Bioconductor's edge package
Version 0.99.0

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

July 8, 2014

Contents

1 Introduction **2**

2 Citing this package **2**

3 Getting help **3**

4 Quick start guide **3**

5 Objects in edge **4**
5.1 edgeSet 4
5.2 edgeFit 4

6 Examples **5**
6.1 Independent time course study 5
6.2 Longitudinal time course study 14
6.3 Static study 22

7 Using the sva package **30**

8 Advanced topic: Using the ExpressionSet object **30**

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). There are already a number of existing software packages that perform significance analysis but `edge` can use 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`.

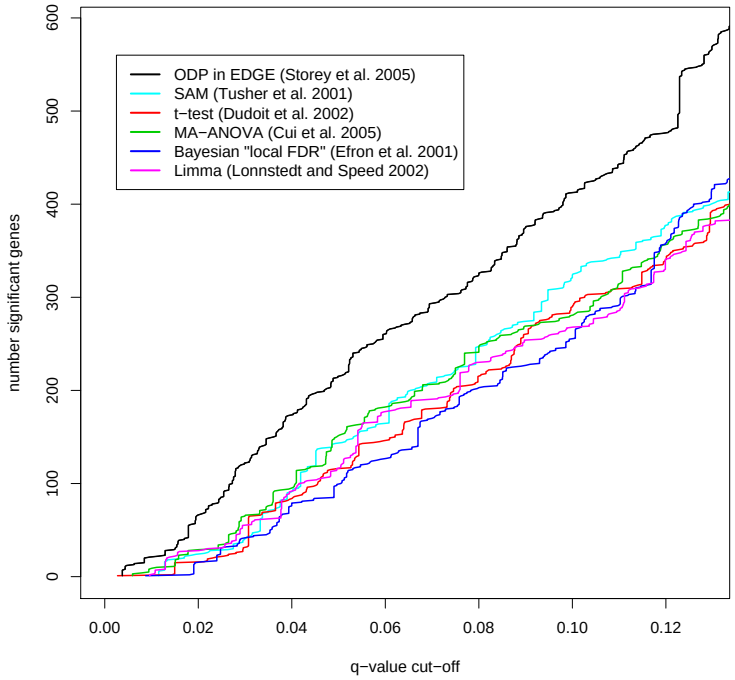


Figure 1: Comparison of EDGE to various other leading methods for identifying differential expressed genes in the Hedenfalk et al., 2001 study. Figure retrieved from Leek et al. (2005).

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

Load all relevant data and use `edgeModel` to create an `edgeSet` object:

```
library(edge)
data(kidney)
sex <- kidney$sex[kidney$tissue == "c"]
age <- kidney$age[kidney$tissue == "c"]
kidexpr <- log(kidney$kidexpr[, kidney$tissue ==
    "c"] + 10)
edgeObj <- edgeModel(data = kidexpr, adj.var = model.matrix(~sex),
    tme = age, sampling = "timecourse", basis.type = "ncs",
    basis.df = 4)
```

The alternative and null hypothesis are formulated by `edgeModel`. The `odp` or `lrt` function can be used as follows:

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

The object `edgeODP` and `edgeLRT` contain a `qvalue` object which is the main slot of interest that can accessed by the `qvalue.obj` function. The following sections of the manual go through various case studies to show additional features of the `edge` package.

5 Objects in edge

5.1 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
- `qvalue.obj`: `qvalue` list
- `ExpressionSet`: inherits the slots from `ExpressionSet` object

As an example of how to access and set the slots of an `edgeObj` lets say we are interested in changing the alternative model to just include the variable `cov`. The current models can be accessed by:

```
nullModel(object)
fullModel(object)
```

A new alternative model can be set by

```
fullModel(object) <- ~1 + cov
```

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

```
summary(object)
```

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.

5.2 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 in the `edgeFit` object:

```
betaCoef(object)
```

Similarly, to access the full and null residuals:

```
resFull(object)
resNull(object)
```

A summary of the object can be displayed by:

```
summary(object)
```

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

6 Examples

Three examples will be used to show the functionality of `edge`. In each example there will be a static, longitudinal and independent case study. 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.

6.1 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:

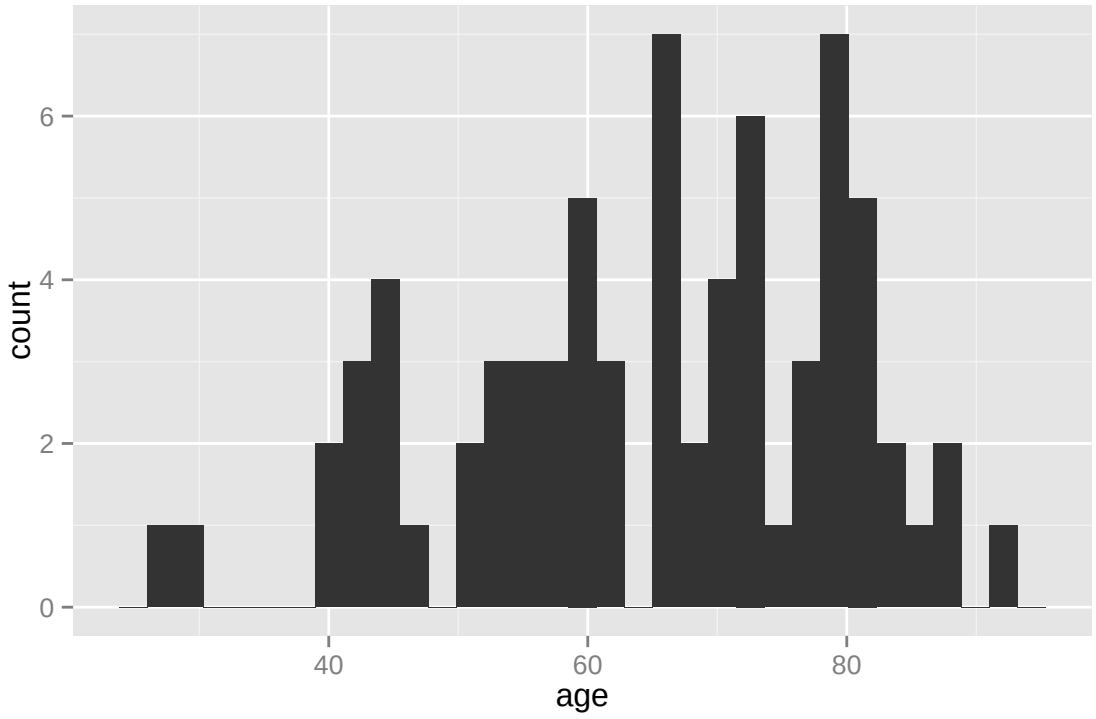
```
data(kidney)
names(kidney)

## [1] "age"     "tissue"  "sex"     "kidexpr"
## [5] "kidcov"
```

There are a few covariates in this data set: `sex`, `age`, `tissue`, `kidexpr` and `kidcov`. We will focus on the expression values of the cortex `tissue` samples:

```
sex <- kidney$sex[kidney$tissue == "c"]
age <- kidney$age[kidney$tissue == "c"]
kidexpr <- log(kidney$kidexpr[, kidney$tissue ==
    "c"] + 10)
```

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 a histogram of the `age` to get a better idea of data:



From the histogram, it is important to keep in mind that a majority of the patients appear to be older than 50.

