User's guide to the "benchmarkR" package

Xiaobei Zhou, Charity Law, Mark D. Robinson

July 24, 2015

Contents

List of Examples

1 Preliminaries

benchmarkR is an R package designed with a simple purpose: to calculate, evaluate and visualize the performance of a classification method (e.g., a statistical method that returns p-values), such as those common in statistical genomics. For example, much of our research has been related to methods to find differentially expressed genes, differential splicing, differential methylation and so on. In principle, *benchmarkR* can be used for any similar situation, for example, where statistical tests are conducted across the features (e.g., genes) of a dataset and one wants to contrast sensitivity and false positive (discovery) rates. The package would typically be used for synthetic datasets or for datasets that are accompanied by independent validated "truth" (in quotes because validation data are typically measured with some uncertainty).

Researchers already look at the performance of methods through simulations using ROC curves, precision-recall plots, false detection plots and the like – so, what does *benchmarkR* offer? In these plots, we are often interested to know at the same time whether methods are "calibrated", that is, do the methods actually achieve the false discovery control that the cutoffs used would suggest? The main idea behind the *benchmarkR* package is to simply augment the the set of standard plots that many methodologists already look at, with some additional calibration information. To do this, we in fact make use of the excellent codebase available in the *ROCR* package for performance assessment.

The typical usage of the *benchmarkR* package would involve p-values, in order to do a calibration assessment. More generally, the package will happily take "scores", but in this case, no calibration assessment can be done.

This guide provides an overview of the *benchmarkR* package and detailed information on how to use it across the different types of plots and features to customize the visualizations.

benchmarkR version: 0.0-19

Cittation:

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

2 Abbreviations

These abbreviations are used through the whole user guide:

Abbreviations:

- ROC: receiver operating characteristic [1]
- TPR: true positive rate
- FPR: false positive rate
- FDR: false discovery rate
- FD: false discovery
- BH: Benjamini & Hochberg [1]

3 A quick start

In the simplest case, *benchmarkR* would start with a set of p-values and corresponding truth labels and a call will be made to one of the plotting functions, as follows:

```
library(benchmarkR)
re <- SimResults(pval,labels)
benchmarkR(re)

rocX(re)
fdX(re)
powerFDR(re)
```

Example 1: A quick start

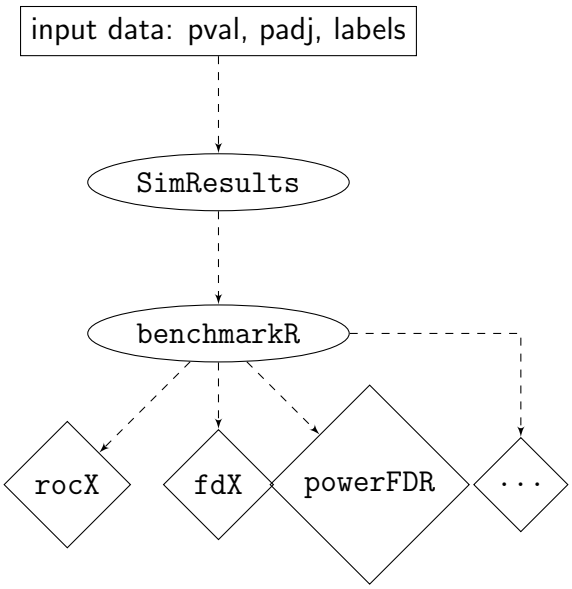
4 A short summary of benchmarkR

benchmarkR is an R package designed to evaluate and visualize classification performance for reference datasets. Generally speaking, the *benchmarkR* package can be divided into two major parts:

1. *SimResults*: a S4 class object to store the p-values, true labels and other necessary values for further analysis and evaluation.
2. `benchmarkR`: a method to produce a visualized benchmark result including a 3-panel plot of a ROC curve (`rocX`), a false discovery (FD) plot (`fdX`) and a true-positive-rate versus achieved false-discovery plot(`powerFDR`).

Its whole structure is shown below:

[1] See details from: http://en.wikipedia.org/wiki/Receiver_operating_characteristic.



The individual methods, `rocX`, `fdX`, `powerFDR`, can be called individually and can be highly customized.

Customized methods:

- `rocX`: ROC curve augmented with X point
- `fdX`: false discovery plot augmented with X point
- `powerFDR`: power (TPR) versus achieved false discovery rate (FDR) plot

More details of these can be found at Section 6.2, 6.3 and 6.4.

5 Data preparation: build a RclassSimResults object

To use the *benchmarkR* package, it is best to build a `SimResults` object, which is just a container for the results. A `SimResults` object contains three basic inputs:

- `pval`: a vector or matrix containing p-values from a classifier or a test.
- `padj`: a vector or matrix containing the adjusted p-values (optional).
- `labels`: a numeric vector indicating class labels (positives with 1, negatives with 0). For more details, see `?ROCR::prediction`.

The function `SimResults` is just the constructor to build a *SimResults* object. To give a practical example, we have pre-packaged the simulation results from Zhou et al. (2014) [**?**] to highlight the construction of a *SimResults* object (see also `help("Pickrell")`):

```
library(benchmarkR)
data(Pickrell)
re <- SimResults(pval=Pickrell$pval, labels=Pickrell$labels)
       ## padj is missing, selected method (BH) is used to generate padj.
re
## An object of class "SimResults"
## @pval
##          edgeR limma_voom    DESeq2
## [1,] 8.090e-09  1.654e-05 1.470e-08
## [2,] 4.257e-01  5.725e-01 4.598e-01
## [3,] 1.470e-01  1.353e-01 7.331e-02
## [4,] 6.600e-01  3.158e-01 6.397e-01
## [5,] 3.389e-02  1.759e-01 1.389e-01
## 3995 more rows ...
##
## @padj
##          edgeR limma_voom    DESeq2
## [1,] 7.525e-07   0.001087 7.172e-07
## [2,] 9.217e-01   0.922638 9.288e-01
## [3,] 6.571e-01   0.627453 4.822e-01
## [4,] 9.742e-01   0.827246 9.715e-01
## [5,] 2.953e-01   0.687859 6.512e-01
## 3995 more rows ...
##
## @labels
## [1] 1 0 0 0 0
## 3995 more elements ...
```

Example 2: A practical example – `pval` is a matrix

In this example, adjusted p-values (`padj`) are missing. In this case, p-values will be adjusted using the chosen method (default is Benjamini-Hochberg [1], `padjMethod="BH"`, but others from `p.adjust` can be used).

