Bioconductor's edge package
Version 0.99.0

John D. Storey and Andrew J. Bass
Princeton University
http://genomine.org/contact.html

July 9, 2014

Contents

1 Introduction

`edge` is a package for significance analysis of DNA micro-array experiments and is able to identify genes that are differentially expressed between two or more different biological conditions (e.g., healthy versus diseased tissue). `edge` performs significance analysis and uses the odp-statistic from the Optimal Discovery Procedure (ODP) for significance testing. Whereas previously existing methods employ statistics that are essentially designed for testing one gene at a time (e.g., t-statistics and F-statistics), the ODP uses information across all genes to test for differential expression.

The improvements in power are substantial; Figure 1 shows a comparison between edge and five leading software packages, based on a well-known breast cancer expression study (Hedenfalk et al. 2001). In addition to identifying deferentially expressed genes, `edge` includes implementations of popular packages such as `snm`, `sva` and `qvalue`.

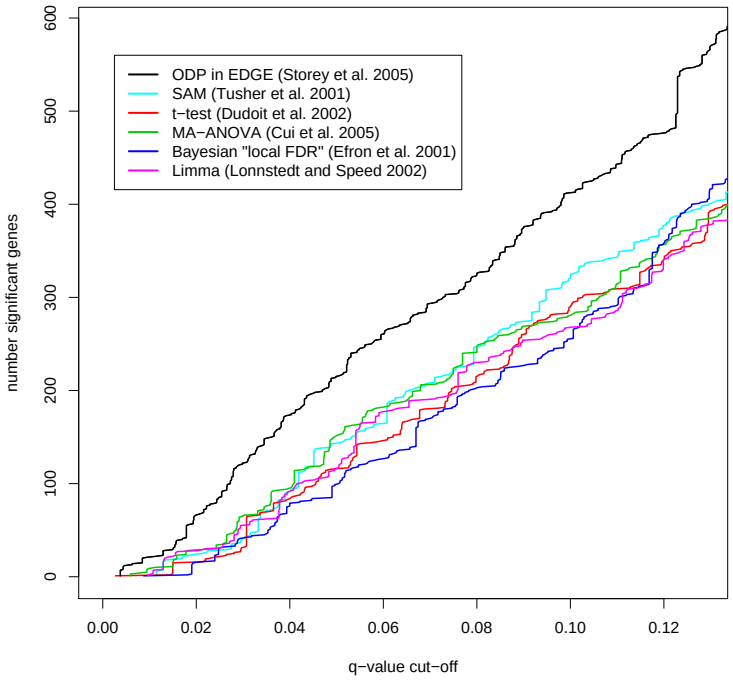


Figure 1: Comparison of EDGE to various other leading methods for identifying differential expressed genes in the Hedenfalk et al., 2001 study. Figure retrieved from Leek et al. (2005).

2 Citing this package

Storey, J. D. The optimal discovery procedure for large-scale significance testing, with applications to comparative microarray experiments. *Journal of the Royal Statistical Society, Series B 69* (2007), 347–368.

Storey, J. D., Dai, J., and Leek, J. T. The optimal discovery procedure for large-scale significance

testing, with applications to comparative microarray experiments. *Biostatistics 8* (2007), 414–432.

Woo, S., Leek, J. T., and Storey, J. D. A computationally efficient modular optimal discovery procedure. *Bioinformatics 27* (2011), 509–515.

3 Getting help

Hopefully most questions relating to the package will be answered in the vignette but to get a more detailed account of how to use the functions simply type within R:

```
help(package = "edge")
```

Please contact the authors directly with any issues regarding bugs. Otherwise, any questions or problems implementing `edge` will most efficiently be addressed on the Bioconductor mailing list, http://stat.ethz.ch/mailman/listinfo/bioconductor.

4 Quick start guide

To get started let's first load the `kidney` dataset that is included in the package:

```
library(edge)
data(kidney)
kidexpr <- kidney$kidexpr
age <- kidney$age
sex <- kidney$sex
```

The `kidney` study was interested in finding out what happens as the kidney ages. The covariate `age` is the age of the subject and the covariate `sex` is whether the subject was male or female. The expression values for the genes are contained in the `kidexpr` variable which is a 1500 by 72 matrix. The `kidney` dataset included in the package is a subset of the full dataset.

Once the data has been loaded, the user can proceed to use the function `edgeModel` to create the alternative and null hypothesis of the experiment:

```
edgeObj <- edgeModel(data = kidexpr, adj.var = data.frame(sex),
    tme = age, sampling = "timecourse", basis.type = "ncs",
    basis.df = 4)
```

Since the study examines the kidney tissue over time, the `sampling` method is "timecourse" where the `tme` argument is `age`. The `sex` variable is an adjustment variable and `basis.type` and `basis.df` describe the spline fit on the `tme` argument.