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

```
edgeObj <- edgeModel(data = kidexpr, adj.var = model.matrix(~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 of `edgeModel`

- `data` Matrix of expression values
- `adj.var` Adjustment variables (matrix)
- `tme` A vector containing the time variable in a time course study

- `sampling` Can either be "timecourse" or "static" depending on the experiment
- `basis.df` The degree of freedom for the spline basis
- `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 "poly" (B-spline for a polynomial spline)

Additional arguments can be viewed by typing `?edgeModel`. `edgeSet` is the main object in the package and the slots can be viewed by:

```
slotNames(edgeObj)

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

The alternative and null models are automatically generated by `edgeModel`. The alternative and null models can be accessed using

```
fullModel(edgeObj)

## ~-1 + adj.var + time.basis
## <environment: 0x99f0ab8>

nullModel(edgeObj)

## ~-1 + adj.var
## <environment: 0x99f0ab8>
```

The `adj.var` corresponds to the adjustment variables and `time.basis` corresponds to the time variable inputed in `edgeModel`. The key slot in `edgeObj` is the `qvalue.obj` which should be the only empty slot. The other slots are directly related to the input data and hypothesis models. See the section on the `edgeSet` object for more details on these slots.

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

```
summary(edgeObj)

##
## ExpressionSet Summary
##
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 1500 features, 72 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: 1 2 ... 72 (72 total)
##   varLabels: X.Intercept. sexm ... X4 (6
##     total)
##   varMetadata: labelDescription
```

```
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation:
##
## edge Analysis Summary
##
## Total number of arrays: 72
## Total number of probes: 1500
##
## Biological variables:
##  Null Model:~-1 + adj.var
## <environment: 0x99f0ab8>
##
##  Full Model:~-1 + adj.var + time.basis
## <environment: 0x99f0ab8>
##
## .......
##
```

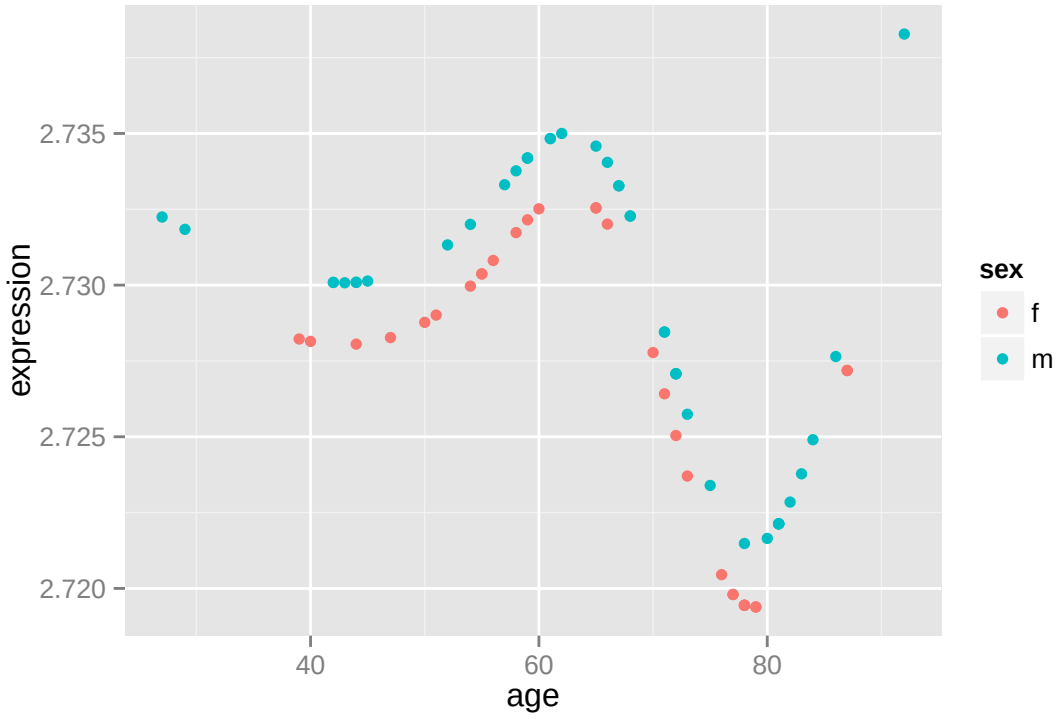
Step 3: Before running any significance analysis, lets view our model fits of the data. The `edgeFit` function can be used to extract residuals and fitted values from the alternative and null models:

```
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:



We can see that with our alternative model chosen, the expression goes up and down as the kidney ages for this particular gene. Once other interesting genes have been inspected, 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.

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

```
edgeLRT <- lrt(edgeObj, nullDistn = "normal")
```

To view a summary of the object:

```
summary(edgeLRT)