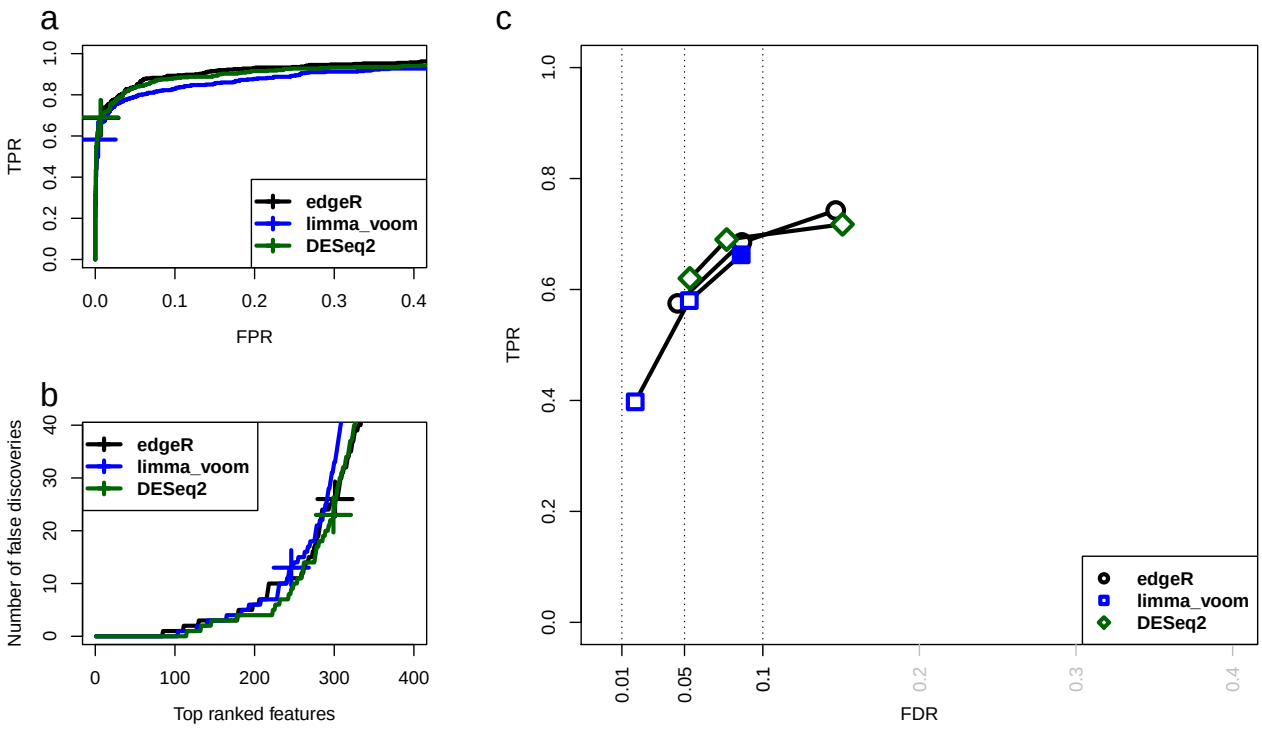
Note that `pval` (and `padj`) can be a vector or matrix (for a single method or multiple methods, respectively). If a matrix is used, each column should represent a score (p-value) for a classification method and each row should represent a feature.

6 Present a benchmark

6.1 benchmarkR: wrapper function to create a multi-panel performance assessment plot

Following with the last example of Section 5, a visualized benchmark result can be generated with:

```
data(Pickrell)
re <- SimResults(pval=Pickrell$pval, labels=Pickrell$labels)
       ## padj is missing, selected method (BH) is used to generate padj.
benchmarkR(re)
```



Example 3: Usage of benchmarkR

benchmarkR will produce a set of performance assessment plots for benchmarking. A 3-panel plot is created with: an ROC curve (rocX), a false discovery (FD) plot (fdX) and a true-positive-rate versus achieved false-discovery plot (powerFDR). All plots are augmented with calibration information of some form. The rocX and fdX highlight a point on the curve for the chosen threshold (default: the method's FDR=5%) while powerFDR shows the power versus the achieved FDR of a method.

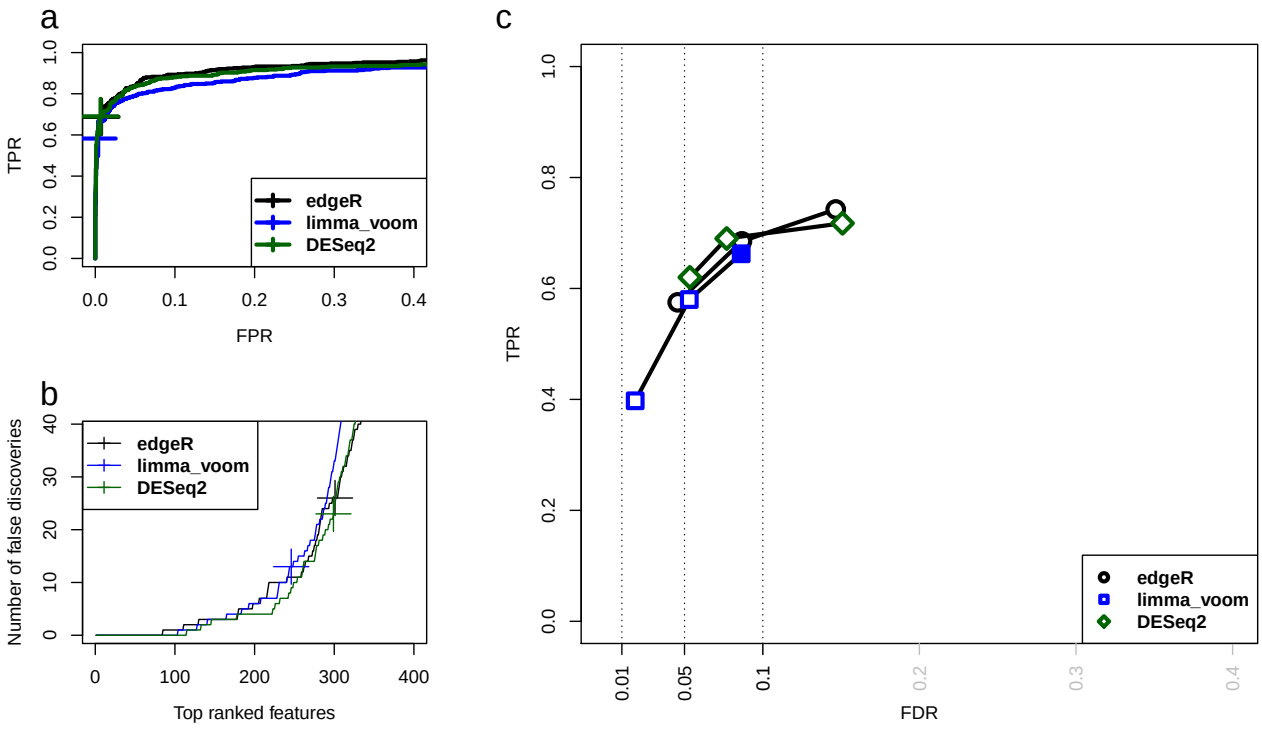
benchmarkR provides many graphical parameters allowing flexible adjustment of each sub-panel. The graphical parameters of par() can directly work in benchmarkR, but follows one common rule:

common rule of graphical parameters in `benchmarkR`:

- a single vector (e.g., 'lwd=1') representing all sub-panels
- a list representing each sub-panel separately (e.g., 'lwd=list(3,1,3)')

