Bioconductor's edge package
Version 0.99.0

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

July 3, 2014

Contents

1 Introduction

`edge` is a package for significance analysis of DNA micro-array experiments and is able to identify genes that are deferentially expressed between two or more different biological conditions (e.g., healthy versus diseased tissue). There are 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`.

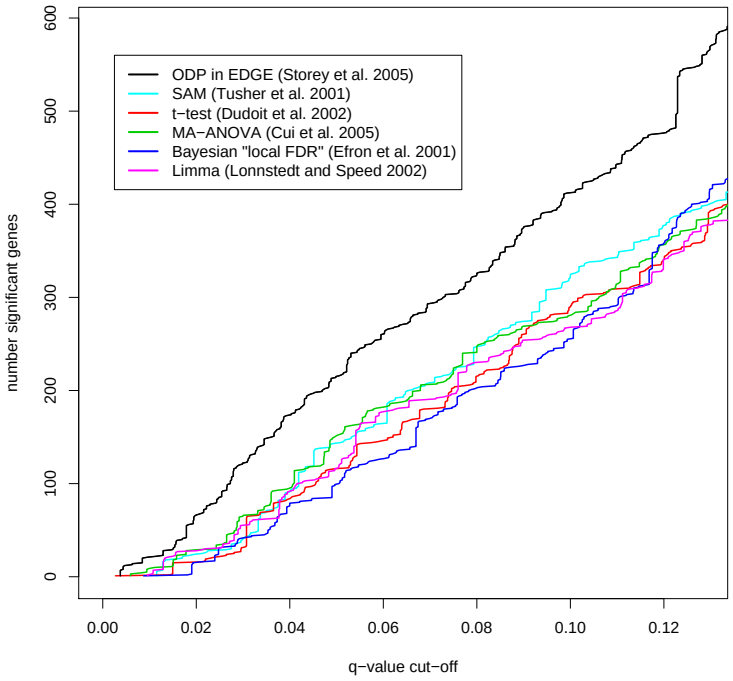


Figure 1: Comparison of EDGE to various other leading methods for identifying differential expressed genes in the Hedenfalk et al., 2001 study. Figure retrieved from Leek et al. (2005).

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

In order to get started using **edge**, it is important to first put all of the experimental data in an **ExpressionSet** 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)
expSet <- ExpressionSet(assayData = kidexpr, phenoData = as(data.frame(sex = sex,
    age = age), "AnnotatedDataFrame"))
```

Once the **ExpressionSet** is created, the alternative and null hypothesis must be formulated based on the experiment. Given an **ExpressionSet** object and the alternative and the null hypothesis, the **odp** or **lrt** function can be used as follows:

```
# Create Models
nullMod <- ~sex
altMod <- ~sex + ns(age, df = 3, intercept = FALSE)
# Create edgeSet object from ExpressionSet object
edgeObj <- edgeSet(expSet, full.model = fModel, null.model = nModel)
# Optimal Discovery Procedure
edgeODP <- odp(edgeObj)
# Likelihood Ratio Test
edgeLRT <- lrt(edgeObj)
```

In the above models, **altMod** is the alternative hypothesis and **nullMod** is the null hypothesis. 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 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 for alternative model
- `fit.null`: fitted values for null model
- `res.full`: residuals for the alternative model
- `res.null`: residuals for the null models
- `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)
```

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`:

- Write the alternative and null models of the experiment. In this manual they will be called `altMod` and `nullMod`.
- Create an `ExpressionSet` object and input the object along with the models in 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

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:

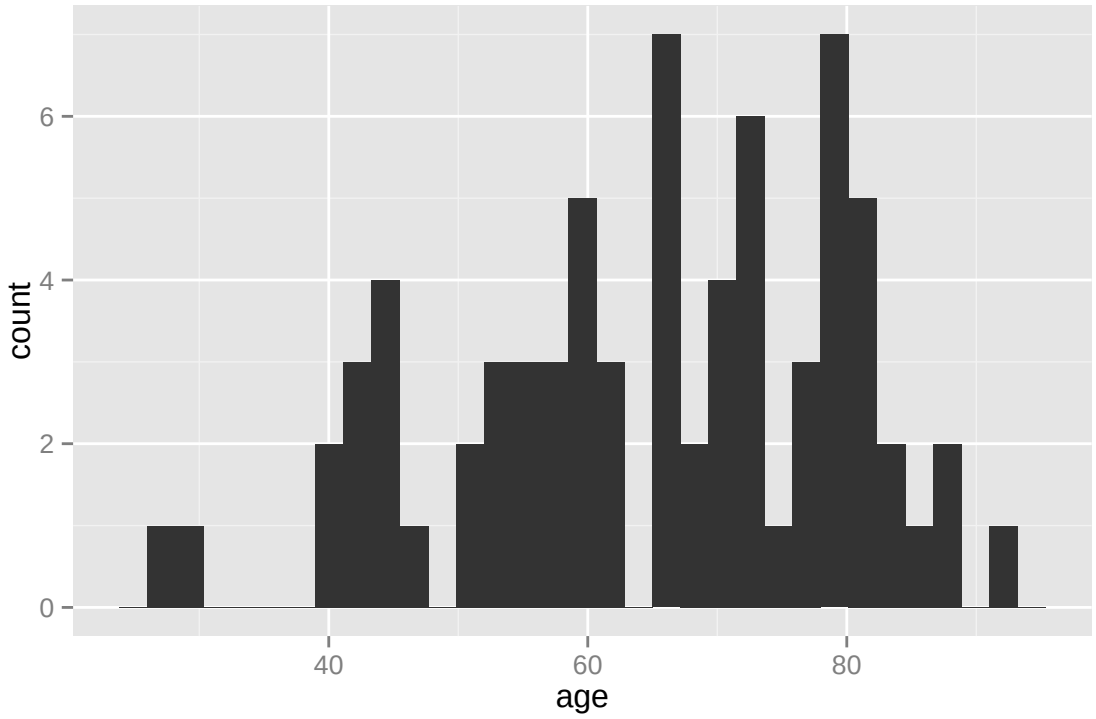
```
data(kidney)
names(kidney)

## [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 for the cortex samples of the `tissue`:

```
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 1: 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.

Step 2: Once the alternative and null models have been decided, create an `ExpressionSet` object:

```
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 ex-

pression 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.

The function `edgeSet` can be used to create an `edgeSet` object from an `ExpressionSet` object:

```
edgeObj <- edgeSet(expSet, full.model = altMod, null.model = nullMod)
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 key slot in the object is the `qvalue.obj` which should be the only empty slot. The other slots are directly related to the `ExpressionSet` object, `full.model` and `null.model`. See the section on `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: age sex
##   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 + sex
##
##  Full Model:~-1 + sex + ns(age, intercept = FALSE, df = 4)
##
## .......
##
```

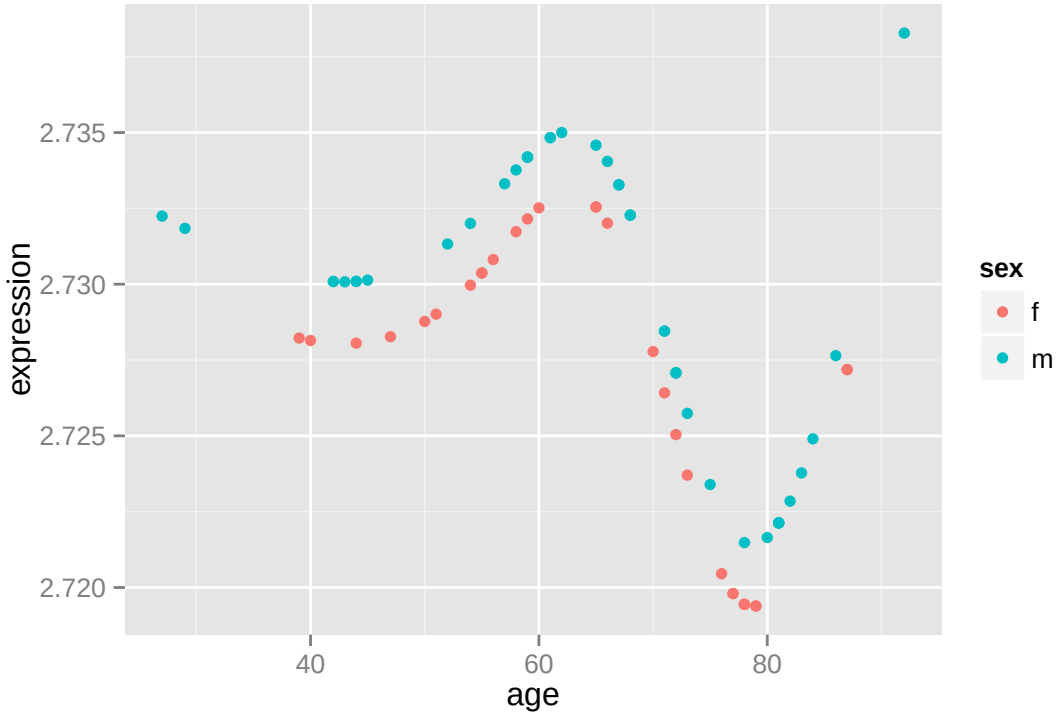
Step 3: Before running any significance analysis, lets view our model fits on the data. The `edgeFit` function can be used to extract residuals and fitted values from the alternative and null models:

```
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)
```

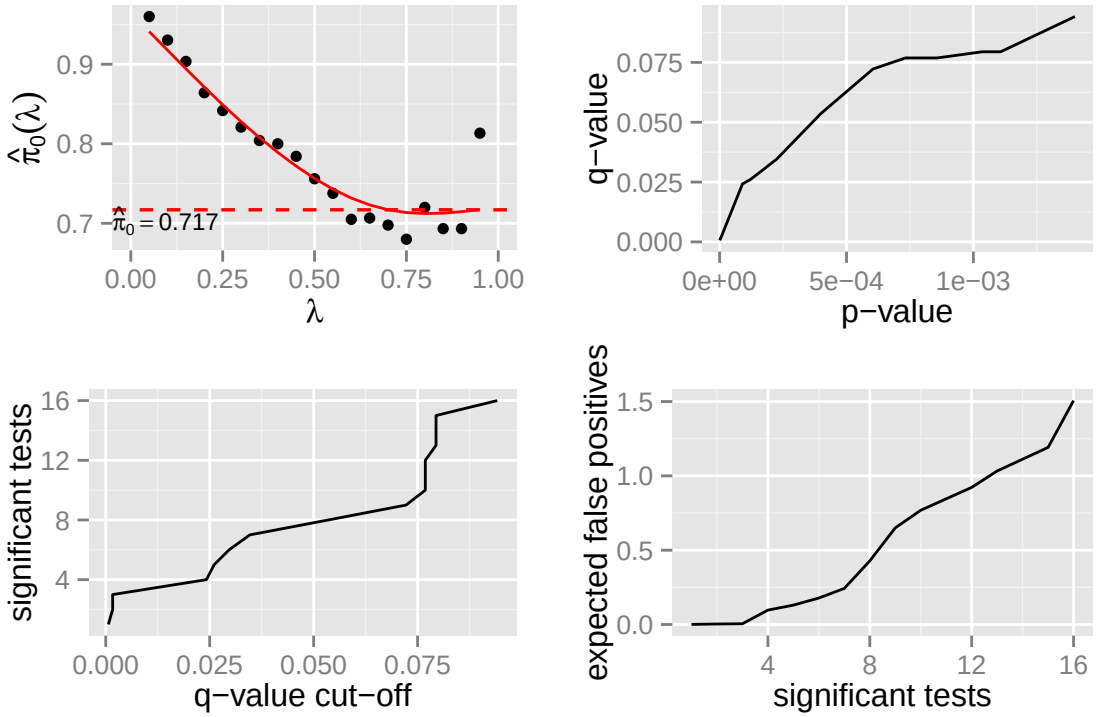
```
##
## ExpressionSet Summary
##
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 1500 features, 72 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: 1 2 ... 72 (72 total)
##   varLabels: age sex
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation:
##
## edge Analysis Summary
##
## Total number of arrays: 72
## Total number of probes: 1500
##
## Biological variables:
##  Null Model:~-1 + sex
##
##  Full Model:~-1 + sex + ns(age, intercept = FALSE, df = 4)
##
## .......
##
##
## 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:

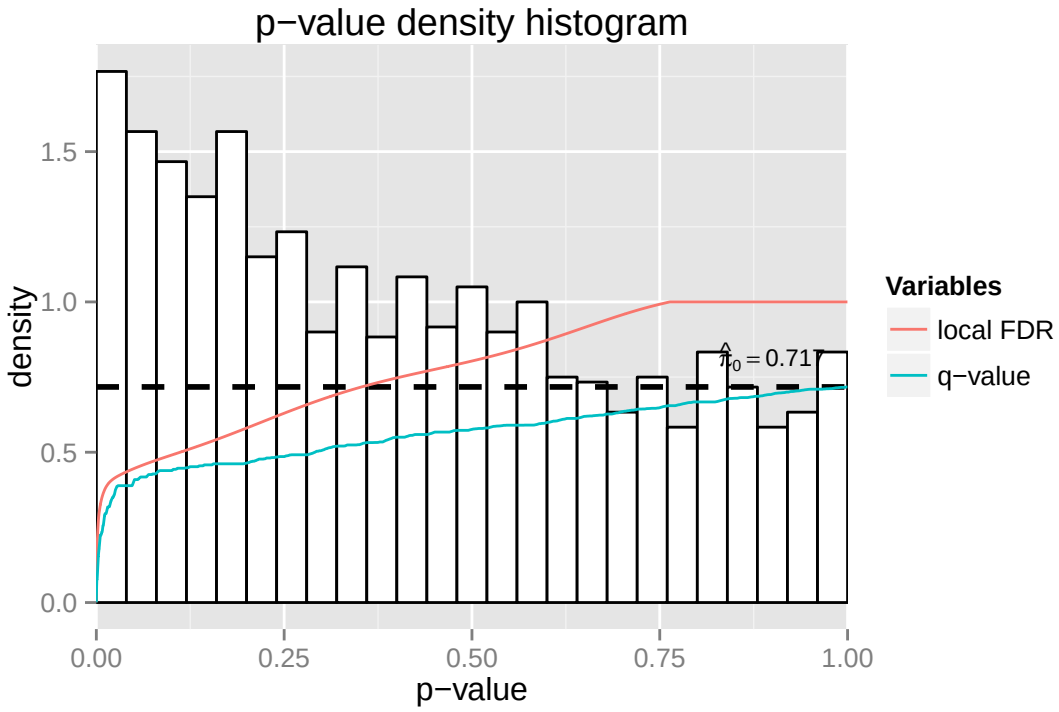
```
qval <- qvalue.obj(edgeLRT)
```

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

```
plot(qval)
```


$\hat{\pi}_0(\lambda)$
0.9
0.8
0.7
$\hat{\pi}_0 = 0.717$
0.00
0.25
0.50
0.75
1.00
λ
q−value
0.075
0.050
0.025
0.000
0e+00
5e−04
1e−03
p−value
significant tests
16
12
8
4
0.000
0.025
0.050
0.075
q−value cut−off
expected false positives
1.5
1.0
0.5
0.0
4
8
12
16
significant tests

```
hist(qval)
```


p−value density histogram
density
1.5
1.0
0.5
0.0
$\hat{\pi}_0 = 0.717$
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.

To summarize the object using the `odp` method:

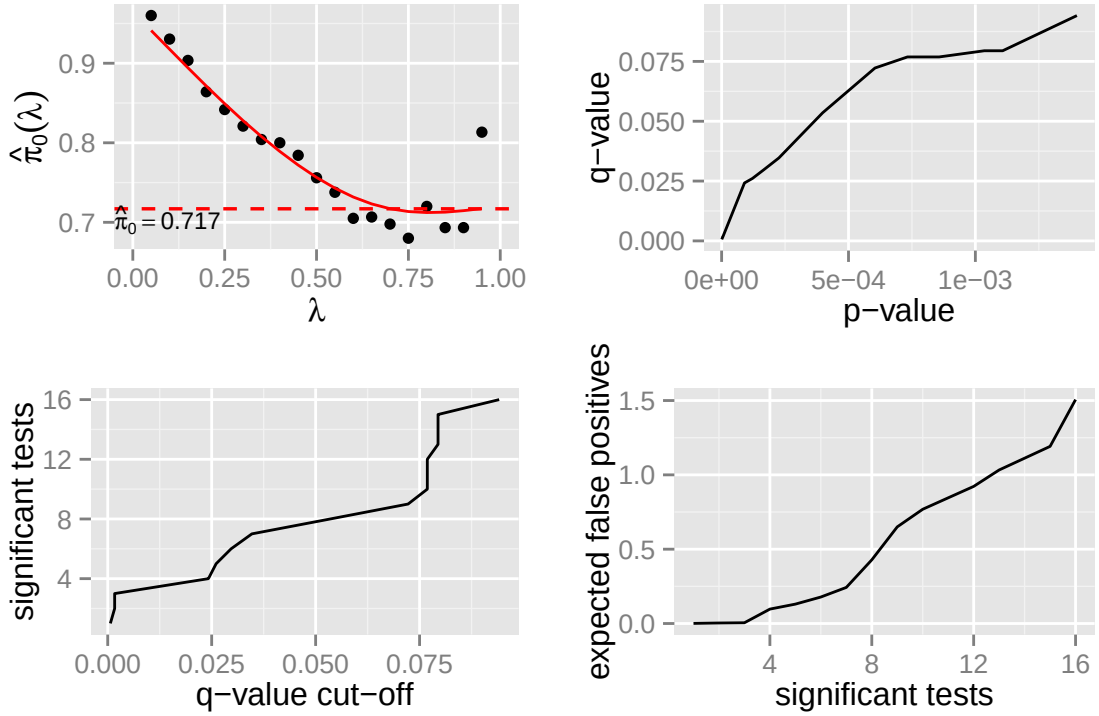
```
summary(edgeODP)
```

```
##
## ExpressionSet Summary
##
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 1500 features, 72 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: 1 2 ... 72 (72 total)
##   varLabels: age sex
##   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 + sex
##
##  Full Model:~-1 + sex + ns(age, intercept = FALSE, df = 4)
##
## .......
##
##
## Statistical significance summary:
## pi0: 0.5265
##
## Cumulative number of significant calls:
##
##           <1e-04 <0.001 <0.01 <0.025 <0.05 <0.1
## p-value        7     27    85    151   224  366
```

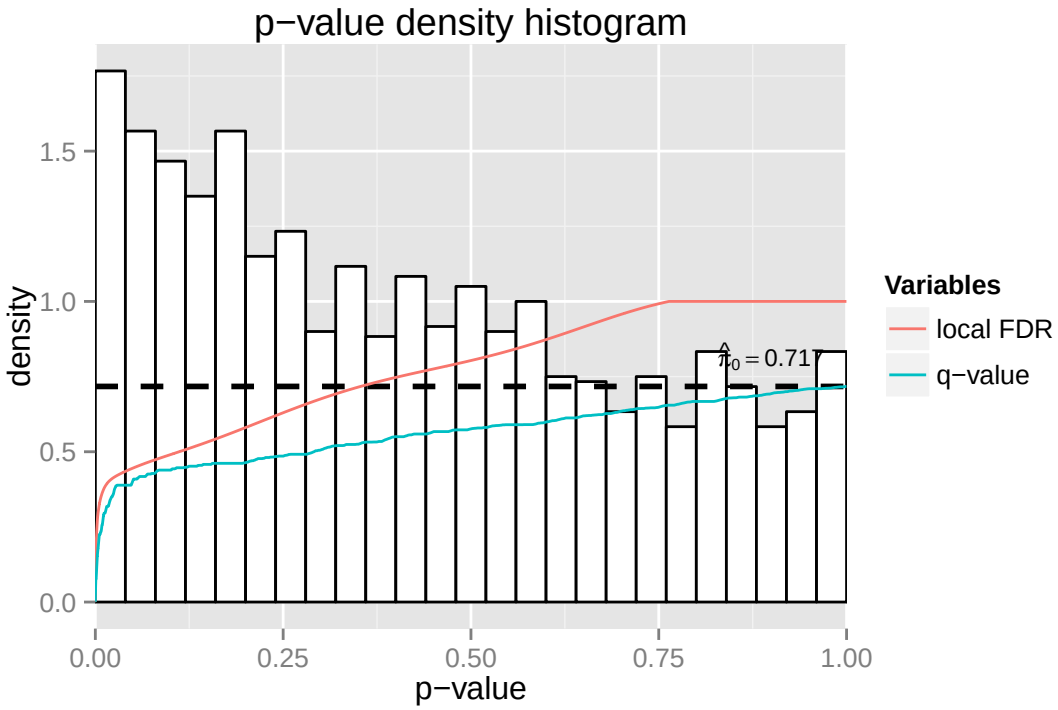
```
## q-value         0      0    12     26     43  129
## local fdr       0      0     7     16     26   53
##              <1
## p-value    1500
## q-value    1500
## local fdr  1500
```

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

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

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.