The alternative and null hypothesis formulated by `edgeModel` can be accessed by `fullModel` and `nullModel`, respectively:

```
fullModel(edgeObj)

## ~sex + ns(tme, df = 4, intercept = FALSE)
## <environment: 0x8e49320>
```

```
nullModel(edgeObj)

## ~sex
## <environment: 0x8e49320>
```

`adj.var` is the adjustment variable `sex` and time.basis is the spline fit of `age` created by `edgeModel`. The alternative and null models created by `edgeModel` are fitted to the data by least squares and test statistics are formed by using either `odp` or `lrt`:

```
# Optimal Discovery Procedure
edgeODP <- odp(edgeObj, verbose = FALSE)
# Likelihood Ratio Test
edgeLRT <- lrt(edgeObj)
```

The `summary` function can be used to summarize the objects:

```
summary(edgeODP)
```

The objects `edgeODP` and `edgeLRT` contain a `qvalue` object which provides p-values, q-values and local false discovery rate values:

```
qval <- qvalueObj(edgeODP)
summary(qval)

##
## Call:
## qvalue(p = pval)
##
## pi0: 0.4927
##
## Cumulative number of significant calls:
##
##           <1e-04 <0.001 <0.01 <0.025 <0.05 <0.1
## p-value       12     20    86    158   233  392
## q-value        0      6    15     17    44   98
## local FDR      0      0     9     13    17   58
##             <1
## p-value   1500
## q-value   1500
## local FDR 1500
```

The following sections of the manual go through various case studies to show additional features of the `edge` package.

5 Examples

Three examples will be used to show the functionality of `edge`. The examples will cover static, longitudinal and independent case studies. It will become evident that in each case, the analysis procedure is similar and the only step that differs is the model setup.

There are three main steps when using `edge`:

- Load experimental data. Optionally, create an `ExpressionSet` object.
- Use `edgeModel` to create an `edgeSet` object. If an `ExpressionSet` object is created use the `edgeSet` function.
- Use functions `odp` or `lrt` to obtain the q-value object which is the slot of interest. The `edgeFit` function can be used to extract information regarding the model fits.

5.1 Static study

Step 1: The gibson dataset provides gene expression measurements in peripheral blood leukocyte samples from three Moroccan Amazigh groups leading distinct ways of life: desert nomadic (DESERT), mountain agrarian (VILLAGE), and coastal urban (AGADIR). Suppose we are interested in finding the genes that differentiate the Moroccan Amazigh groups the most.

To import the data:

```
data(gibson)
names(gibson)

## [1] "batch"    "gibexpr"  "gender"   "location"
```

There are a few variables in this data set: `batch`, `gibexpr`, `gender`, and `location`. The three covariates of interest are `gender`, `batch` and `location`. There are three locations where individuals were sampled (`location`): "VILLAGE", "DESERT" and "AGADIR". At each location there were either "males" or "females" (`gender`) and there were different `batches`. The `gibexpr` variable contains the expression matrix of the experiment.

Step 2: Use the function `edgeModel` to create an `edgeSet` object:

```
edgeObj <- edgeModel(data = gibson$gibexpr, adj.var = data.frame(gibson$gender,
    gibson$batch), grp = gibson$location, sampling = "static")
```

The `gibson` study is a static experiment so the `sampling` argument will be "static". The `grp` argument is for the `location` variable and the `adj.var` argument is the adjustment variables. The `adj.var` must be in `model.matrix` form. A brief overview of the arguments of `edgeModel`:

- `data` Matrix of expression values
- `tme` A vector of time measurements for a time-course study
- `ind` A factor that assigns each observations to an individual in the experiment
- `basis.df` Degree of freedom of spline fit in a time-course study
- `basis.type` A spline curve is fitted to the `tme` variable in a "timecourse" study. The type can be "ncs" (B-spline for a natural cubic spline) or "ps" (polynomial spline)
- `bio.var` Biological variables
- `adj.var` Adjustment variables (matrix)
- `grp` Numerical vector describing which group each observation belong (i.e "DESERT", "VILLAGE"

or "AGADIR")

- `sampling` Can either be "timecourse" or "static" depending on the experiment

Other examples in the vignette will show when to use the additional arguments in `edgeModel`. `edgeSet` is the main object in the package:

```
slotNames(edgeObj)
```

```
##  [1] "null.model"        "full.model"
##  [3] "null.matrix"       "full.matrix"
##  [5] "individual"        "qvalueObj"
##  [7] "experimentData"    "assayData"
##  [9] "phenoData"         "featureData"
## [11] "annotation"        "protocolData"
## [13] ".__classVersion__"
```

The `edgeObj` is an `edgeSet` object that extends an `ExpressionSet` object. It contains the covariates and expression values along with the alternative and null models. The alternative and null models generated by `edgeModel` can be accessed using

```
fullModel(edgeObj)
```

```
## ~gibson.gender + gibson.batch + grp
## <environment: 0x91e8260>
```

```
nullModel(edgeObj)
```

```
## ~gibson.gender + gibson.batch
## <environment: 0x91e8260>
```

The key slot in `edgeObj` is the `qvalueObj` which should be the only empty slot. The other slots are directly related to the input data and hypothesis models. See the section 6.1 for more details on the `edgeSet` object.