##
## ExpressionSet Summary
##
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 1500 features, 72 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: 1 2 ... 72 (72 total)
##   varLabels: X.Intercept. sexm ... X4 (6
##     total)
```

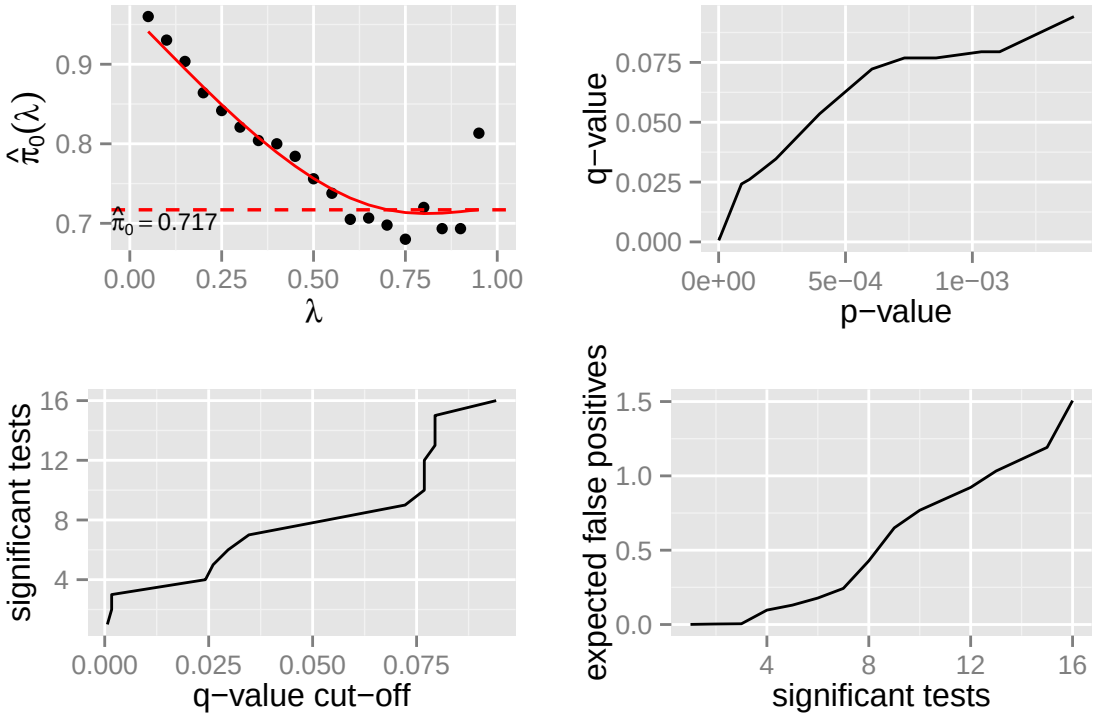
```
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation:
##
## edge Analysis Summary
##
## Total number of arrays: 72
## Total number of probes: 1500
##
## Biological variables:
##  Null Model:~-1 + adj.var
## <environment: 0x99f0ab8>
##
##  Full Model:~-1 + adj.var + time.basis
## <environment: 0x99f0ab8>
##
## .......
##
##
## Statistical significance summary:
## pi0: 0.7172
##
## Cumulative number of significant calls:
##
##           <1e-04 <0.001 <0.01 <0.025 <0.05 <0.1
## p-value        4     12    39     72   132  244
## q-value        0      1     3      4     7   16
## local fdr      0      0     3      3     3    7
##             <1
## p-value   1500
## q-value   1500
## local fdr 1253
```

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

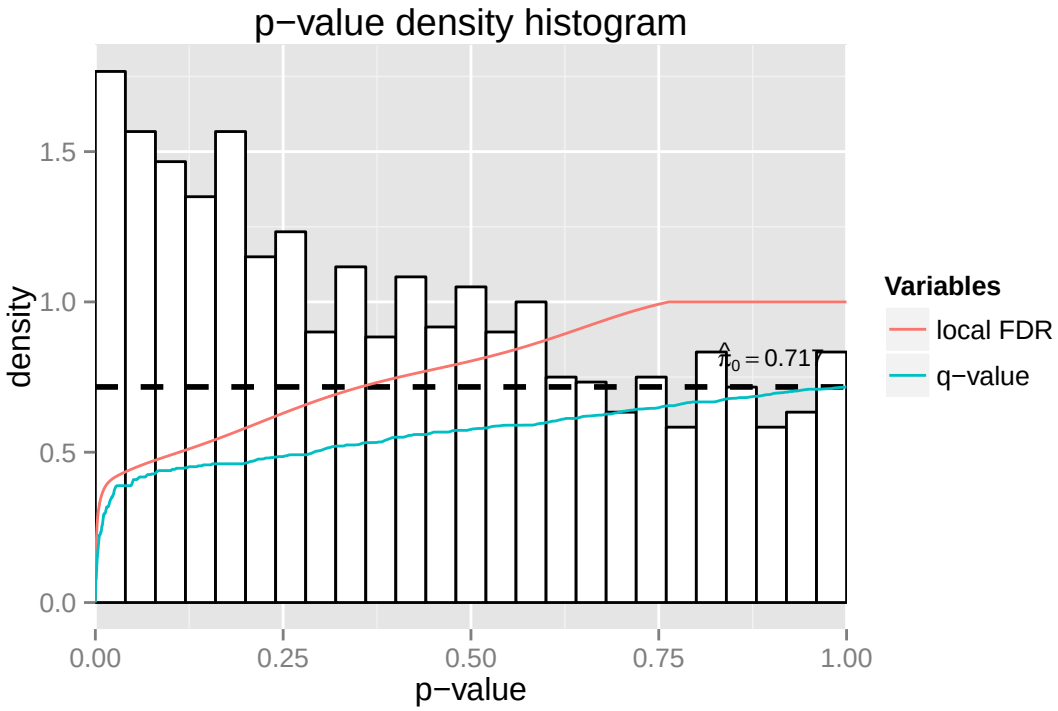
```
qval <- qvalue.obj(edgeLRT)
```

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

```
plot(qval)
```


$\hat{\pi}_0(\lambda)$
$\hat{\pi}_0 = 0.717$
λ
q−value
p−value
significant tests
q−value cut−off
expected false positives
significant tests

```
hist(qval)
```


p−value density histogram
density
$\hat{\pi}_0 = 0.717$
p−value
Variables
local FDR
q−value

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` on an `edgeSet object`:

```
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.

To summarize the object using the `odp` method:

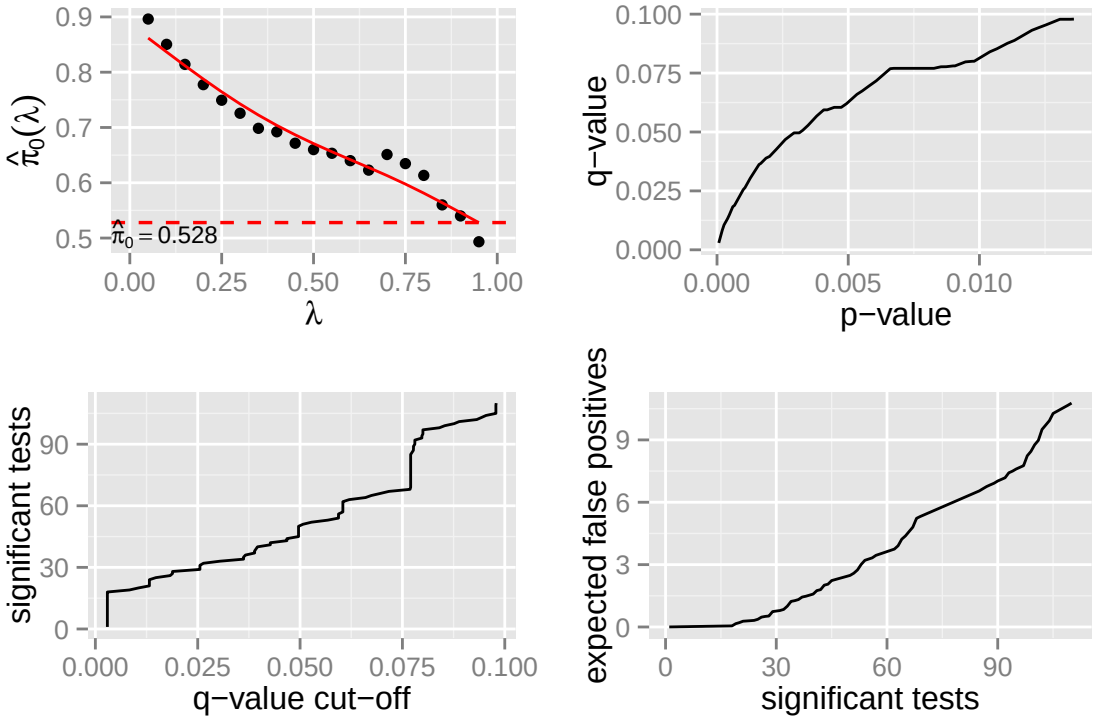
```
summary(edgeODP)
```

```
##
## ExpressionSet Summary
##
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 1500 features, 72 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: 1 2 ... 72 (72 total)
##   varLabels: X.Intercept. sexm ... X4 (6
##     total)
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation:
##
## edge Analysis Summary
##
## Total number of arrays: 72
## Total number of probes: 1500
##
## Biological variables:
##  Null Model:~-1 + adj.var
## <environment: 0x99f0ab8>
##
##  Full Model:~-1 + adj.var + time.basis
## <environment: 0x99f0ab8>
##
## .......
##
##
## Statistical significance summary:
## pi0: 0.5279
##
```