An example is shown below:

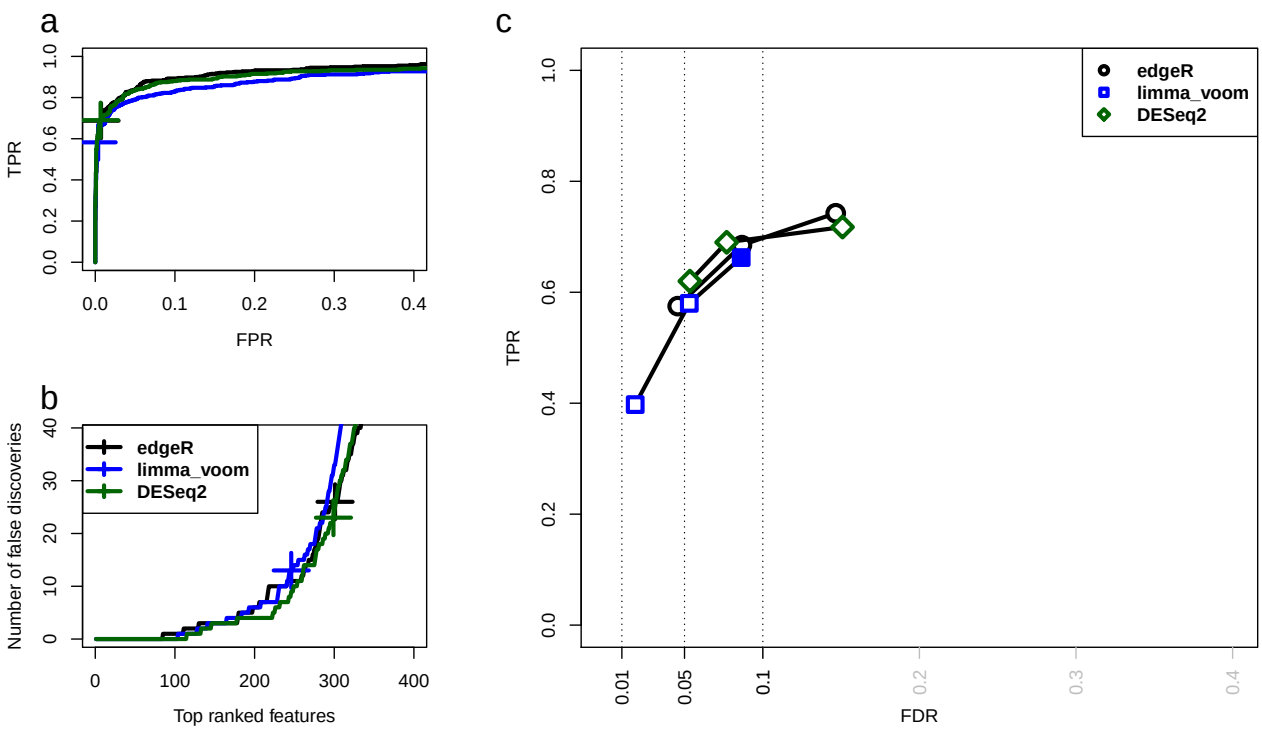
```
benchmarkR(re, lwd=list(3,1,3), cex=2)
```



Example 4: Graphical parameters in `benchmarkR`

In this case 'cex' are the same for all panels, while 'lwd' are 3 or 1. In addition, 'legend' is plugged in `benchmarkR` function as a graphical parameter that can be also freely customized.

```
benchmarkR(re, legend=list(NULL,"topleft","topright"))
```



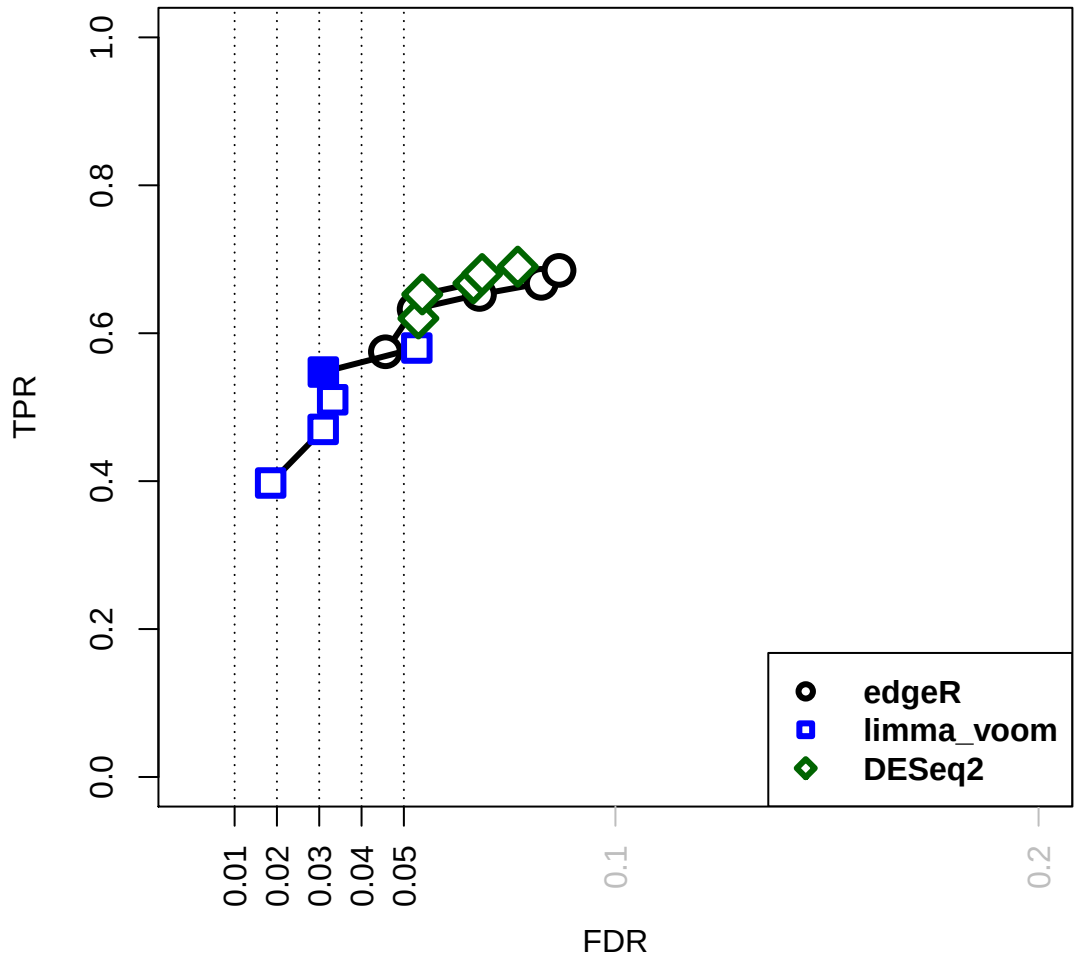
Example 5: Set up 'legend' in `benchmarkR`

6.2 `powerFDR`: power (TPR) versus achieved false discovery rate (FDR) plot

`powerFDR` calculates and plots power (true positive rate) versus the *achieved* FDR using adjusted P-values and the known true positives (labels). By default, it calculates TPR and achieved FDR at three cutoffs: 0.01, 0.05 and 0.10.