The `summary` function summarizes the slots in the `edgeSet` object:

```
summary(edgeObj)
```

```
##
## ExpressionSet Summary
##
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 2000 features, 46 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: 1 2 ... 46 (46 total)
##   varLabels: gibson.gender gibson.batch
##     grp
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation:
```

```
##
## edge Analysis Summary
##
## Total number of arrays: 46
## Total number of probes: 2000
##
## Biological variables:
##  Null Model:~gibson.gender + gibson.batch
## <environment: 0x91e8260>
##
##  Full Model:~gibson.gender + gibson.batch + grp
## <environment: 0x91e8260>
##
## .......
##
```

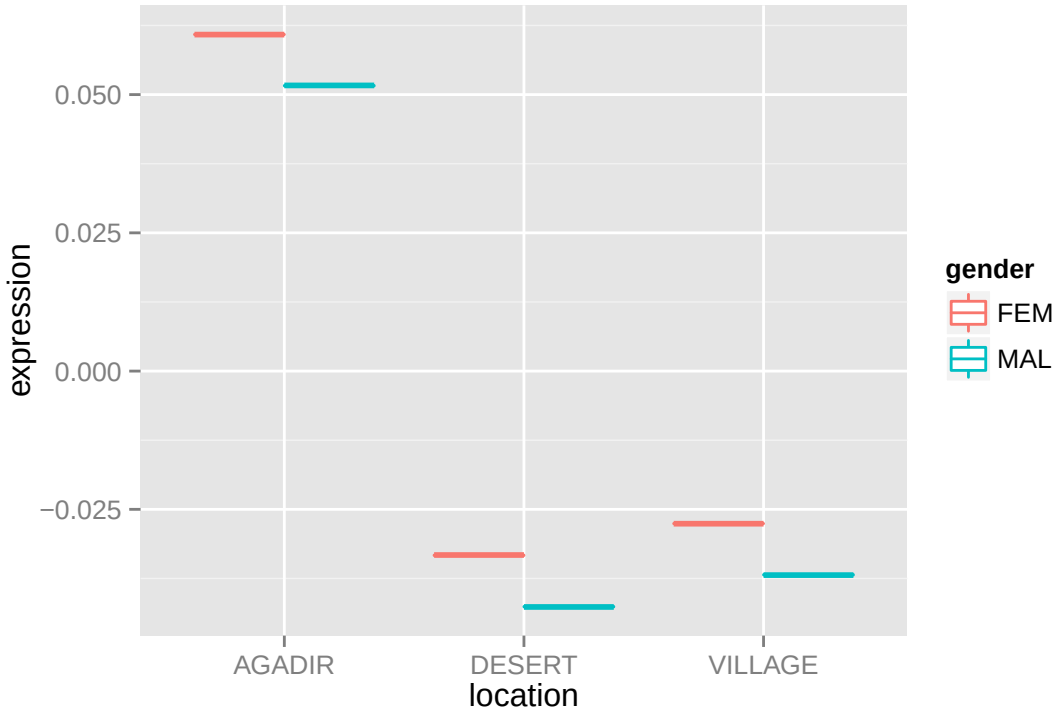
Step 3: Before running any significance analysis, lets view our model fits of the data. The `edgeFit` function can be used to extract residuals and fitted values from the alternative and null models:

```
efObj <- edgeFit(edgeObj, stat.type = "odp")
```

`edgeFit` is simply an implementation of least squares using the alternative and null models. The `stat.type` argument specifies whether you want the `odp` or `lrt` fitted values. The difference between choosing "odp" and "lrt" is that "odp" centers the data by the null model fit. To access the alternative model fitted values:

```
fitVals <- fitFull(efObj)
```

The fitted values of the first gene are shown below:



The user can either use the function `odp` or `lrt` to get the `qvalue` object. The `lrt` function performs a likelihood ratio test to determine p-values. If the null distribution, `nullDistn`, is calculated using "bootstrap" then residuals from the alternative model are re-sampled and added to the null model to simulate a distribution where there is no differential expression. Otherwise, the default input is "normal" and the assumption is that the data set follows an F-distribution.

The `odp` function uses information across all tests when formulating the test statistic. In order to improve the speed of the algorithm, we utilize a k-means clustering algorithm where genes are assigned to a cluster based on the Kullback-Leiber distance. Each gene is assigned a module-average parameter to calculate the odp-statistic. The number of clusters can be adjusted by `n.mods`. Type `?odp` for more details on the algorithm.

To use `odp` or `lrt` on the `edgeSet` object:

```
edgeLRT <- lrt(edgeObj, nullDistn = "normal")
edgeODP <- odp(edgeObj, bs.its = 10, verbose = FALSE,
    n.mods = 20)
```

The argument `bs.its` controls the number of bootstrap iterations, `verbose` prints the iteration step and `n.mods` is the number of clusters formed. If `n.mods` is equal to the number of genes than the full Optimal Discovery Procedures is used.

