Protein Microarray Data Analysis using the *PAA* Package

Michael Turewicz

October 13, 2014

Contents

1 Introduction

1.1 General information

Protein Array Analyzer (*PAA*) is a package for protein microarray data analysis (esp., *ProtoArray* data). It imports single color (protein) microarray data that has been saved in 'gpr' file format. After pre- processing (background correction, batch filtering, normalization) univariate feature pre-selection is performed (e.g., using the "minimum M statistic" approach - hereinafter referred to as "mMs", [1]). Subsequently, a multivariate feature selection is conducted to discover biomarker candidates. Therefore, either a frequency-based backwards elimination approach or ensemble feature selection can be used. *PAA* provides a complete toolbox of analysis tools including several different plots for results examination and evaluation.
In this vignette the general workflow of *PAA* will be outlined by analyzing an exemplary data set that accompanies this package.

1.2 Installation

The recommended way to install *PAA* is to type the commands described below in the *R* console *comment: (note: an active internet connection is needed)*:

```
> # only if you install a Bioconductor package for the first time
> source("http://www.bioconductor.org/biocLite.R")
> # else
> library("BiocInstaller")
> biocLite("PAA", dependencies=TRUE)
```

This will install *PAA* including all dependencies.

Furthermore, *PAA* has an external dependency that is needed to provide full functionality. This external dependency is the free *C++* software package *"Random Jungle"* that can be downloaded from `http://www.randomjungle.de/`. *comment: Note: PAA will be usable without Random Jungle. However, it needs this package for random jungle recursive feature elimination (RJ-RFE) provided by the function `selectFeatures()`. Please follow the instructions for your OS in the README file to install Random Jungle properly on your machine.*

2 Loading PAA and importing data

After launching *R*, the first step of the exemplary analysis is to load *PAA*.

```
> library(PAA)
```

New microarray data should be imported using the function `loadGPR()` which is mainly a wrapper to *limma*'s function `read.maimages()` featuring optional duplicate aggregation for *ProtoArray* data. *PAA* supports the import of files in 'gpr' file format. The imported data is stored in an expression list object (*EList*, respectively, *EListRaw*, see Bioconductor package *limma*). Paths to a targets file and to a folder containing 'gpr' files (all 'gpr' files in this folder that are listed in the targets file will be read) are mandatory arguments. The folder that can be obtained by the command `system.file("extdata", package = "PAA")` contains an exemplary targets file that can be used as a template. Below, the first 3 rows of this targets file are shown.

```
> targets <- read.table(file=list.files(system.file("extdata", package="PAA"),
+  pattern = "^targets", full.names = TRUE), header=TRUE)
> print(targets[1:3,])
```

```
  ArrayID                    FileName Group  Batch       Date Array SerumID
1     AD1 GSM734833_PA41992_-_AD1.gpr    AD Batch1 10.11.2010 41992     AD1
2     AD2 GSM734834_PA41994_-_AD2.gpr    AD Batch2 10.11.2010 41994     AD2
3     AD3  GSM734835_PA42006_-AD3.gpr    AD Batch1 12.11.2010 42006     AD3
```

The columns "ArrayID", "FileName", and "Group" are mandatory. "Batch" is mandatory for microarray data that has been processed in batches. The remaining three columns as well as custom columns containing further information (e.g., clinical data) are optional.

If `array.type` is set to `"ProtoArray"` (default) duplicate spots will be aggregated. After importing, the object can be saved in a '.RData' file for further sessions. In the following code chunk, `loadGPR()` is demonstrated using a exemplary dummy data set that comes with PAA and has been created from the real data described below.

```
> gpr <- system.file("extdata", package="PAA")
> targets <- list.files(system.file("extdata", package="PAA"),
+  pattern = "dummy_targets", full.names=TRUE)
> dummy.elist <- loadGPR(gpr.path=gpr, targets.path=targets)
> save(dummy.elist, file=paste(gpr, "/DummyData.RData",
+    sep=""))
```

PAA comes with an exemplary protein microarray data set. This 20 Alzheimer's disease serum samples vs. 20 controls data is a subset of a publicly available *ProtoArray* data set. It can be downloaded from the repository *"Gene Expression Omnibus"* (*GEO*, `http://www.ncbi.nlm.nih.gov/geo/`, record "GSE29676"). It has been contributed by *Nagele E et al.* [2] (note: Because a data set stored in 'gpr' files would be too large to accompany this package the exemplary data is stored as an '.RData' file).