The constructor function `powerFDR` will invisibly return an S4 object of *powerFDR*. A typical usage for an object of *SimResults* is:

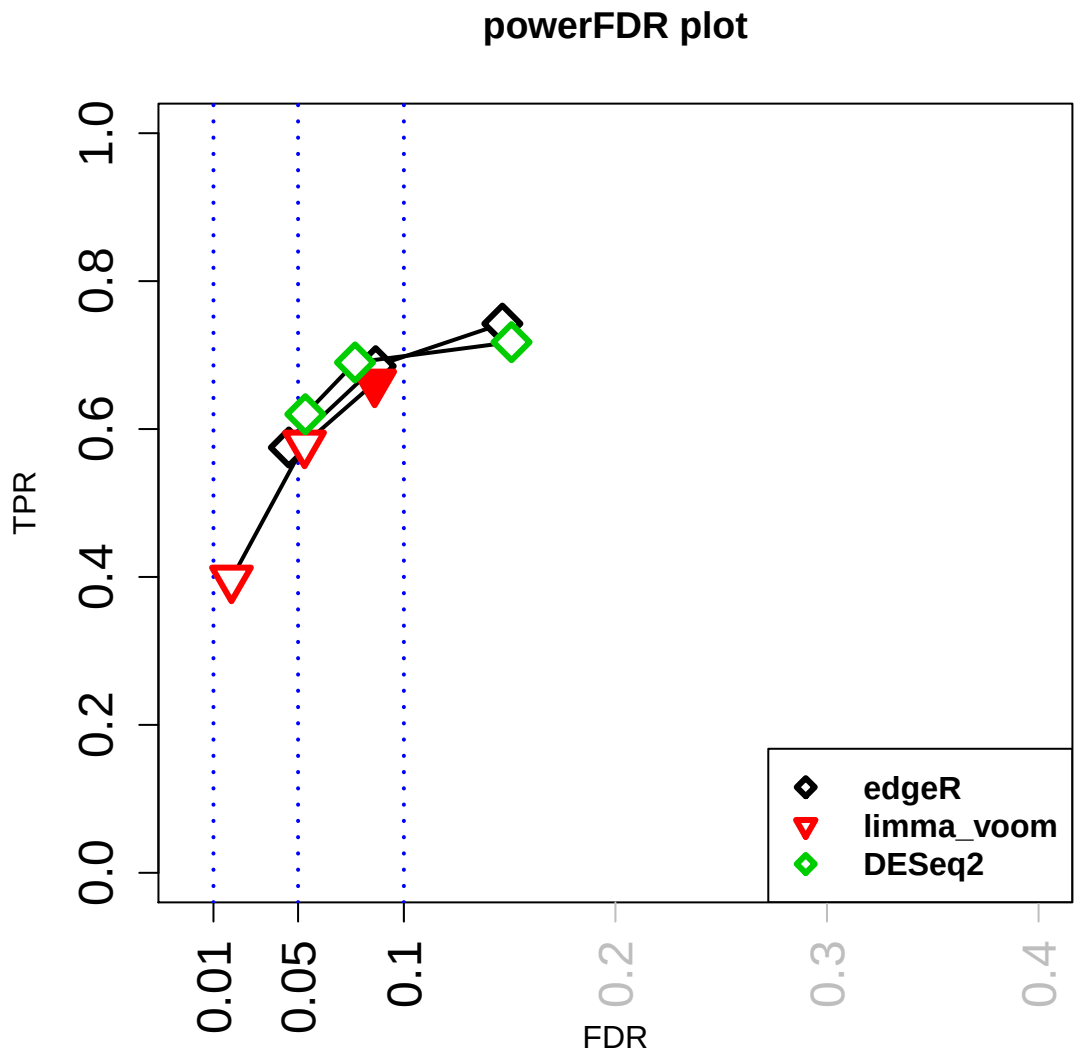
```
data(Pickrell)
re <- SimResults(pval=Pickrell$pval, labels=Pickrell$labels)
       ## padj is missing, selected method (BH) is used to generate padj.
powerFDR(re, threshold=c(0.01,0.02,0.03,0.04,0.05), xlim=c(0,0.2))
```



Example 6: `powerFDR` examples

Most of the graphical parameters such as 'col', 'cex', 'pch', etc. from `par`, can be directly passed to `plot` for the power-FDR plots. 'point.type', letter indicating how power-FDR values should be plotted: `"b"` for both points and lines; `"p"` for points only; and `"l"` for lines only. 'col.line' and 'lwd.line', 'col' and 'lwd' of line connecting power-FDR points, if 'point.type' is either `"b"` or `"l"`. 'lwd.threshold', 'lty.threshold' and 'col.threshold' are 'wd', 'lty' and 'col' referred to the lines drawn for the threshold.

```
p <- powerFDR(re, plot=FALSE)
plot(p, cex=2, pch=c(23,25), col=1:3, main="powerFDR plot",
     lwd.line=2, cex.axis=1.5, col.threshold=4, lwd.threshold=2)
```