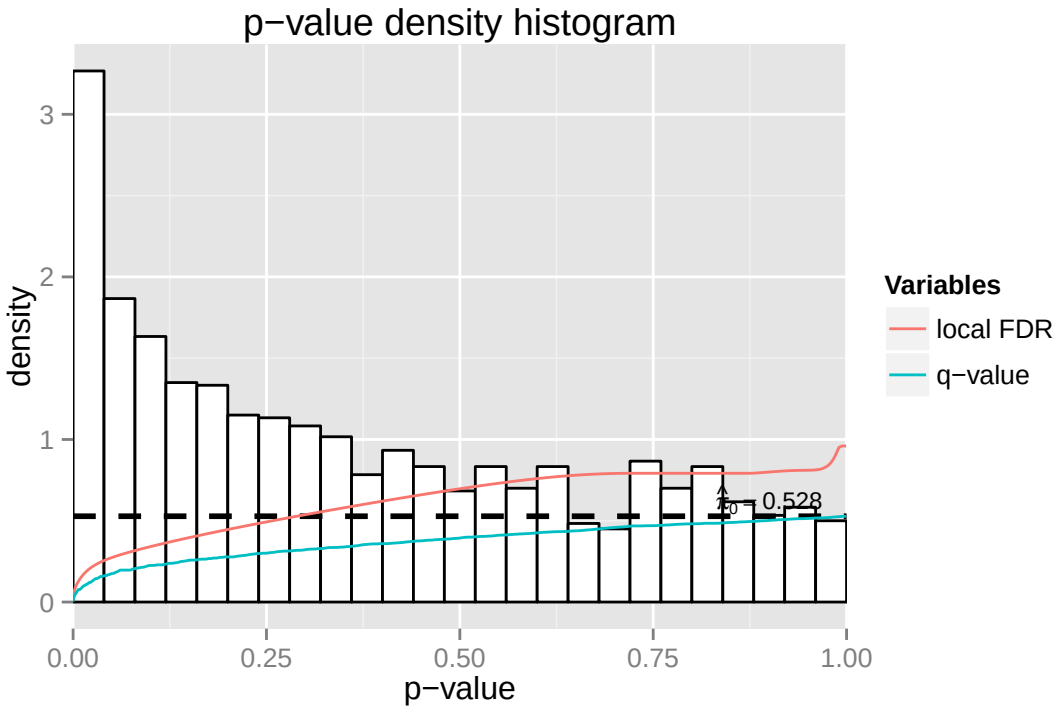
```
## Cumulative number of significant calls:
##
##           <1e-04 <0.001 <0.01 <0.025 <0.05 <0.1
## p-value       18     28    97    152   223  352
## q-value        0      0    19     28    50  110
## local fdr      0      0    18     19    28   53
##              <1
## p-value    1500
## q-value    1500
## local fdr 1500
```

Using `plot` and `hist` functions on the `qvalue` object:

```
qval <- qvalue.obj(edgeODP)
plot(qval)
```




```
hist(qval)
```



We notice that the optimal discovery method finds more significant genes. As shown in previous research by Storey [1], the ODP finds more significant genes for a fixed FDR when compared to the likelihood ratio test and other popular statistical methods.

6.2 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$time, grp = endotoxin$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`. A brief overview of the arguments of `edgeModel`

- `data` Matrix of expression values
- `adj.var` Adjustment variables (matrix)
- `ind` Vector describing what observations belong to which individuals in the experiment
- `grp` Numerical vector describing which group each observation belong (i.e "control" or "endotoxin")
- `tme` A vector containing the time variable in a time course study
- `sampling` Can either be "timecourse" or "static" depending on the experiment
- `basis.df` The degree of freedom for the spline basis
- `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 "poly" (B-spline for a polynomial spline)

Additional arguments can be viewed by typing `?edgeModel`. `edgeSet` is the main object in the package and the slots can be viewed by:

```
slotNames(edgeObj)
```

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

The alternative and null models are automatically generated by `edgeModel`. The alternative and null models can be accessed using

```
fullModel(edgeObj)
```

```
## ~-1 + grp + time.basis + time.basis:grp
## <environment: 0xb343b78>
```

```
nullModel(edgeObj)
```

```
## ~-1 + grp + time.basis
## <environment: 0xb343b78>
```

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

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

```
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: V1 X1 ... X4 (5 total)
##   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:~-1 + grp + time.basis
## <environment: 0xb343b78>
##
##  Full Model:~-1 + grp + time.basis + time.basis:grp
## <environment: 0xb343b78>
##
## 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
##
## .......
##
```

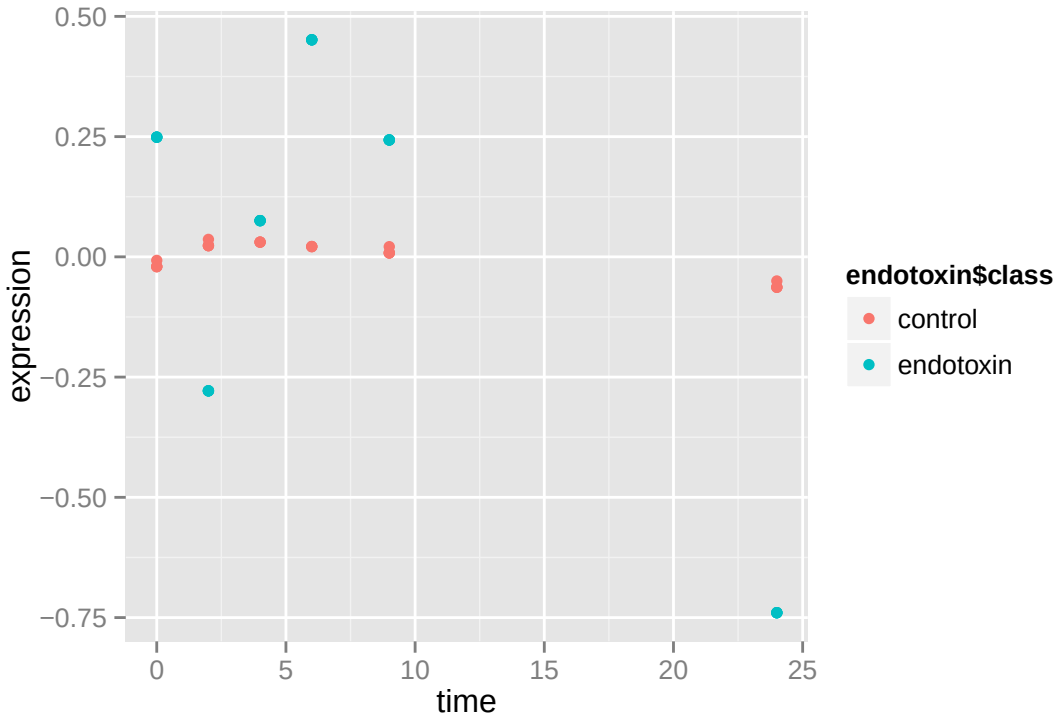
Step 3: Before running any significance analysis, lets view our model fits of the data. The `edgeFit` function can be used to extract residuals and fitted values from the alternative and null models:

```
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:



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.

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

```
edgeLRT <- lrt(edgeObj, nullDistn = "normal")
```

To view a summary of the object:

```
summary(edgeLRT)