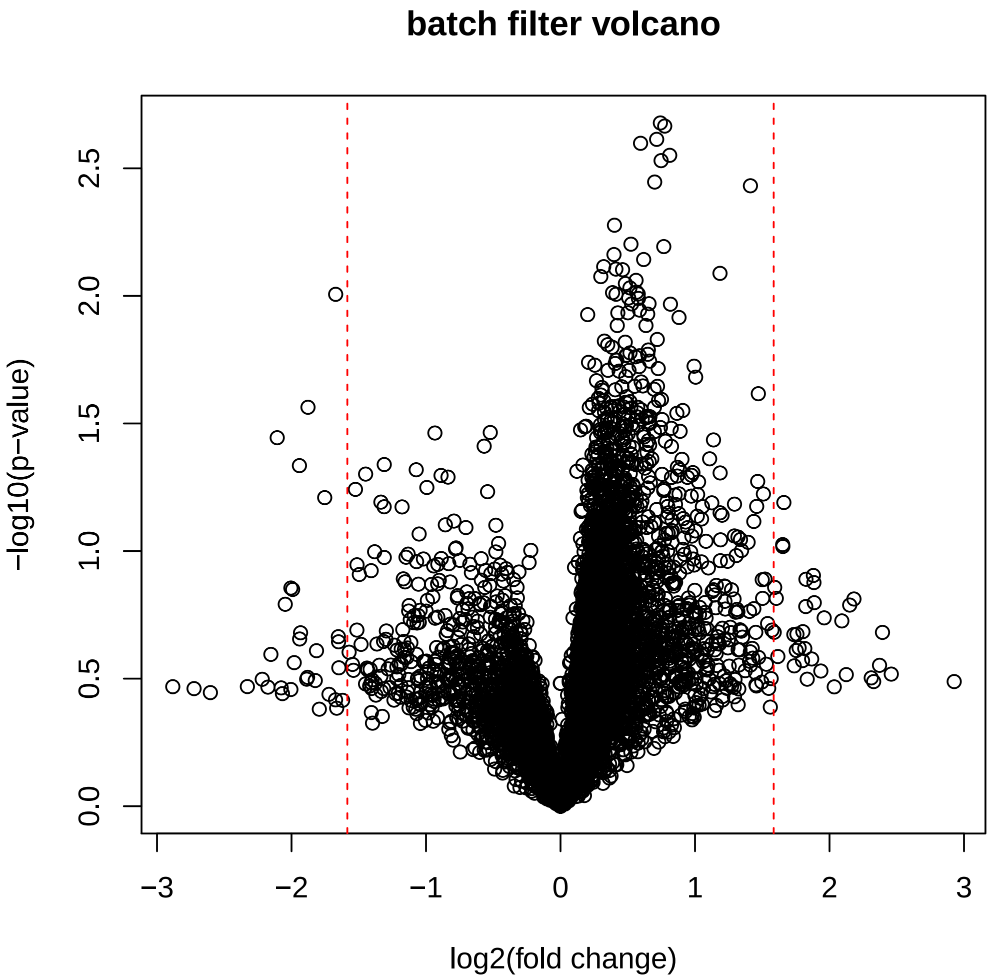
In the following code chunk, the *PAA* installation path (where exemplary data is located) is localized, the new folder 'demo_output' (where all output of the following analysis will be saved) is created, and the exemplary data set is loaded (note: exceptionally not via `loadGPR()`).

```
> cwd <- system.file(package="PAA")
> dir.create(paste(cwd, "/demo/demo_output", sep=""))
> output.path <- paste(cwd, "/demo/demo_output",  sep="")
> load(paste(cwd, "/extdata/Alzheimer.RData", sep=""))
```

3 Pre-processing

If the microarrays were manufactured or processed in lots/batches, data analysis will suffer from batch effects resulting in wrong results. Hence, the elimination of batch effects is a crucial step of data pre-processing. A simple method to remove the most obvious batch effects is to find features that are extremely differential in different batches. In *PAA* this can be done for two batches using the function `batchFilter()`. This function takes an *EList* or *EListRaw* object and the batch-specific column name vectors `lot1` and `lot2` to find differential features regarding batches/lots. For this purpose, thresholds for p-values (Student's t-test) and fold changes can be defined. To visualize the differential features a volcano plot is drawn. Finally, the differential features are removed and the remaining data is returned.

```
> lot1 <- elist$targets[elist$targets$Batch=='Batch1','ArrayID']
> lot2 <- elist$targets[elist$targets$Batch=='Batch2','ArrayID']
> elist <- batchFilter(elist=elist, lot1=lot1, lot2=lot2, p.thresh=0.001,
+ fold.thresh=3)
```



