4 Summary

Dynamic changes in transcriptional programs to changing stimulus or cell cycle stages necessitate an experimental design that tracks expression changes over a period of time. There are often constrains on the replication levels in such designs, posing a challenging inference problem. A number of methods have been developed to infer differential expression between conditions. We extend a previously developed method by [8] of using B-splines to perform hypothesis testing. We extended the method to include a model selection step that optimizes for distinct subsets within the expression data, corresponding to diverse transcriptional regulatory patterns in cells. We present this method as an comprehensive analysis R package *Rnits* to enable the data import, pre-processing, normalization, analysis and visualization of gene expression data from a variety of sources (GEO, raw or normalized data tables).

References

[1] Ping Ma, Wenxuan Zhong, and Jun S. Liu. Identifying differentially expressed genes in time course microarray data. *Statistics in Biosciences*, 1(2):144–159, 2009.

[2] Martin Aryee, Jose Gutierrez-Pabello, Igor Kramnik, Tapabrata Maiti, and John Quackenbush. An improved empirical bayes approach to estimating differential gene expression in microarray time-course data: Betr (bayesian estimation of temporal regulation). *BMC bioinformatics*, 10(1):409, 2009.

[3] Yu Chuan Tai and Terence P. Speed. A multivariate empirical bayes statistic for replicated microarray time course data. *The Annals of Statistics*, 34(5):2387–2412, 2006.

[4] Ziv Bar-Joseph, Georg Gerber, David K. Gifford, Tommi S. Jaakkola, and Itamar Simon. A new approach to analyzing gene expression time series data. In *Proceedings of the sixth annual international conference on Computational biology*, pages 39–48. ACM, 2002.

[5] Ziv Bar-Joseph, Georg Gerber, Itamar Simon, David K. Gifford, and Tommi S. Jaakkola. Comparing the continuous representation of time-series expression profiles to identify differentially expressed genes. *Proceedings of the National Academy of Sciences*, 100(18):10146–10151, 2003.

[6] Norma Coffey and John Hinde. Analyzing time-course microarray data using functional data analysis-a review. *Statistical Applications in Genetics and Molecular Biology*, 10(1), 2011.

[7] Xueli Liu and Mark C. K. Yang. Identifying temporally differentially expressed genes through functional principal components analysis. *Biostatistics*, 10(4):667–679, 2009. URL: http://biostatistics.oxfordjournals.org/content/10/4/667.abstract, doi:10.1093/biostatistics/kxp022.

[8] John D. Storey, Wenzhong Xiao, Jeffrey T. Leek, Ronald G. Tompkins, and Ronald W. Davis. Significance analysis of time course microarray experiments. *Proceedings of the National Academy of Sciences of the United States of America*, 102(36):12837–12842, 2005.

[9] Jeffrey T. Leek, Eva Monsen, Alan R. Dabney, and John D. Storey. Edge: extraction and analysis of differential gene expression. *Bioinformatics*, 22(4):507–508, 2006.

[10] Claudia Angelini, Daniela De Canditiis, Margherita Mutarelli, and Marianna Pensky. A bayesian approach to estimation and testing in time-course microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, 6(1), 2007.

[11] F. Hong and H. Li. Functional hierarchical models for identifying genes with different time-course expression profiles. *Biometrics*, 62(2):534–544, 6 2006. doi:10.1111/j.1541-0420.2005.00505.x.

[12] Christopher Minas, Simon J. Waddell, and Giovanni Montana. Distance-based differential analysis of gene curves. *Bioinformatics*, 27(22):3135–3141, 2011. URL: http://bioinformatics.oxfordjournals.org/content/27/22/3135.abstract, doi:10.1093/bioinformatics/btr528.

[13] Chongqi Zhang, Heng Peng, and Jin-Ting . T. Zhang. Two samples tests for functional data. *Communications in StatisticsTheory and Methods*, 39(4):559???578, 2 2010. doi:10.1080/03610920902755839.

[14] Virginia Goss Tusher, Robert Tibshirani, and Gilbert Chu. Significance analysis of microarrays applied to the ionizing radiation response. *Proceedings of the National Academy of Sciences*, 98(9):5116–5121, 2001.

[15] Michal Ronen and David Botstein. Transcriptional response of steady-state yeast cultures to transient perturbations in carbon source. *Proceedings of the National Academy of Sciences of the United States of America*, 103(2):389–394, 2006.