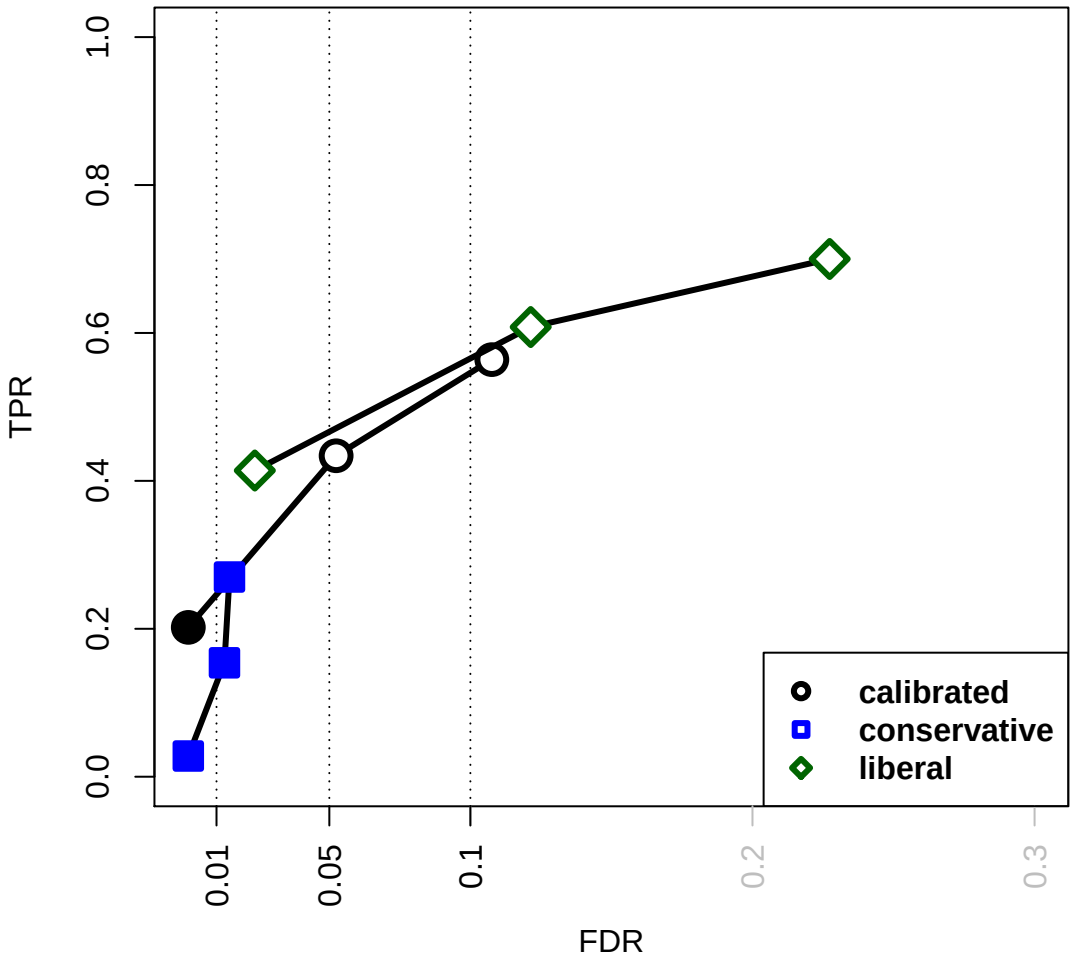


Example 7: Graphical parameters of `powerFDR` plot

A method's FDR is considered to be controlled if its achieved FDR falls below the set threshold; if this happens, plotted power-FDR points are filled-in. power-FDR points are unfilled, or empty, if the FDR is not controlled and is equal to or greater than the cutoff it is assessed at. Note that the fill-unfill option is only compatible with 'pch' values of 21 to 25. Visually, this means that for a given threshold, a point (of the same color) will be filled if it falls left of the threshold line, and will be unfilled if it sits on or to the right of the threshold line.

The function `powerFDR` can also directly work with input value `padj`, `padj` and `labels`.

```
data(calibration)
padj <- calibration$padj
labels <- calibration$labels
powerFDR(padj=padj, labels=labels, xlim=c(0,.3))
```



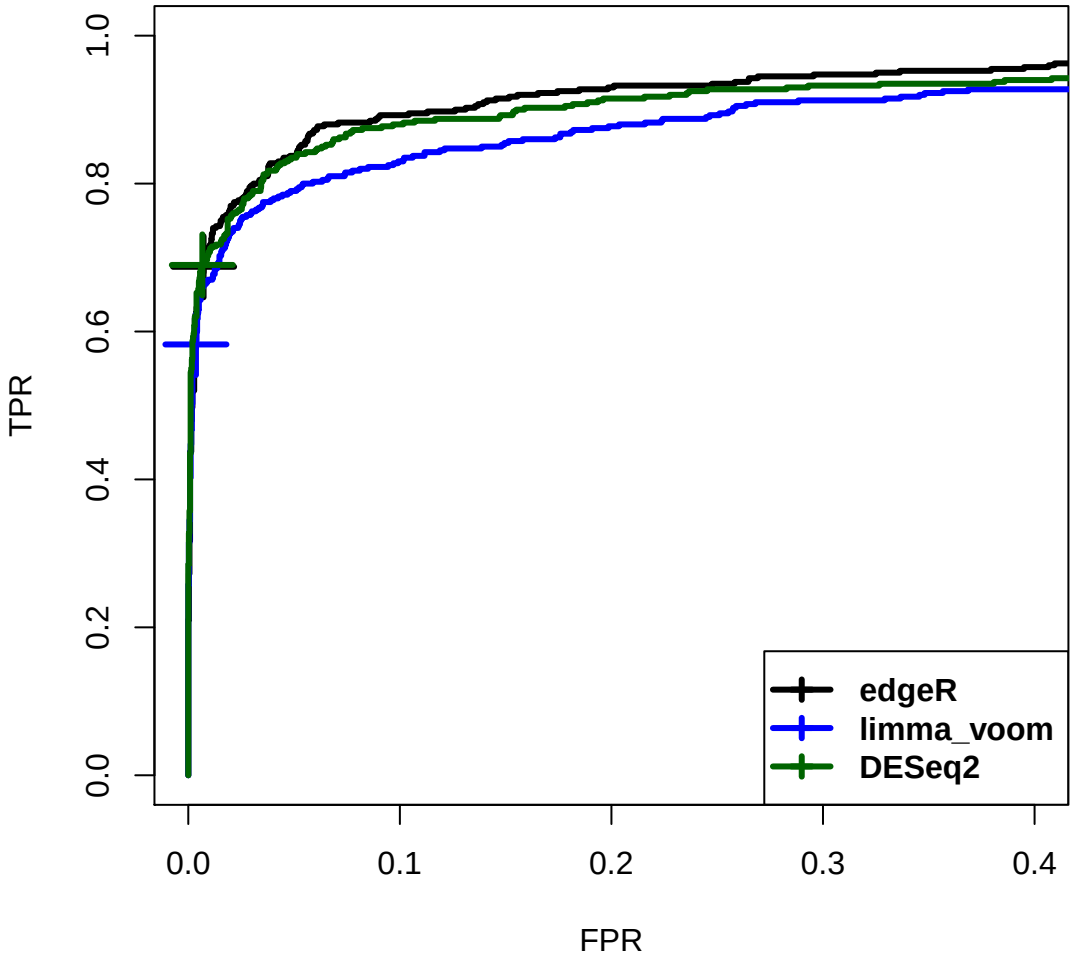
Example 8: Direct usage of `powerFDR` plot without `SimResults`

6.3 `rocX`: ROC curve augmented with X point

`rocX` will produce a ROC plot with an X point marking the location of the method's FDR (at a chosen level), using the features of *ROCR* [2]. The special feature of `rocX` beyond *ROCR* is that an X point is employed to shows the actual position along the ROC curve (e.g., at the method's 5% FDR point).