For background correction *limma*'s function `backgroundCorrect()` can be used:

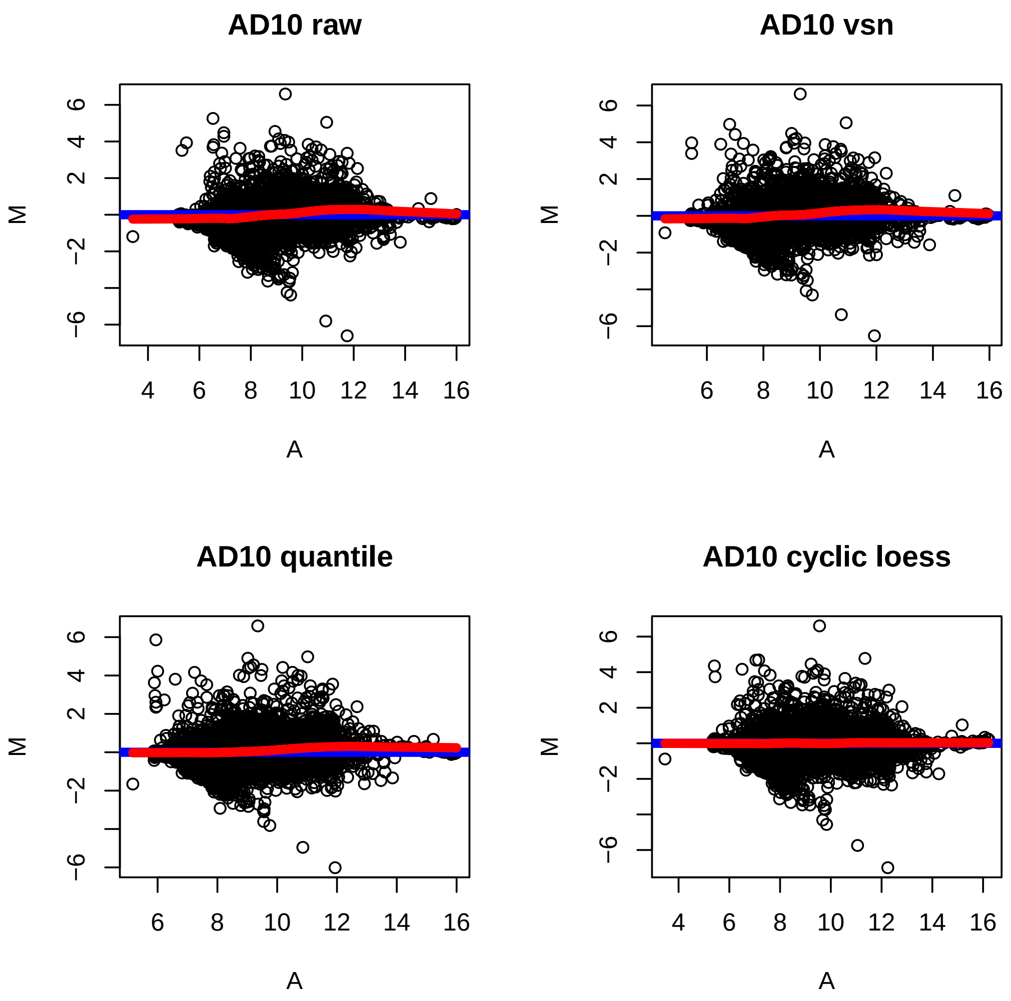
```
> library(limma)
> elist <- backgroundCorrect(elist, method="normexp",
+  normexp.method="saddle")
```

Another important step in pre-processing is normalization. To assist in choosing an appropriate normalization method for a given data set, *PAA* provides two functions: `plotNormMethods()` and `plotMAPlots()`. `plotNormMethods()` draws boxplots (one boxplot per sample) of raw data and data after all kinds of normalization provided by *PAA*. For each normalization approach sample-wise boxplots are created. All boxplots will be saved as a high-quality 'tiff' file, if an output path is specified.

```
> plotNormMethods(elist=elist)
```

`plotMAPlots()` draws MA plots of raw data and data after applying all kinds of normalization methods provided by *PAA*. If `idx="all"` and an output path is defined (default), for each microarray one 'tiff' file containing MA plots will be created. If `idx` is an integer indicating the column index of a particular sample, MA plots only for this sample will be created.

```
> plotMAPlots(elist=elist, idx=10)
```