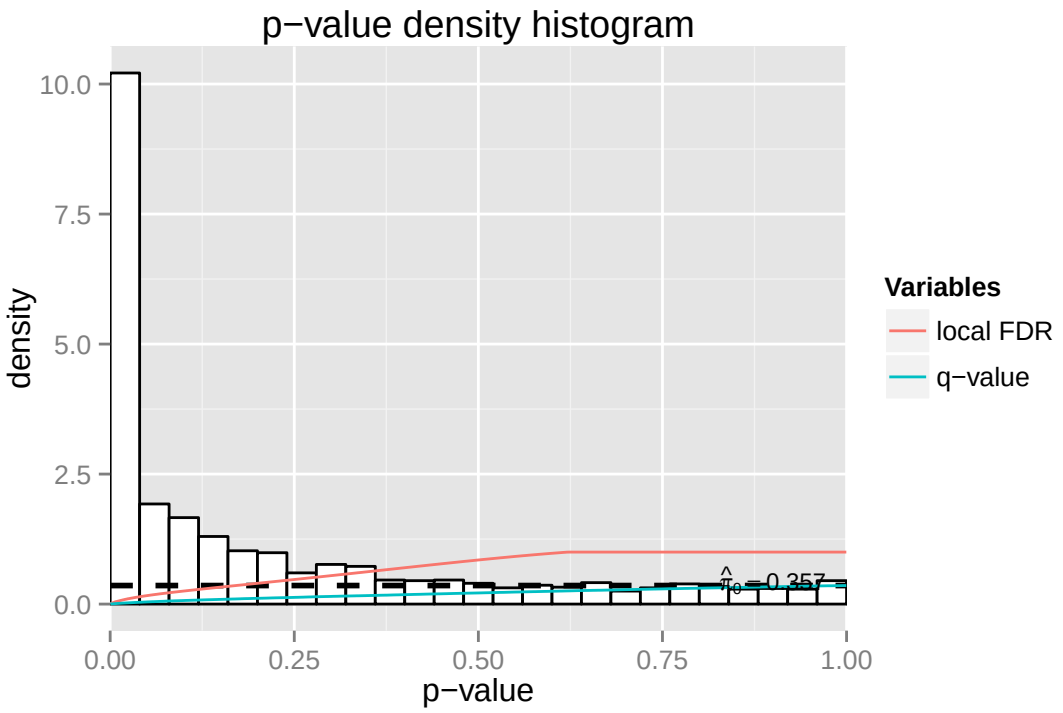
The slot of interest for significance analysis is the `qvalueObj` slot. To access the slot:

```
qval <- qvalueObj(edgeODP)
summary(qval)

##
```

```
## Call:
## qvalue(p = pval)
##
## pi0: 0.3566
##
## Cumulative number of significant calls:
##
##           <1e-04 <0.001 <0.01 <0.025 <0.05 <0.1
## p-value      267    368   607    733   853 1039
## q-value        0    333   579    746   927 1232
## local FDR      0    270   373    464   593  725
##             <1
## p-value   2000
## q-value   2000
## local FDR 1748
```

To visualize the results, `plot` or `hist` functions can be used on `qval`:

```
hist(qval)
```

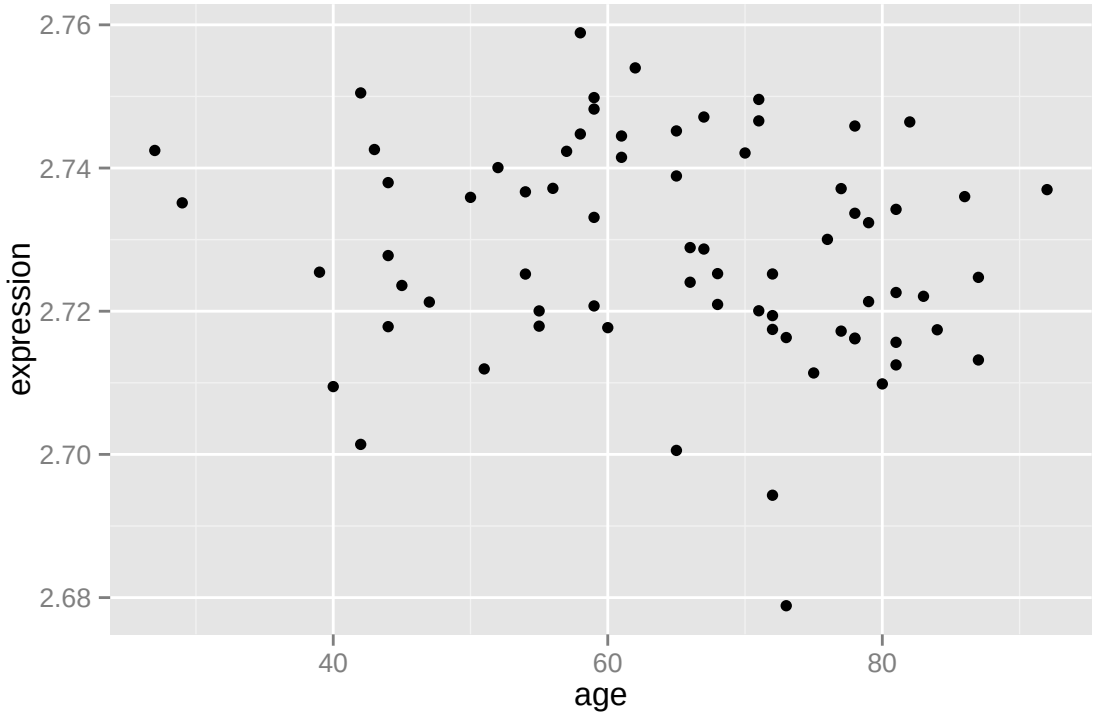


5.2 Independent time course study

Step 1: Gene expression measurements from kidney samples were obtained from 72 human subjects ranging in age from 27 to 92 years. Only one array was obtained per sample and the age and tissue type of each subject was recorded. To import the data:

```
data(kidney)
names(kidney)

## [1] "age"     "sex"     "kidexpr"

kidexpr <- kidney$kidexpr
age <- kidney$age
sex <- kidney$sex
```

There are a few covariates in this data set: `sex`, `age`, `tissue`, `kidexpr` and `kidcov`. The two main covariates of interest for this example are the `sex` and `age` covariates. The `sex` variable is whether the subject was male or female and the `age` variable is the age of the patients. Lets view the first gene to get a better idea of data:



For this particular gene, it seems that there is a slight decrease in expression past 60 and an increase past 80.