##
## ExpressionSet Summary
##
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 5000 features, 46 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: 1 2 ... 46 (46 total)
```

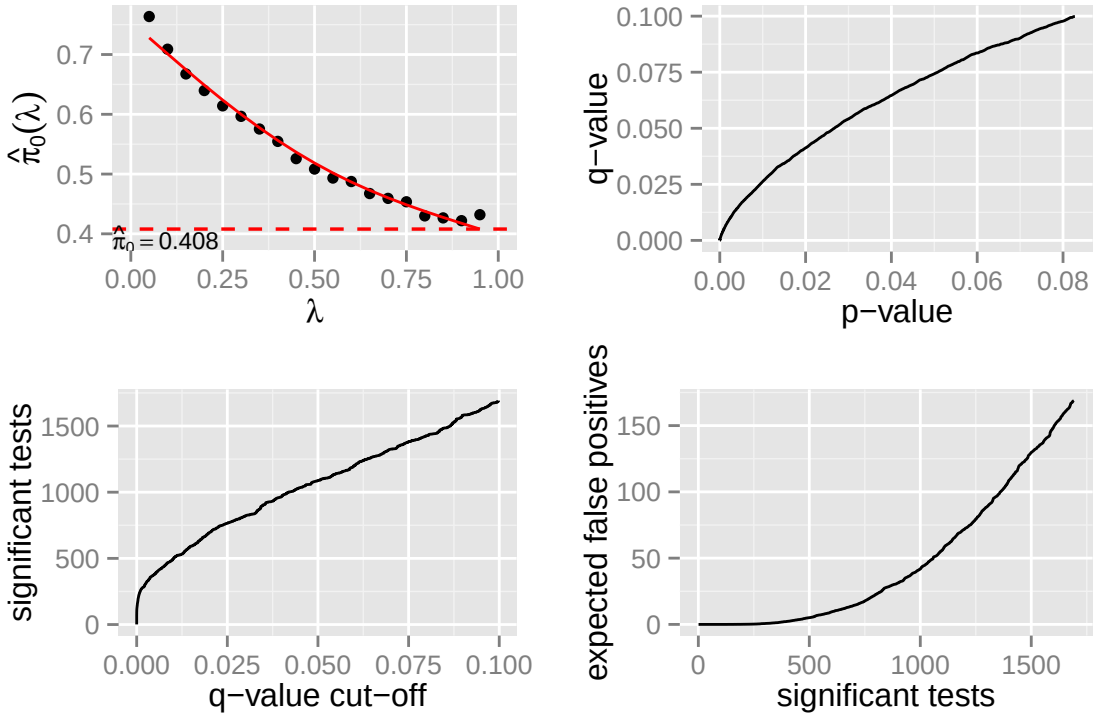
```
##   varLabels: V1 X1 ... X4 (5 total)
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation:
##
## edge Analysis Summary
##
## Total number of arrays: 46
## Total number of probes: 5000
##
## Biological variables:
##  Null Model:~-1 + grp + time.basis
## <environment: 0xb343b78>
##
##  Full Model:~-1 + grp + time.basis + time.basis:grp
## <environment: 0xb343b78>
##
## 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 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
##
## .......
##
##
## Statistical significance summary:
## pi0: 0.408
##
## Cumulative number of significant calls:
##
##           <1e-04 <0.001 <0.01 <0.025 <0.05 <0.1
## p-value      242    386   778   1060  1373 1810
## q-value      134    251   494    764  1089 1692
## local fdr    100    160   301    431   607  895
##             <1
## p-value   5000
## q-value   5000
## local fdr 5000
```

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

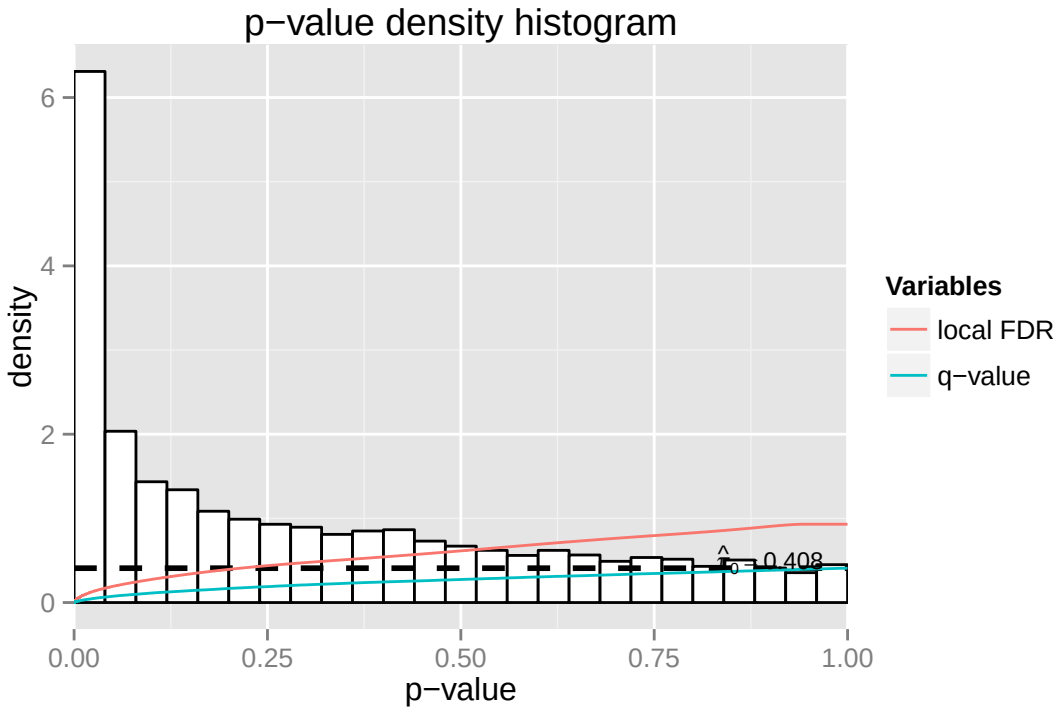
```
qval <- qvalue.obj(edgeLRT)
```

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

```
plot(qval)
```


$\hat{\pi}_0(\lambda)$
0.7
0.6
0.5
0.4
$\hat{\pi}_0 = 0.408$
0.00
0.25
0.50
0.75
1.00
λ
q−value
0.100
0.075
0.050
0.025
0.000
0.00
0.02
0.04
0.06
0.08
p−value
significant tests
1500
1000
500
0
0.000
0.025
0.050
0.075
0.100
q−value cut−off
expected false positives
150
100
50
0
0
500
1000
1500
significant tests

```
hist(qval)
```


p−value density histogram
density
6
4
2
0
$\hat{\pi}_0 = 0.408$
0.00
0.25
0.50
0.75
1.00
p−value
Variables
local FDR
q−value

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` on an `edgeSet object`:

```
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.