After choosing a normalization method, the function `normalizeArrays()` can be used in order to normalize the data. `normalizeArrays()` takes an *EListRaw* object, normalizes the data, and returns an *EList* object containing normalized data in log2 scale. As normalization methods `"cyclicloess"`, `"quantile"` or `"vsn"` can be chosen. Furthermore, for *ProtoArrays* robust linear normalization (`"rlm"`, see *Sboner A. et al.* [3]) is provided.

```
> elist <- normalizeArrays(elist=elist, method="cyclicloess",
+ cyclicloess.method="fast")
```

In addition to `batchFilter()`, the function `batchAdjust()` can be used after normalization via `normalizeArrays()` to adjust the data for batch effects. This is a wrapper to *sva*'s function `ComBat()` for batch adjustment using the empirical Bayes approach [4]. To use `batchAdjust()` the targets file information of the *EList* object must contain the columns *"Batch"* and *"Group"*.

```
> elist <- batchAdjust(elist=elist, log=TRUE)
```

```
Found 2 batches
Found 1  categorical covariate(s)
Standardizing Data across genes
Fitting L/S model and finding priors
Finding parametric adjustments
Adjusting the Data
```

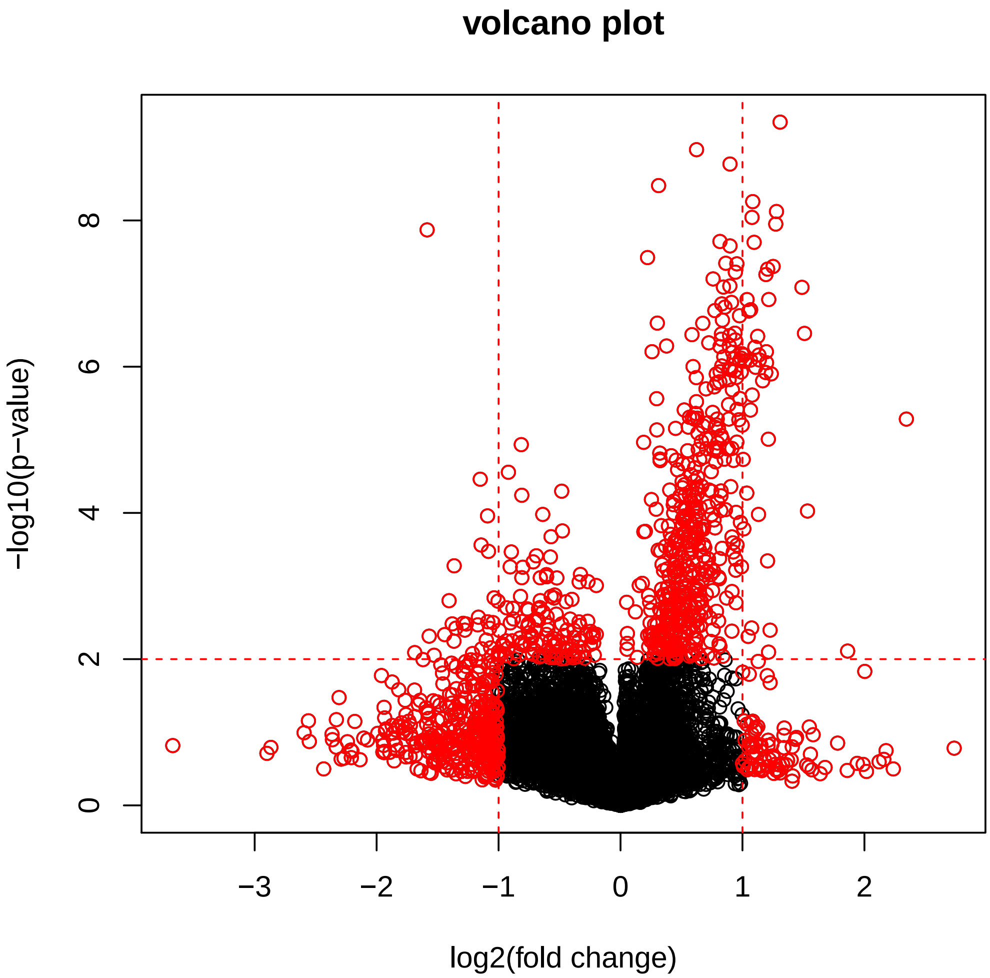
Since for further analysis also data in original scale will be needed, a copy of the *EList* object containing unlogged data should be created.

```
> elist.unlog <- elist
> elist.unlog$E <- 2^(elist$E)
```