Step 2: Use the function `edgeModel` to create an `edgeSet` object:

```
edgeObj <- edgeModel(data = kidexpr, adj.var = data.frame(sex),
    tme = age, sampling = "timecourse", basis.type = "ncs",
    basis.df = 4)
```

Since the `kidney` study is a time-course study the sampling method will be "timecourse". The adjustment variable in the study is `sex` while the time variable is `age`. A brief overview of the arguments can be found in section 5.1.

The alternative and null models can be accessed using

```
fullModel(edgeObj)

## ~sex + ns(tme, df = 4, intercept = FALSE)
## <environment: 0xb344b98>

nullModel(edgeObj)

## ~sex
## <environment: 0xb344b98>
```

The `adj.var` corresponds to the adjustment variables and `time.basis` corresponds to the time variable inputed in `edgeModel`. See section 6.1 for more details.

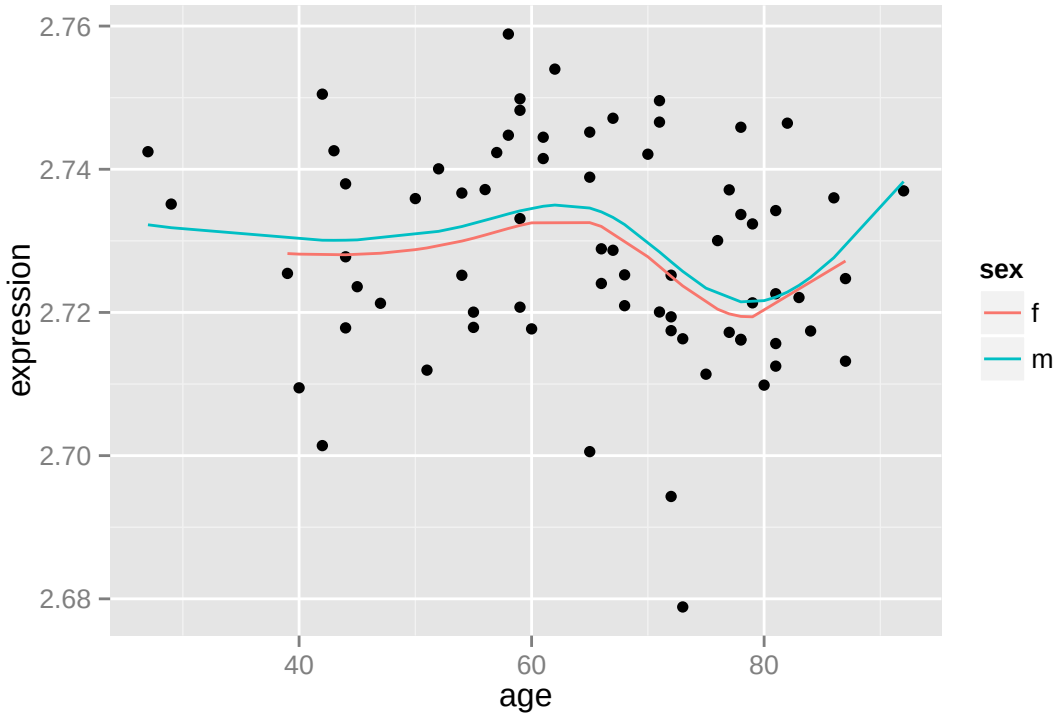
Step 3: Before running any significance analysis, lets view our model fits of the data. The `edgeFit` function can be used to extract residuals and fitted values from the alternative and null models. `edgeFit` performs a least squares regression on the alternative and null models. In the `endotoxin` dataset since there is a `ind` factor, the data is centered by individual in order to remove the effects from various individuals:

```
efObj <- edgeFit(edgeObj, stat.type = "lrt")
```

The `stat.type` argument specifies whether you want the `odp` or `lrt` fitted values. As mentioned in section 5.1, the only difference between the fitted values of "odp" and "lrt" is that the "odp" method centers the data by the null model fit. To access the alternative model fitted values:

```
fitVals <- fitFull(efObj)
```

The fitted values for the gene shown in the first step is shown below:



Our initial intuition was correct: A slight decrease in expression follow by an increase as time goes on. As mentioned in section 5.1, the `lrt` or `odp` function can be used for differential analysis:

To use `odp` or `lrt` on the `edgeSet` object:

```
edgeLRT <- lrt(edgeObj, nullDistn = "normal")
edgeODP <- odp(edgeObj, bs.its = 50, verbose = FALSE,
    n.mods = 100)
```