The function `rocX` is just the constructor to build a *rocX* object. It can be simply used as:

```
data(Pickrell)
re <- SimResults(pval=Pickrell$pval, labels=Pickrell$labels)
       ## padj is missing, selected method (BH) is used to generate padj.
r <- rocX(re)
```



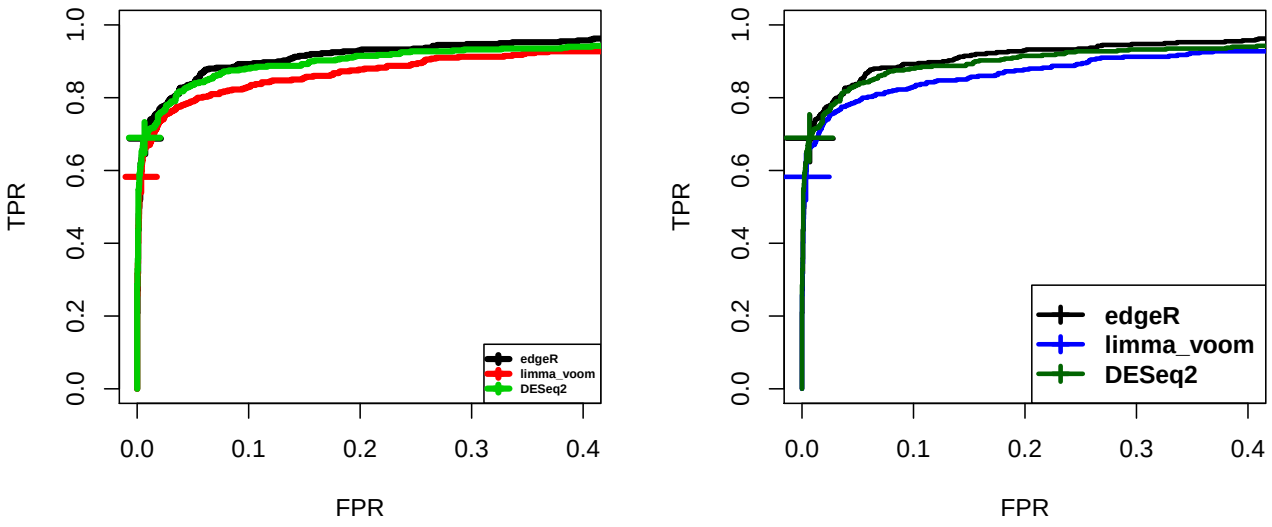
Example 9: `rocX`

`rocX` will call *prediction* and *performance* from the *ROCR* package using the p-value and labels from the `SimResults` object. Additionally, the value of TPR and FPR corresponding to the threshold X will be calculated from the adjust P-values. We introduced `rocX` to augment the standard ROC curves in order to show where on the curve a method is operating.

For convenience, an `rocX` object can be re-plotted by function `plot` allowing flexible resetting of graphical parameters from `par()`.

There are several special graphical parameters ('cexX', 'pchX' and 'colX') referring to setting of X point of *rocX*. You can simply change them as:

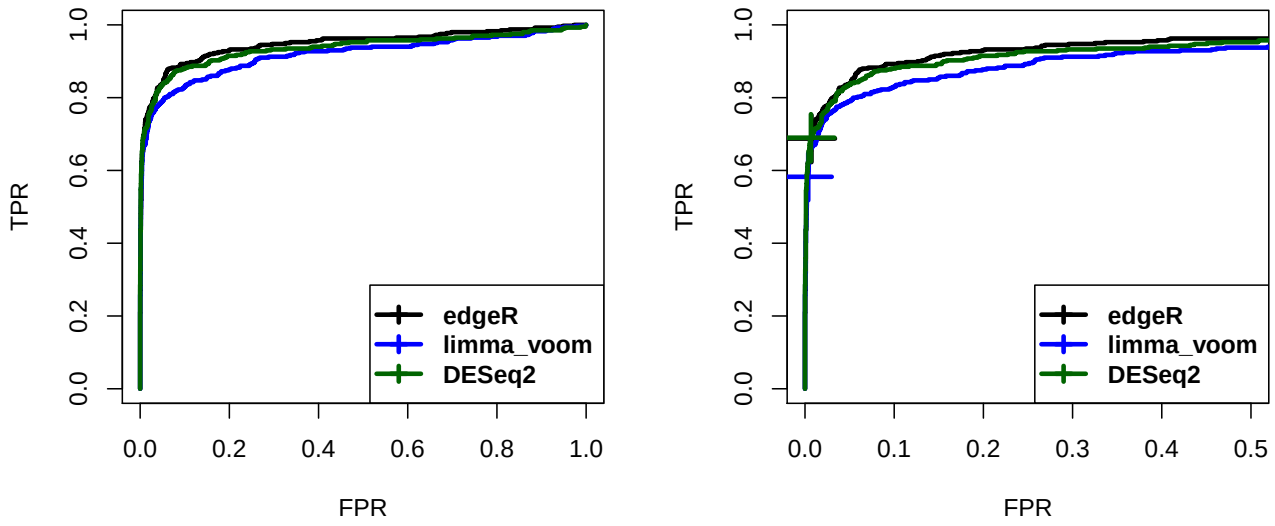
```
par(mfrow=c(1,2))
plot(r, lwd=4, cex=1, col=1:3)
plot(r, cexX=3, lwdX=3)
```



Example 10: re-plot of `rocX`, using special graphical parameters

In the next examples, additional variations are shown. For example, you can remove X, if it is not needed. Or you can change 'xlim'.

```
par(mfrow=c(1,2))
rocX(re, thresholdX=NULL)
rocX(re, xlim=c(0,.5))
```



Example 11: Various examples of `rocX`

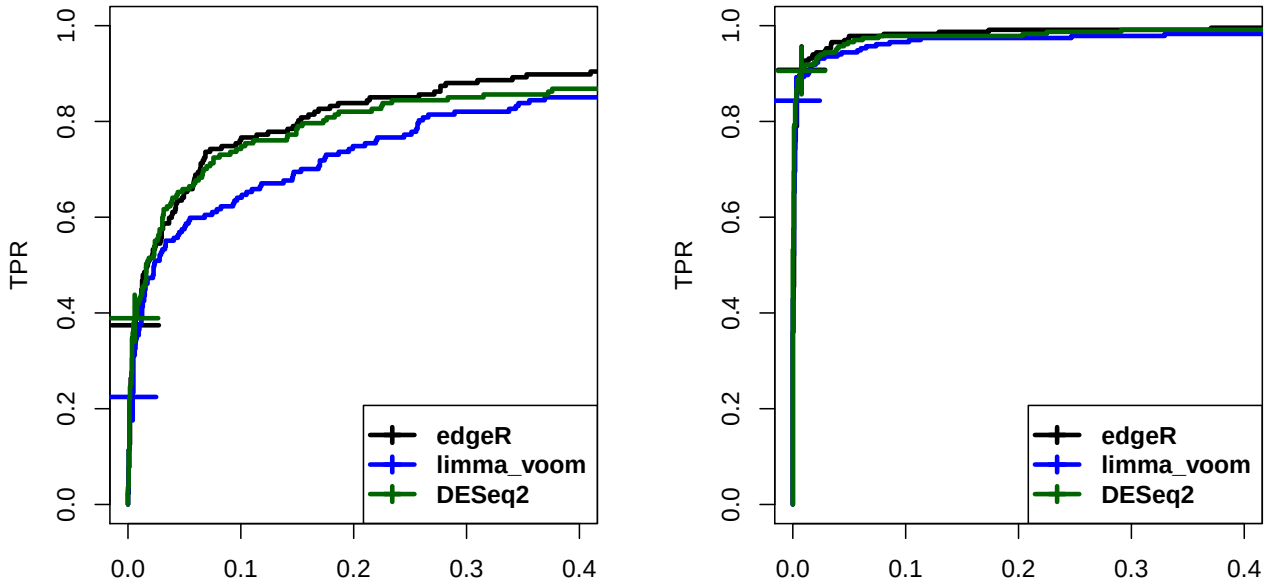
In addition, `rocX` allows any flexible transformation formula for input data, in order to have more resolution along the ROC curve.