4 Differential analysis

The goal of univariate differential analysis is to detect relevant differential features. Therefore, statistical measures such as t-test p-values or mMs as well as fold changes are considered. *PAA* provides plotting functions in order to depict the number and the quality of the differential features in the data set. Accordingly, the function `volcanoPlot()` draws a volcano plot to visualize differential features. Therefore, thresholds for p-values and fold changes can be defined. Furthermore, the p-value computation method (`"mMs"` or `"tTest"`) can be set. When an output path is defined (via `output.path`) the plot will be saved as a 'tiff' file. In the next code chunk, an example with `method="tTest"` is given.

```
> c1 <- paste(rep("AD",20), 1:20, sep="")
> c2 <- paste(rep("NDC",20), 1:20, sep="")
> volcanoPlot(elist=elist.unlog, group1=c1, group2=c2, method="tTest",
+ p.thresh=0.01, fold.thresh=2)
```

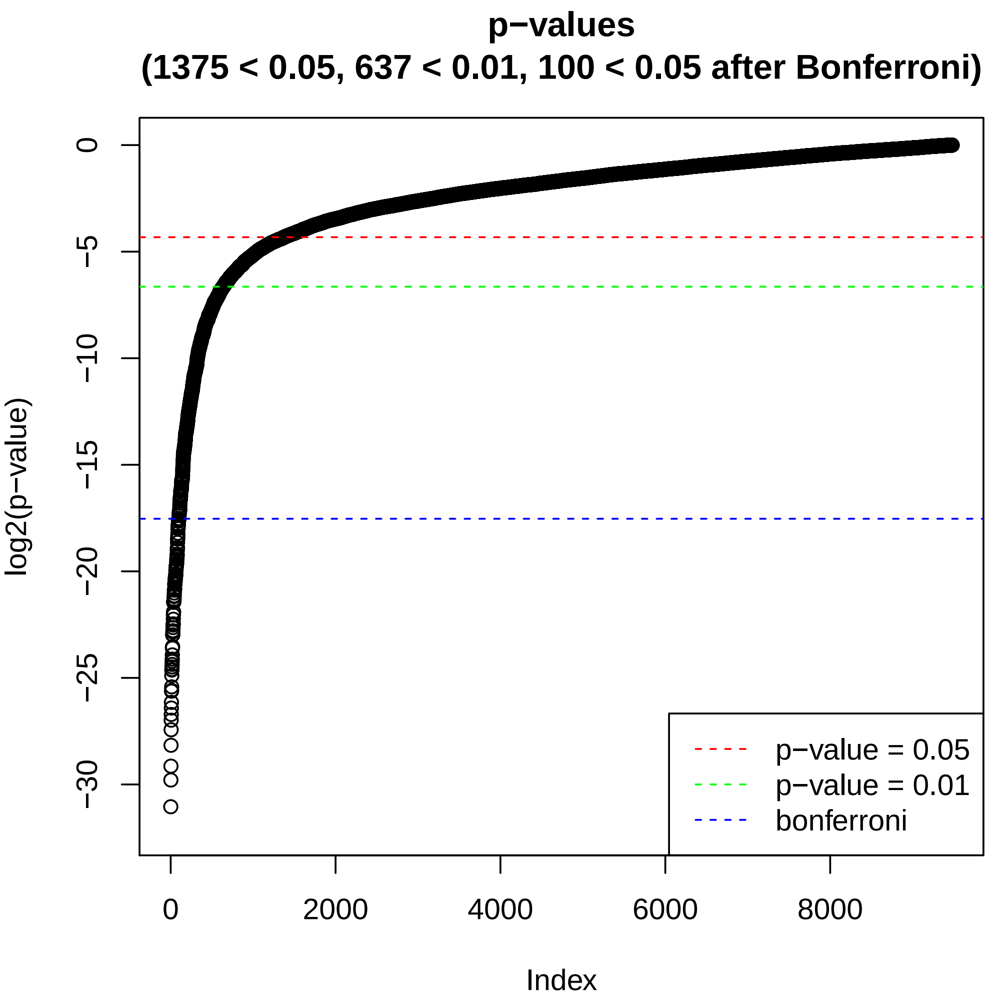


Here, an example with `method="mMs"` is given:

```
> mMs.matrix1 <- mMs.matrix2 <- mMsMatrix(x=20, y=20)
> volcanoPlot(elist=elist.unlog, group1=c1, group2=c2, method="mMs",
+ p.thresh=0.01, fold.thresh=2, mMs.matrix1=mMs.matrix1,
+ mMs.matrix2=mMs.matrix2, above=1500, between=400)
```