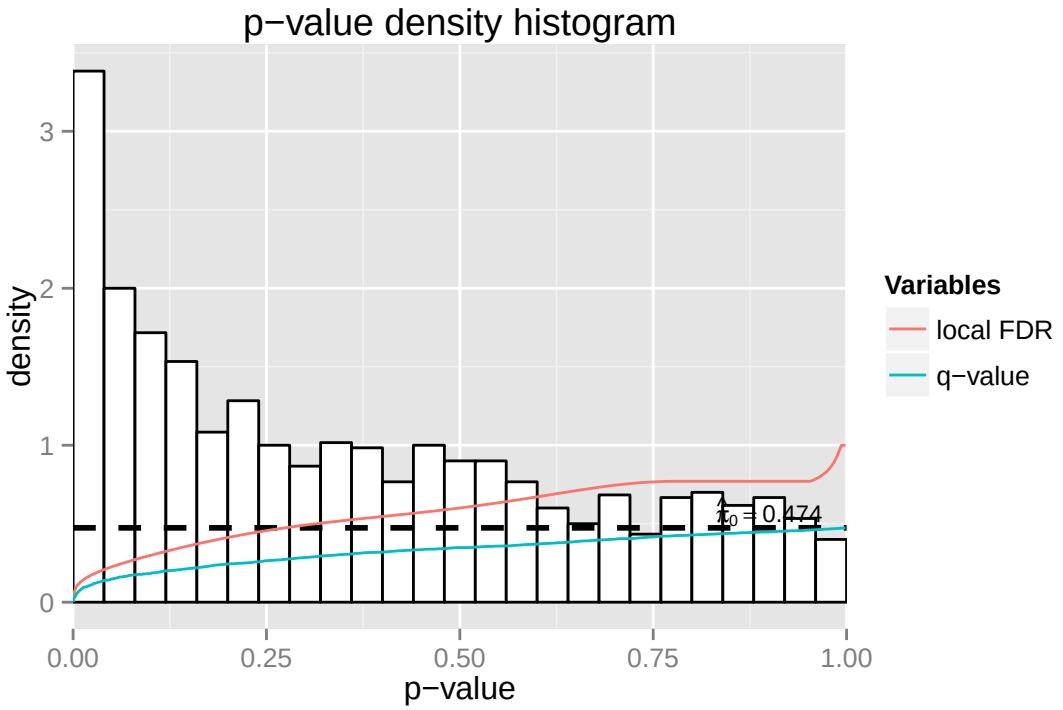
Next, lets see if the gene mentioned above is significant:

```
qval <- qvalueObj(edgeODP)
qval$pvalues[1]

## [1] 0.06131

qval$qvalue[1]

## [1] 0.1609

qval$lfdr[1]

## [1] 0.2436
```

If we assume a p-value threshold of 0.5 then the gene does not show any significant changes over time. The high q-value and local FDR value confirm that the gene is not significant.

Using the `hist` function on the `qvalue` object:

```
hist(qval)
```



5.3 Longitudinal time course study

Step 1: The endotoxin dataset provide gene expression measurements in an endotoxin study where four subjects were given endotoxin and four subjects were given a placebo. Blood samples were collected and leukocytes were isolated from the samples before infusion and measurement were recorded at times 2, 4, 6, 9, 24 hours. We are interested in identifying genes that vary over time between the endotoxin and control groups.

To import the data:

```
data(endotoxin)
names(endotoxin)

## [1] "class"    "endoexpr" "ind"      "time"
```

There are a few covariates in this data set: `expr`, `class`, `individual`, and `time`. There are 8 individuals in the experiment (`ind`) that were sampled at multiple time points (`time`) that were either "endotoxin" or "control" (`class`). The `expr` variable contains the expression values of the experiment.

Step 2: Use the function `edgeModel` to create an `edgeSet` object:

```
edgeObj <- edgeModel(data = endotoxin$endoexpr, ind = endotoxin$ind,
    tme = endotoxin$time, grp = data.frame(endotoxin$class),
```

```
    sampling = "timecourse", basis.type = "ncs",
    basis.df = 4)
```

The `endotoxin` experiment is a time-course study so the `sampling` argument will be "timecourse". The `tme` argument is for the time variable in the experiment and the `ind` argument is to identify which observations corresponds what individuals. Since the `sampling` method is "timecourse", we fit a spline curve described by variables `basis.type` and `basis.df`. Additional arguments can be viewed by typing `?edgeModel`.

The alternative and null models can be accessed using

```
fullModel(edgeObj)

## ~endotoxin.class + ns(tme, df = 4, intercept = FALSE) + (endotoxin.class):ns(tme,
##     df = 4, intercept = FALSE)
## <environment: 0xbaafbf0>

nullModel(edgeObj)

## ~endotoxin.class + ns(tme, df = 4, intercept = FALSE)
## <environment: 0xbaafbf0>
```

The `adj.var` corresponds to the adjustment variables, the `time.basis` corresponds to the time variable and the `grp` corresponds to the treatment variable inputed in `edgeModel`. See the section 5.1 object for more details on accessing and setting `edgeSet` slots.