To summarize the object using the `odp` method:

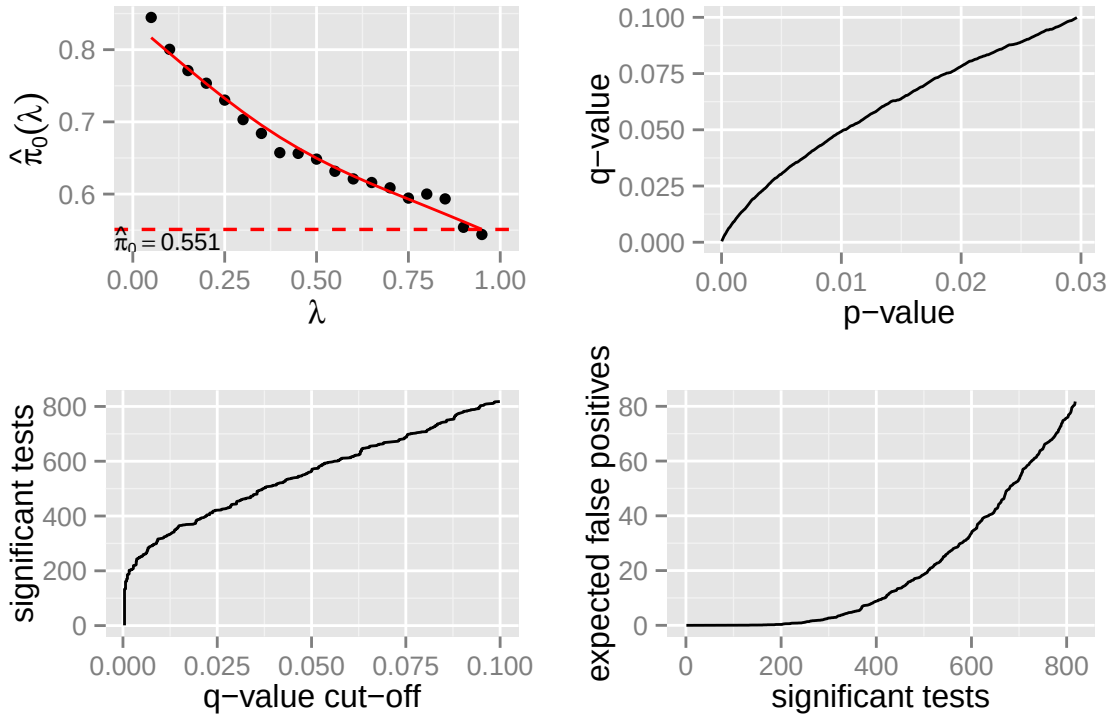
```
summary(edgeODP)
```

```
##
## ExpressionSet Summary
##
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 5000 features, 46 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: 1 2 ... 46 (46 total)
##   varLabels: V1 X1 ... X4 (5 total)
##   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:~-1 + grp + time.basis
## <environment: 0xb343b78>
##
##  Full Model:~-1 + grp + time.basis + time.basis:grp
## <environment: 0xb343b78>
##
## 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
##
## .......
```

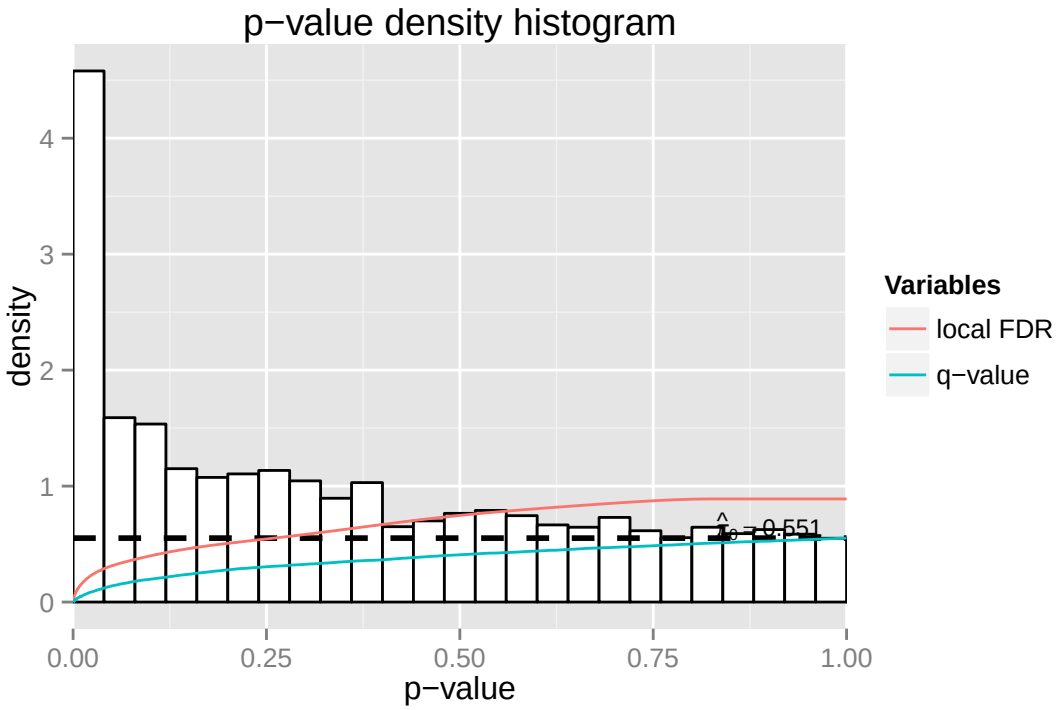
```
##
##
## Statistical significance summary:
## pi0: 0.5512
##
## Cumulative number of significant calls:
##
##           <1e-04 <0.001 <0.01 <0.025 <0.05 <0.1
## p-value      186    304   559    773   988 1398
## q-value        0    170   317    421   564  818
## local fdr      0    133   206    261   333  443
##             <1
## p-value   4999
## q-value   5000
## local fdr 5000
```

Using `plot` and `hist` functions on the `qvalue` object:

```
qval <- qvalue.obj(edgeODP)
plot(qval)
```




```
hist(qval)
```



6.3 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`, `expr`, `gender`, and `location`. There are 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 `expr` variable contains the expression matrix of the experiment.

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

```
edgeObj <- edgeModel(data = gibson$gibexpr, adj.var = model.matrix(~gibson$batch +
    gibson$gender), grp = gibson$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 in the experiment and the `adj.var` argument is the adjustment variables are assigned. The `adj.var` must be in `model.matrix` form. A brief overview of the arguments of `edgeModel`:

- `data` Matrix of expression values
- `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

Additional arguments can be viewed by typing `?edgeModel`. `edgeSet` is the main object in the package and the slots can be viewed by:

```
slotNames(edgeObj)
```

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

The alternative and null models are automatically generated by `edgeModel`. The alternative and null models can be accessed using

```
fullModel(edgeObj)
```

```
## ~-1 + adj.var + grp
## <environment: 0x5fd6168>
```

```
nullModel(edgeObj)
```

```
## ~-1 + adj.var
## <environment: 0x5fd6168>
```

The `adj.var` corresponds to the adjustment variables, the `grp` corresponds to the various location groups in the experiment. The key slot in `edgeObj` is the `qvalue.obj` which should be the only empty slot. The other slots are directly related to the input data and hypothesis models. See the section on the `edgeSet` object for more details on these slots.

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: X.Intercept. gibson.batchB
##     gibson.genderMAL 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:~-1 + adj.var
## <environment: 0x5fd6168>
##
##  Full Model:~-1 + adj.var + grp
## <environment: 0x5fd6168>
##
## .......
##
```

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:

We can see that with our alternative model chosen, the expression goes up and down as the kidney ages for this particular gene.

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.