Another plotting function is `pvaluePlot()` which draws a plot of p-values for all features in the data set (sorted in increasing order and in log2 scale). The p-value computation method (`"tTest"` or `"mMs"`) can be set via the argument `method`. Furthermore, when `adjust=TRUE` adjusted p-values (method: Benjamini & Hochberg, 1995, computed via `p.adjust()`) will be used. For a better orientation, horizontal dashed lines indicate which p-values are smaller than 0.05 and 0.01. If `adjust=FALSE`, additionally, the respective Bonferroni significance threshold (to show p-values that would be smaller than 0.05 after a possible Bonferroni correction) for the given data is indicated by a third dashed line. *comment: Note: Bonferroni is not used for the adjustment. The dashed line is for better orientation only.* When an output path is defined (via `output.path`) the plot will be saved as a 'tiff' file. In the next code chunk, an example with `method="tTest"` is given.

```
> pvaluePlot(elist=elist.unlog, group1=c1, group2=c2, method="tTest")
```

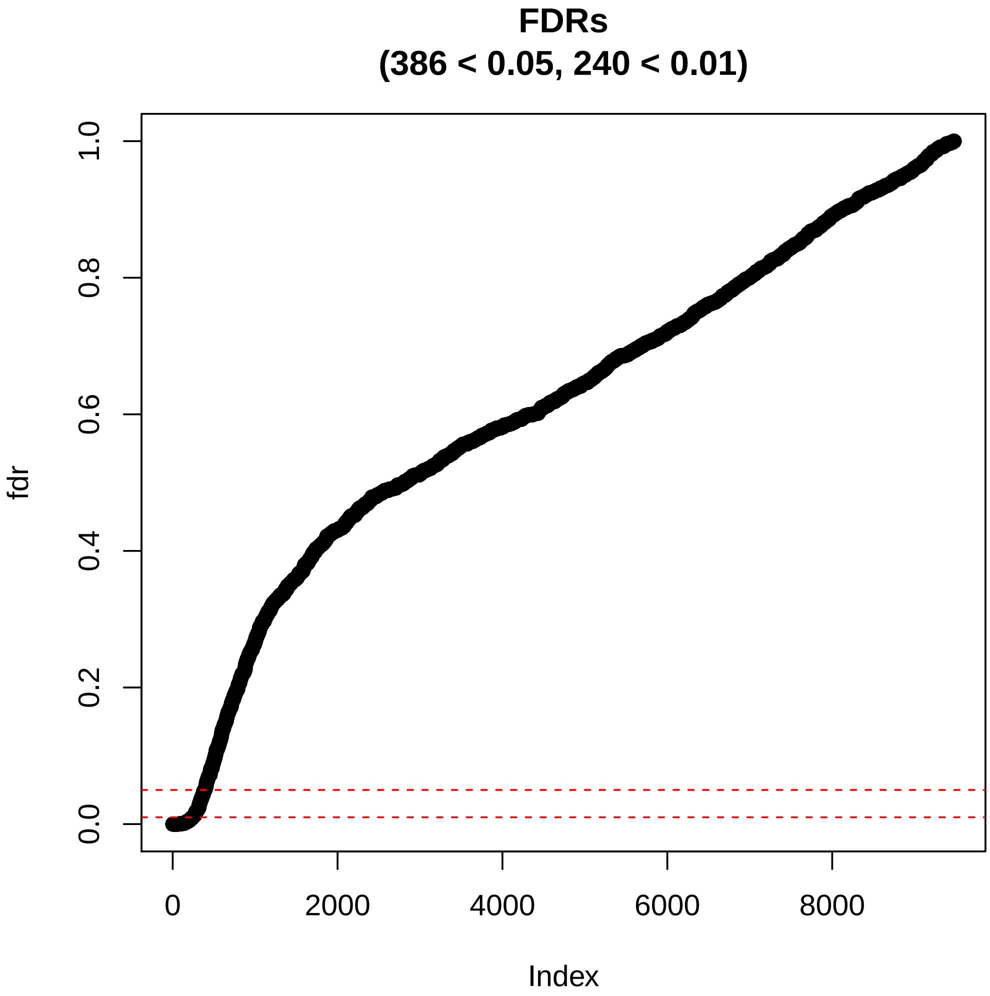


Here, an example with `method="mMs"` is given:

```
> mMs.matrix1 <- mMs.matrix2 <- mMsMatrix(x=20, y=20)
> pvaluePlot(elist=elist.unlog, group1=c1, group2=c2, method="mMs",
+ mMs.matrix1=mMs.matrix1, mMs.matrix2=mMs.matrix2, above=1500,
+ between=400)
```