transformation:

= "1-x" (default setting)
= "-log10(x)" (pval or padj is sparse.)
= "x" (inverse score)
= "f(x)" (any other functions)

`rocX` can be separated into several sub-ROC curves by using the 'stratify' factor, which can be useful to subdivide results according to other features of the assay (e.g., gene expression strength):

```
rm <- rowMeans(Pickrell$counts)
f <- cut(log(rm), 2)
re <- SimResults(pval=Pickrell$pval, labels=Pickrell$labels, stratify=f)
par(mfrow=c(1,2), mar=c(2,5,3,1), mgp=c(2.6,1,0))
rocX(re, ylim=c(0,1))
```



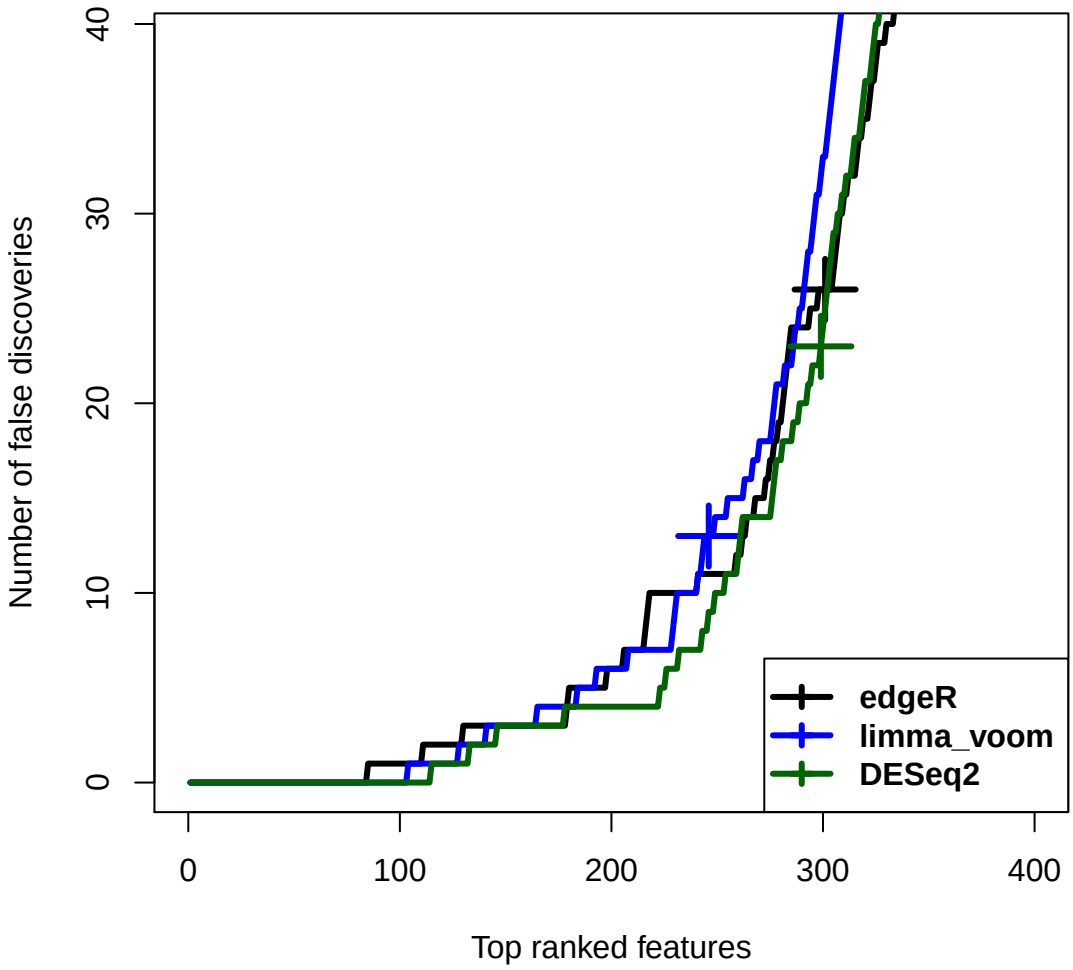
Example 12: 'stratify' factor of `rocX`

6.4 `fdX`: false discovery plot augmented with X point

`fdX` plots the cumulative number of false discoveries versus the number of top ranked features with an X point marking the location of the method's FDR (at a chosen level).The constructor function `fdX` will produce an S4 object of `fdX`.

A couple quick examples of `fdX` include:

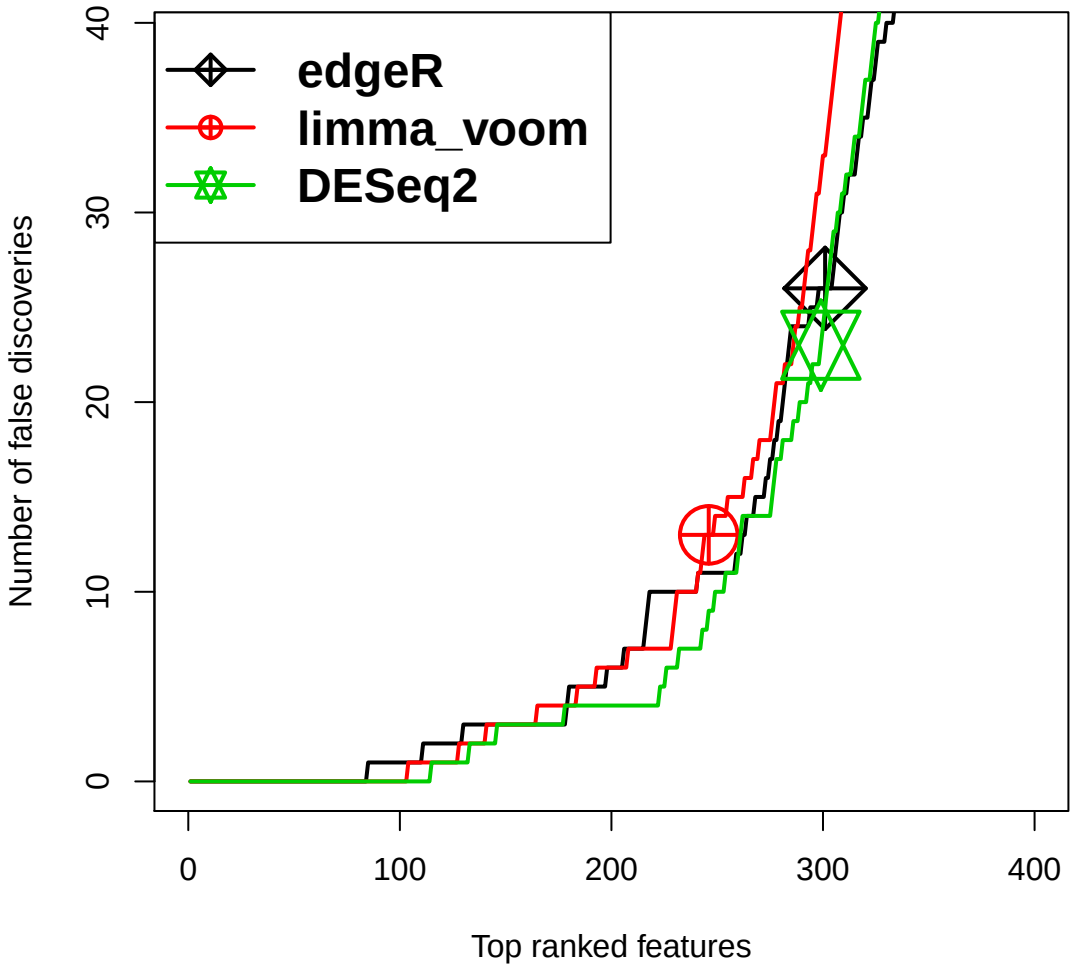
```
data(Pickrell)
re <- SimResults(pval=Pickrell$pval, labels=Pickrell$labels)
       ## padj is missing, selected method (BH) is used to generate padj.
f <- fdX(re)
```