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

```
edgeLRT <- lrt(edgeObj, nullDistn = "normal")
```

To view a summary of the object:

```
summary(edgeLRT)

##
## ExpressionSet Summary
##
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 2000 features, 46 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: 1 2 ... 46 (46 total)
##   varLabels: X.Intercept. gibson.batchB
##     gibson.genderMAL grp
```

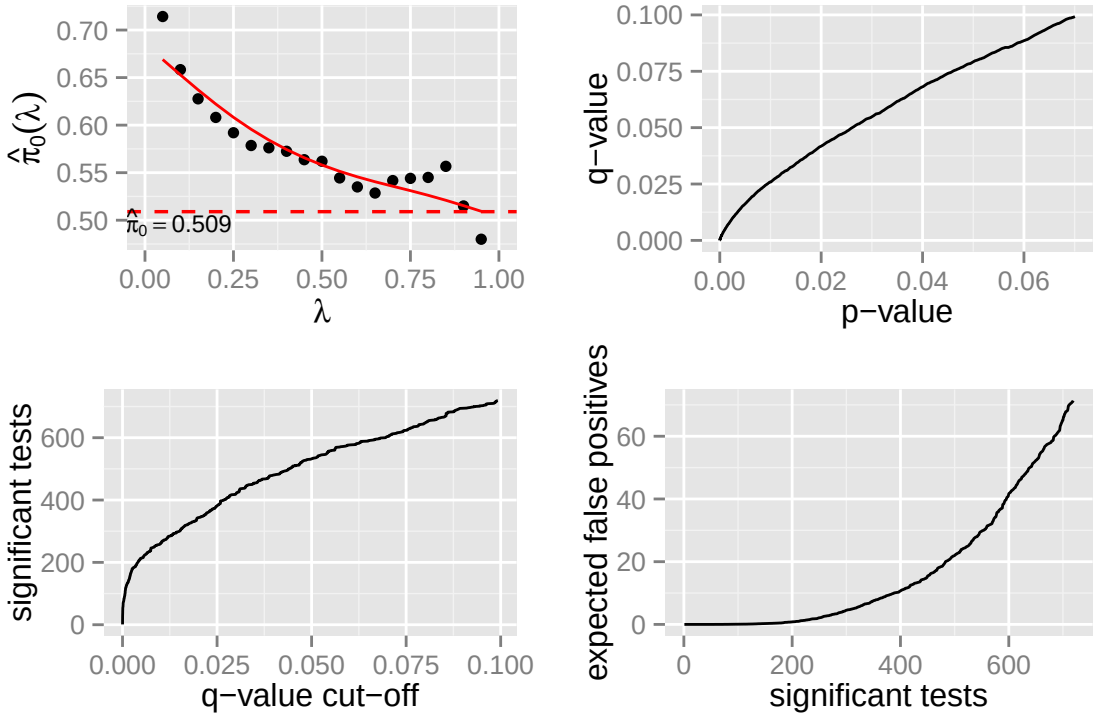
```
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation:
##
## edge Analysis Summary
##
## Total number of arrays: 46
## Total number of probes: 2000
##
## Biological variables:
##  Null Model:~-1 + adj.var
## <environment: 0x5fd6168>
##
##  Full Model:~-1 + adj.var + grp
## <environment: 0x5fd6168>
##
## .......
##
##
## Statistical significance summary:
## pi0: 0.5094
##
## Cumulative number of significant calls:
##
##           <1e-04 <0.001 <0.01 <0.025 <0.05 <0.1
## p-value      118    212   392    527   643  815
## q-value       57    123   260    382   532  719
## local fdr     34     74   176    233   299  421
##             <1
## p-value   2000
## q-value   2000
## local fdr 2000
```

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

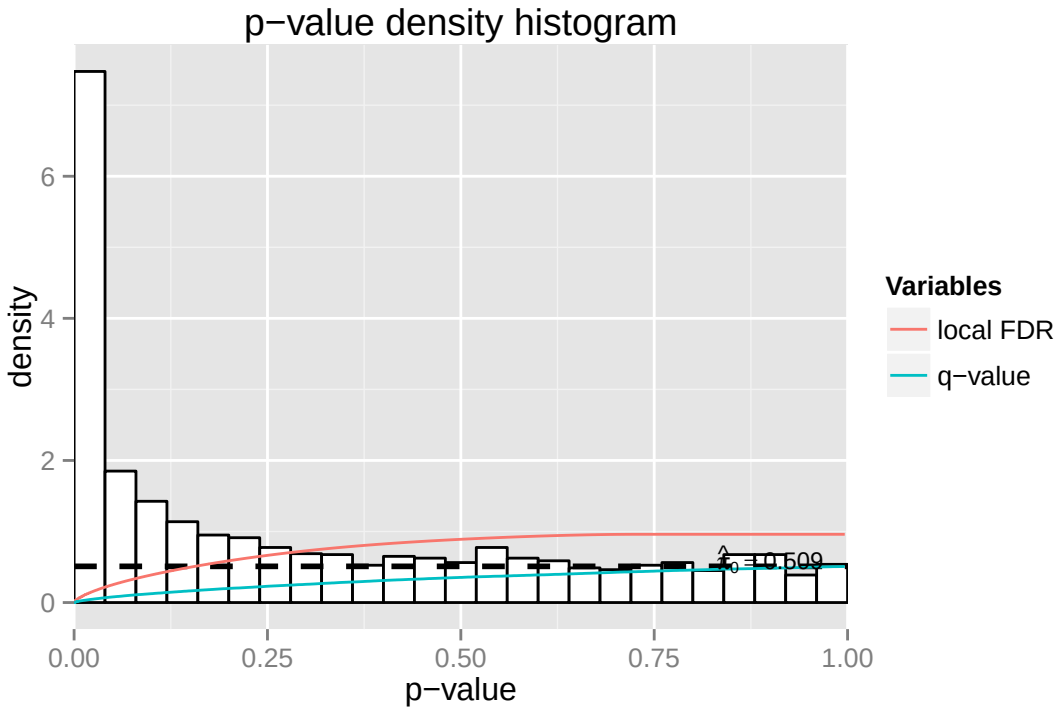
```
qval <- qvalue.obj(edgeLRT)
```

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

```
plot(qval)
```


0.70
0.65
0.60
0.55
0.50
$\hat{\pi}_0(\lambda)$
$\hat{\pi}_0 = 0.509$
0.00
0.25
0.50
0.75
1.00
λ
0.100
0.075
0.050
0.025
0.000
q−value
0.00
0.02
0.04
0.06
p−value
significant tests
600
400
200
0
0.000
0.025
0.050
0.075
0.100
q−value cut−off
expected false positives
60
40
20
0
0
200
400
600
significant tests

```
hist(qval)
```


p−value density histogram
density
6
4
2
0
0.00
0.25
0.50
0.75
1.00
p−value
$\hat{\pi}_0 = 0.509$
Variables
local FDR
q−value