Here, an example with `method="tTest"` and `adjust=TRUE` is given:

```
> pvaluePlot(elist=elist.unlog, group1=c1, group2=c2, method="tTest", adjust=TRUE)
```



Here, an example with `method="mMs"` and `adjust=TRUE` is given:

```
> mMs.matrix1 <- mMs.matrix2 <- mMsMatrix(x=20, y=20)
> pvaluePlot(elist=elist.unlog, group1=c1, group2=c2, method="mMs",
+ mMs.matrix1=mMs.matrix1, mMs.matrix2=mMs.matrix2, above=1500,
+ between=400, adjust=TRUE)
```

Finally, `diffAnalysis()` performs a detailed univariate differential analysis. This function takes an `EList$E`- or `EListRaw$E`- matrix (e.g., `temp <- elist$E`) extended by row names comprising *"BRC"*-IDs of the corresponding features. The BRC-IDs can be created via:
`brc <- paste(elist$genes[,1], elist$genes[,3], elist$genes[,2])`.
Next, the row names can be assigned as follows: `rownames(temp) <- brc`. Furthermore, the corresponding column name vectors, group labels and mMs- parameters are needed to perform the univariate differential analysis. This analysis covers inter alia p-value computation, p-value adjustment (method: Benjamini & Hochberg, 1995), and fold change computation. Since the results table is usually large, a path for saving the results should be defined via `output.path`. Optionally, a vector of row indices (`features`) and additionally (not mandatory for subset analysis) a vector of corresponding feature names (`feature.names`) can be forwarded to perform the analysis for a feature subset.

```
> E <- elist.unlog$E
> rownames(E) <- paste(elist.unlog$genes[,1], elist.unlog$genes[,3],
+     elist.unlog$genes[,2])
> write.table(x=cbind(rownames(E),E), file=paste(cwd,"/demo/demo_output/data.txt",
+     sep=""), sep="\t", eol="\n", row.names=FALSE, quote=FALSE)
> mMs.matrix1 <- mMs.matrix2 <- mMsMatrix(x=20, y=20)
> diff.analysis.results <- diffAnalysis(input=E, label1=c1, label2=c2,
+     class1="AD", class2="NDC", output.path=output.path,
+     mMs.matrix1=mMs.matrix1, mMs.matrix2=mMs.matrix2, above=1500,
+     between=400)
> print(diff.analysis.results[1:10,])
```

```
      BRC              t.test            FDR.t. min..M.stat...mMs.          FDR.mMs.
1  1 2 11   0.351983694209704 0.653973479313475  0.243589743589744 0.830359859486074
2  1 2 13   0.151259072141883 0.503022336421458 0.0241860325286354 0.330856548876571
3  1 2 15   0.319069313374231 0.632594128569213                  1                 1
4  1 2 17   0.178148370508765 0.526723457826107  0.150422391245528 0.830359859486074
5  1 2 19     0.2710374163393 0.598295359038388  0.243589743589744 0.830359859486074
6  1 2 21  0.0693618530358553 0.391110753434343 0.0457380457380457 0.483713149738174
7   1 3 1  0.0282571557303755  0.26787783632396                  1                 1
8   1 3 3 0.00910966767019863 0.140422194330867                0.5 0.908394020697585
9   1 3 5 0.00601881506586476 0.107860806851414  0.053014553014553 0.483713149738174
10  1 3 7   0.805098309573955 0.916151278840726  0.302494802494802 0.908394020697585
         fold.change          mean.AD         mean.NDC        median.AD       median.NDC
1   1.36310218942113  1387.2485761259 1017.71428942905 842.099704479654  859.41605723968
2  0.260164203455039  2189.8102223788  8417.0312183522 1306.14075983063 2551.86979248343
3   1.10246479498722 451.984655028129 409.976497284314 415.049207136659 418.503234905479
4  0.595242176244786 1520.86202090464 2555.03067759634 1215.58374522394 1690.44497083079
5  0.453628378851139 2531.33318363497 5580.19141140558 1827.95965127251 1867.42479547766
6   0.75771628269766 2637.13557152227 3480.37336895204 2249.79121136302 2928.81612007342
7   1.26296277410801 486.300802717072 385.047613980966 447.786899398197 350.215519118442
8   1.47980349935485 693.646150408715 468.742066572435  557.91164059218 456.690818828812
9   1.35894544224913 1993.46155077707 1466.91801510278  1874.0807319835 1440.05368538955
10 0.907896179874406 820.138301874487 903.339302504765 731.867870050019 470.674837625358
              sd.AD           sd.NDC
1  1646.15876708673 564.287945192397
2  2967.05315916363 18425.3711275151
3  165.941695749475 82.1672930729327
4  1062.94977081862 3156.77911820633
5  2444.53642628685 11805.2557183828
6  1276.80414608874 1559.53973583637
7  155.258168361036 123.083850273067
8  338.859052471436 93.5614899941144
9  718.813809075477 323.921664377581
10 432.900862308966 1425.95241316285
```