Example 13: `fdX` examples

The `fdX` function (and objects) use almost all the characteristics of the `rocX` function, including 'thresholdX', 'transformation', special graphical parameters ('cexX', 'pchX' and 'colX') and so on. A customized example is:

```
plot(f, lwd=2, col=1:3, pchX=9:11, cex=3, legend=list("topleft"))
```

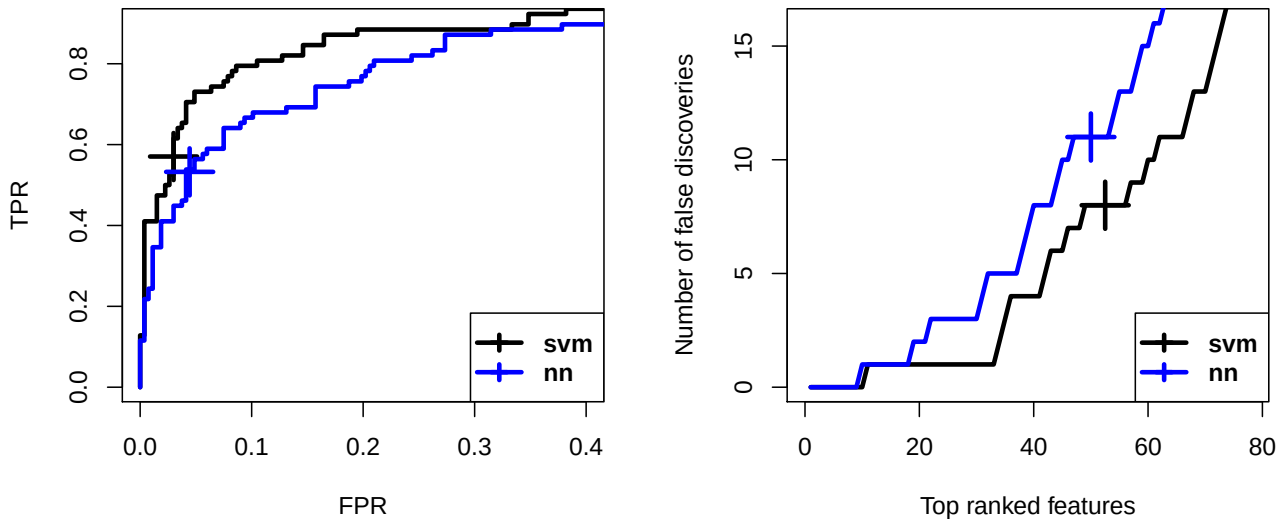


Example 14: A customized `fdX` example

6.5 "Scores" instead of p-values ?

More generally for assessing methods beyond p-values, *benchmarkR* can take predictions or scores from classification methods on any scale. In this case, calibration assessment (`powerFDR`) is not practical. The X point of `rocX` and `fdX` marks the location on the scale of the methods' scores, according to the chosen. For this setting, `pval` and `padj` are used to load scores (e.g., `pval=score, padj=score`); `transformation` is set up as "x" instead of "1-x"; `thresholdX` is chosen depending on the scores (e.g, `thresholdX=0`). A modified example from *ROCR* evaluating performances between support vector machines (SVM) and neural networks (NN) applied to the HIV envelope protein dataset [3] is presented below:

```
library(ROCR)
          ## Warning:  package 'gplots' was built under R version 3.2.0
data(ROCR.hiv)
svm <- ROCR.hiv$hiv.svm$predictions[[1]]
nn <- ROCR.hiv$hiv.nn$predictions[[1]]
s <- cbind(svm=svm, nn=nn)
labels <- ROCR.hiv$hiv.svm$labels[[1]]
labels[labels==-1] <- 0
res <- SimResults(pval=s, padj=s, labels=labels)
par(mfrow=c(1,2))
rocX(res, transformation="x",  thresholdX=0)
fdX(res, transformation="x",  thresholdX=0)
```



Example 15: "scores" example

7 Session info

- R version 3.1.1 (2014-07-10), x86_64-unknown-linux-gnu
- Locale: LC_CTYPE=en_CA.UTF-8, LC_NUMERIC=C, LC_TIME=en_CA.UTF-8, LC_COLLATE=C, LC_MONETARY=en_CA.UTF-8, LC_MESSAGES=en_CA.UTF-8, LC_PAPER=en_CA.UTF-8, LC_NAME=C, LC_ADDRESS=C, LC_TELEPHONE=C, LC_MEASUREMENT=en_CA.UTF-8, LC_IDENTIFICATION=C
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: ROCR~1.0-7, benchmarkR~0.0-19, gplots~2.17.0, knitr~1.5

- Loaded via a namespace (and not attached): BiocStyle~1.4.1, KernSmooth~2.23-13, bitops~1.0-6, caTools~1.17.1, evaluate~0.5.1, formatR~0.10, gdata~2.17.0, gtools~3.4.2, highr~0.3, stringr~0.6.2, tools~3.1.1

References

[1] Y~Benjamini, Y~Hochberg, and Y.~Benjamini, Y. and Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series B Methodological*, 57(1):289–300, 1995. doi:10.2307/2346101.

[2] Tobias Sing, Oliver Sander, Niko Beerenwinkel, and Thomas Lengauer. ROCR: Visualizing classifier performance in R. *Bioinformatics*, 21(20):3940–3941, 2005.

[3] Tobias Sing, Niko Beerenwinkel, and Thomas Lengauer. Learning mixtures of localized rules by maximizing the area under the ROC curve. In *Proceedings of the First International Workshop on ROC Analysis in Artificial Intelligence*, volume~22, pages 96–98. Citeseer, 2004. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.2786&rep=rep1&type=pdf.