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` on an `edgeSet object`:

```
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.

To summarize the object using the `odp` method:

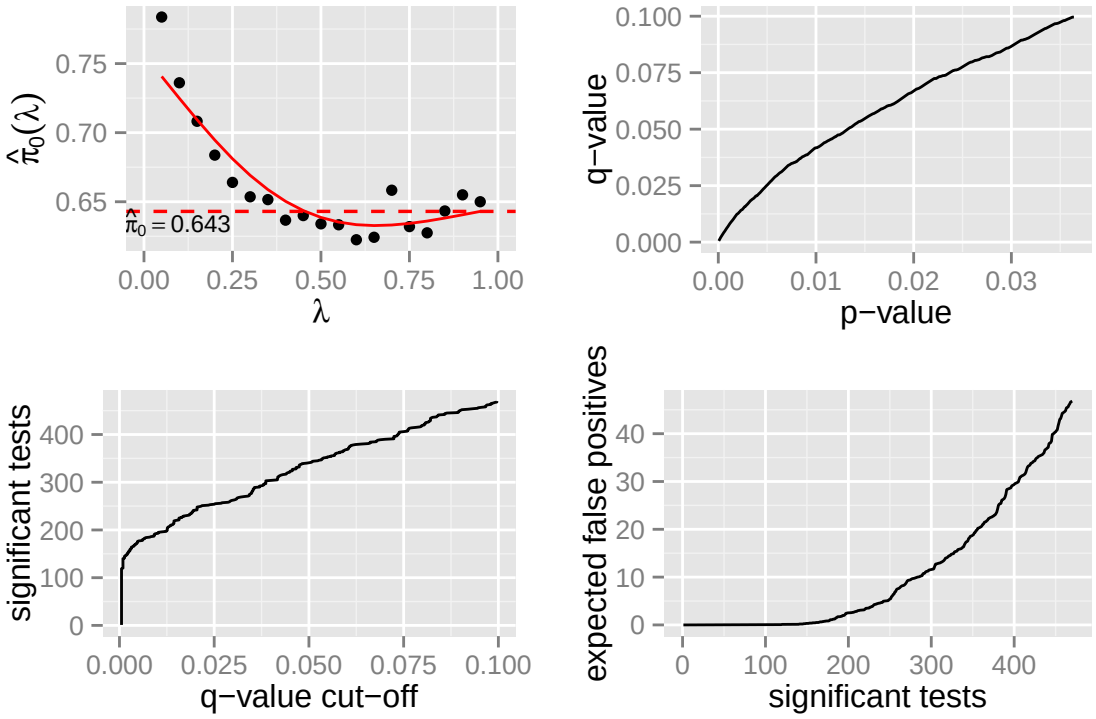
```
summary(edgeODP)
```

```
##
## ExpressionSet Summary
##
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 2000 features, 46 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: 1 2 ... 46 (46 total)
##   varLabels: X.Intercept. gibson.batchB
##     gibson.genderMAL 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:~-1 + adj.var
## <environment: 0x5fd6168>
##
##  Full Model:~-1 + adj.var + grp
## <environment: 0x5fd6168>
##
## .......
##
##
## Statistical significance summary:
## pi0: 0.643
##
```

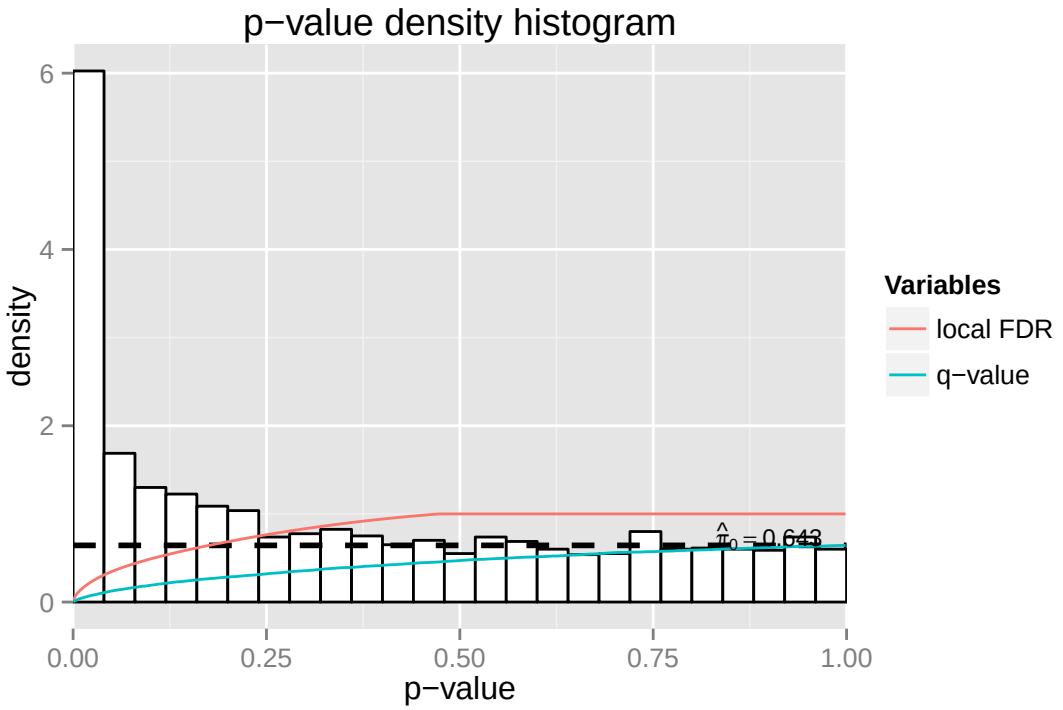
```
## Cumulative number of significant calls:
##
##           <1e-04 <0.001 <0.01 <0.025 <0.05 <0.1
## p-value      119    183   307    414   511  675
## q-value        0    140   192    253   340  470
## local fdr      0      0   156    177   196  257
##             <1
## p-value   2000
## q-value   2000
## local fdr 1337
```

Using `plot` and `hist` functions on the `qvalue` object:

```
qval <- qvalue.obj(edgeODP)
plot(qval)
```




```
hist(qval)
```



7 Using the sva package

`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. To access the new matrices:

```
fullMod <- fullMatrix(newEdgeObj)
nullMod <- nullMatrix(newEdgeObj)
```

See `?sva` for additional input parameters in `edgeSVA`.

8 Advanced topic: Using the ExpressionSet object

The `ExpressionSet` object is another alternative to using `edge` but requires a deeper understanding of R and statistics. Let's create an `ExpressionSet` object from the `kidney` dataset:

```
library(edge)
anonDf <- as(data.frame(age=age, sex=sex), "AnnotatedDataFrame")
expSet <- ExpressionSet(assayData = kidexpr,
                        phenoData = anonDf)
```

`expSet` contains the expression measurements and the covariates of the experiment. To access the expression values, one can use the function `exprs(expSet)` or to access the covariates, `pData(expSet)`. 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.

In the `kidney` experiment they were interested in finding the effect of age on gene expression. In this case, we handle the time variable, `age`, by fitting a natural spline curve [3]. The relevant models for the experiment can be written as

```
library(splines)
nullMod <- ~-1 + sex
altMod <- ~-1 + sex + ns(age, intercept = FALSE,
    df = 4)
```

Where `nullMod` is the null model and `altMod` is the alternative model. The `sex` covariate is an adjustment variable while `age` is the biological variable of interest. It is important to note that it is necessary to include the adjustment variables in the formulation of the alternative models as done above.

Having both `expSet` and the hypothesis models, `edgeSet` can then be used to create an `edgeSet` object:

```
edgeObj <- edgeSet(expSet, full.model = altMod, null.model = nullMod)
```

We can now simply run `odp`, `lrt` or `edgeFit` as in the previous example.

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] Storey, J. D., Xiao, W., Leek, J. T., Tompkins, R. G., and Davis, R. W. Significance analysis of time course microarray experiments. *Proceedings of the National Academy of Sciences of the United States of America (PNAS) 102* (2005), 12837–12842.

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