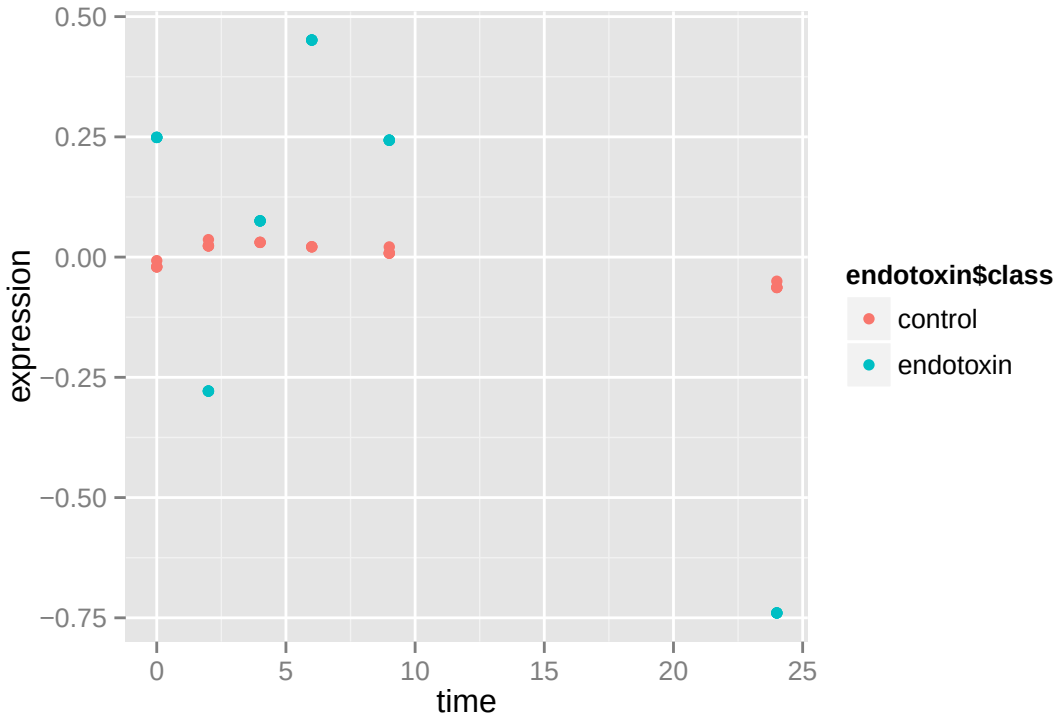
Step 3: Before running any significance analysis, lets view our model fits of the data. The `edgeFit` function can be used to extract residuals and fitted values from the alternative and null models:

```
efObj <- edgeFit(edgeObj, stat.type = "lrt")
```

The `stat.type` argument specifies whether you want the `odp` or `lrt` fitted values. To access the alternative model fitted values:

```
fitVals <- fitFull(efObj)
```

The fitted values of the first gene are shown below:



Next, the user can either use the function `odp` or `lrt` to get the `qvalue` object. See section 5.1 for more details on the `odp` and `lrt` method. To use `odp` or `lrt`:

```
edgeLRT <- lrt(edgeObj, nullDistn = "normal")
edgeODP <- odp(edgeObj, bs.its = 10, verbose = FALSE,
    n.mods = 20)
```

The slot of interest for significance analysis is the `qvalueObj` slot. To access the slot:

```
qval <- qvalueObj(edgeODP)
```

To summarize the object using the `odp` method:

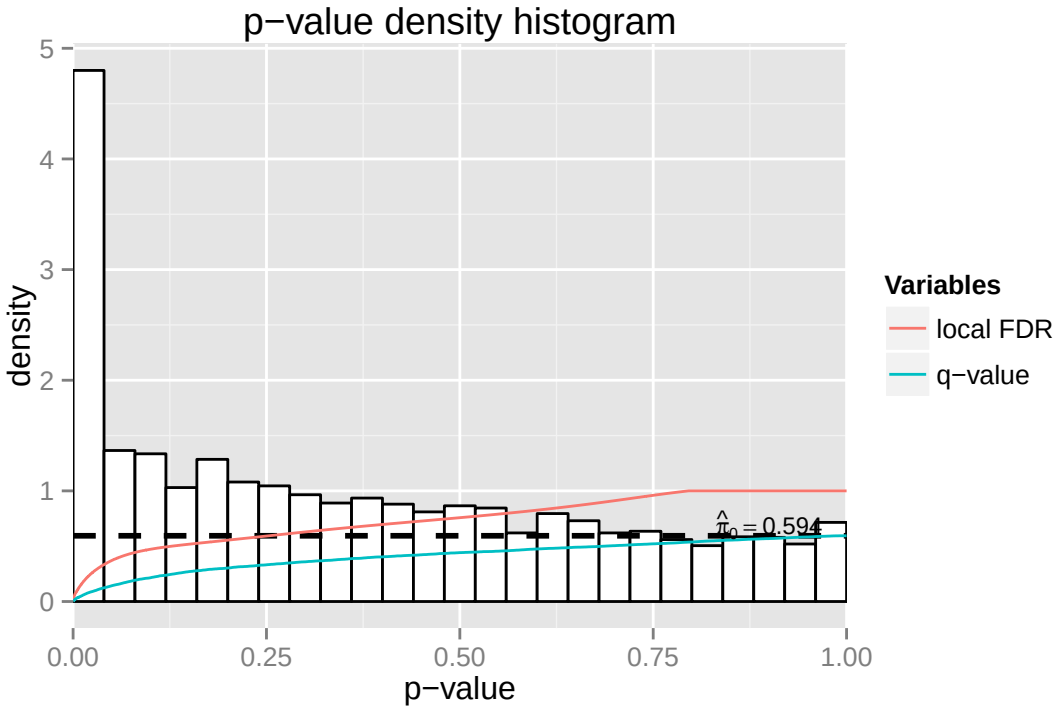
```
summary(qval)
```

```
##
## Call:
## qvalue(p = pval)
##
## pi0: 0.5942
##
## Cumulative number of significant calls:
##
##           <1e-04 <0.001 <0.01 <0.025 <0.05 <0.1
## p-value      196    313   614    820  1039 1381
## q-value        0    185   315    424   623  866
## local FDR      0    134   228    274   322  485
##             <1
```

```
## p-value   4999
## q-value   5000
## local FDR 4402
```

Using the `hist` function on the `qvalue` object:

```
hist(qval)
```



6 Objects in edge

6.1 edgeSet

The main object in `edge` is the `edgeSet` object and it contains the following slots:

- `full.model`: the alternative model of the experiment
- `null.model`: the null model of the experiment
- `full.matrix`: the alternative model in matrix form
- `null.matrix`: the null model in matrix form
- `individual`: containing information on individuals in the experiment
- `qvalueObj`: `qvalue` list

- `ExpressionSet`: inherits the slots from `ExpressionSet` object

`ExpressionSet` contains the expression measurements and the covariates of the experiment. To access the expression values, one can use the function `exprs` or to access the covariates, `pData`. The `ExpressionSet` class is a widely used object in Bioconductor and more information can be found http://www.bioconductor.org/packages/2.14/bioc/html/Biobase.html.

As an example of how to access the slots of an `edgeObj` lets say we are interested in viewing the alternative model. The model can be accessed by:

```
fullModel(edgeObj)

## ~endotoxin.class + ns(tme, df = 4, intercept = FALSE) + (endotoxin.class):ns(tme,
##     df = 4, intercept = FALSE)
## <environment: 0xbaafbf0>
```

To get a summary of the object and all the slots use the `summary` function:

```
summary(edgeObj)

##
## ExpressionSet Summary
##
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 5000 features, 46 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: 1 2 ... 46 (46 total)
##   varLabels: endotoxin.class tme
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation:
##
## edge Analysis Summary
##
## Total number of arrays: 46
## Total number of probes: 5000
##
## Biological variables:
##  Null Model:~endotoxin.class + ns(tme, df = 4, intercept = FALSE)
## <environment: 0xbaafbf0>
##
##  Full Model:~endotoxin.class + ns(tme, df = 4, intercept = FALSE) + (endotoxin.class):ns(tme,
##     df = 4, intercept = FALSE)
## <environment: 0xbaafbf0>
##
## Individuals:
##  [1] 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 4 4 4 4
## [23] 4 4 5 5 5 5 5 5 6 6 6 6 7 7 7 7 7 7 8 8 8 8
## [45] 8 8
## Levels: 1 2 3 4 5 6 7 8
```

```
##
## .......
##
```

The package offers great flexibility in model choice and experimentation. See `?edgeSet` for more details on how to extract and set each slot in the object.

6.2 edgeFit

The function `edgeFit` fits a linear model to each gene and returns that information in an `edgeFit` object that contains the following slots:

- `fit.full`: fitted values from the alternative model
- `fit.null`: fitted values from null model
- `res.full`: residuals from the alternative model
- `res.null`: residuals from the null model
- `dH.full`: diagonal elements in the projection matrix for the full model
- `beta.coef`: the coefficients for the full model
- `stat.type`: statistic type used, either "odp" or "lrt"

To access the coefficients of the `edgeFit` object in section 5.2:

```
betaCoef(efObj)
```

Similarly, to access the full and null residuals:

```
resFull(efObj)
resNull(efObj)
```

A summary of the object can be displayed by:

```
summary(efObj)
```

See `?edgeFit` for more details on how to extract and set each slot of the object.

7 Using the sva package

The `sva` package is useful for removing batch effects or any unwanted variation in an experiment. It does this by forming surrogate variables to adjust for sources of unknown variation. `edge` uses the sva package in the function `edgeSVA`. An example of how to use this on the `kidney` dataset:

```
newEdgeObj <- edgeSVA(edgeObj, n.sv = 5, B = 10)

## Number of significant surrogate variables is:  5
```

```
## Iteration (out of 10 ):1  2  3  4  5  6  7  8  9  10
```

A new `edgeObj` is created that includes the surrogate variables in the null and full matrices from sva. See `?sva` for additional input parameters in `edgeSVA`.

Now `odp` or `lrt` can simply be used as before:

```
edgeODP <- odp(newEdgeObj, verbose = FALSE)
edgeLRT <- lrt(newEdgeObj)
```

Acknowledgements

This software development has been supported in part by funding from the National Institutes of Health and the Office of Naval Research.

References

[1] Storey, J. D. The optimal discovery procedure for large-scale significance testing, with applications to comparative microarray experiments. *Journal of the Royal Statistical Society, Series B 69* (2007), 347–368.

[2] Storey, J. D., Dai, J., and Leek, J. T. The optimal discovery procedure for large-scale significance testing, with applications to comparative microarray experiments. *Biostatistics 8* (2007), 414–432.

[3] Woo, S., Leek, J. T., and Storey, J. D. A computationally efficient modular optimal discovery procedure. *Bioinformatics 27* (2011), 509–515.