Subsequently, the most relevant differential features (i.e., features having low p-values and high absolute fold changes) can be extracted as a univariate feature selection. Nevertheless, it is recommended to perform also multivariate feature selection and to consider feature panels obtained from both approaches.

5 Feature pre-selection

Before multivariate feature selection will be performed, it is recommended to discard features that are obviously not differential. Discarding them will accelerate runtimes without any negative impact on results. In *PAA*, this task is called *"feature pre-selection"* and it is performed by the function `preselect()`. This function iterates all features of the data set to score them via *mMs*, *Student's t-test*, or *mRMR*. If `discard.features` is `TRUE` (default), all features that are considered as obviously not differential will be collected and returned for discarding. Which features are considered as not differential depends on the parameters `method`, `discard.threshold`, and `fold.thresh`.

- If `method = "mMs"`, features having an *mMs* value larger than `discard.threshold` (here: numeric between 0.0 and 1.0) or do not satisfy the minimal absolute fold change `fold.thresh` will be considered as not differential.

- If `method = "tTest"`, features having a p-value larger than `discard.threshold` (here: numeric between 0.0 and 1.0) or do not satisfy the minimal absolute fold change `fold.thresh` will be considered as not differential.

- If `method = "mrmr"`, *mRMR* scores for all features will be computed as scoring method (using the function `mRMR.classic()` of the *R* package *mRMRe*). Subsequently, features that are not the `discard.threshold` (here: integer indicating a number of features) features having the best *mRMR* scores are considered as not differential.

```
> mMs.matrix1 <- mMs.matrix2 <- mMsMatrix(x=20, y=20)
> pre.sel.results <- preselect(elist=elist.unlog, columns1=c1, columns2=c2,
+     label1="AD", label2="NDC", discard.threshold=0.5, fold.thresh=1.5,
+     discard.features=TRUE, mMs.above=1500, mMs.between=400,
+     mMs.matrix1=mMs.matrix1, mMs.matrix2=mMs.matrix2,
+     method="mMs")
> elist <- elist[-pre.sel.results$discard,]
```

6 Feature selection

For multivariate feature selection *PAA* provides the function `selectFeatures()`. It performs a multivariate feature selection using *"frequency-based"* feature selection (based on *RF-RFE*, *RJ-RFE* or *SVM-RFE*) or *"ensemble"* feature selection (based on *SVM-RFE*).

Frequency-based feature selection (`method="frequency"`): The whole data is splitted in k cross validation training and test set pairs. For each training set a multivariate feature selection procedure is performed. The resulting k feature subsets are tested using the corresponding test sets (via classification). As a result, `selectFeatures()` returns the average k-fold cross validation classification accuracy as well as the selected feature panel (i.e., the union set of the k particular feature subsets). As multivariate feature selection methods random forest recursive feature elimination (*RF-RFE*), random jungle recursive feature elimination (*RJ-RFE*) and support vector machine recursive feature elimination (*SVM-RFE*) are supported. To reduce running times, optionally, an additional univariate feature pre-selection can be performed (usage via `preselection.method`). As univariate pre-selection methods mMs (`"mMs"`), Student's t-test (`"tTest"`) and mRMR (`"mrmr"`) are supported. Alternatively, no pre-selection can be chosen (`"none"`). This approach is similar to the method proposed in *Baek et al.* [5].

Ensemble feature selection (`method="ensemble"`): From the whole data a previously defined number of subsamples is drawn defining pairs of training and test sets. Moreover, for each training set a previously defined number of bootstrap samples is drawn. Then, for each bootstrap sample *SVM-RFE* is performed and a feature ranking is obtained. To obtain a final ranking for a particular training set, all associated bootstrap rankings are aggregated to a single ranking. To score the `cutoff` best features, for each subsample a classification of the test set is performed (using a svm trained with the `cutoff` best features from the training set) and the classification accuracy is determined. Finally, the stability of the subsample-specific panels is assessed (via Kuncheva index, *Kuncheva LI, 2007* [6]), all subsample-specific rankings are aggregated, the top n features (defined by `cutoff`) are selected, the average classiification accuracy is computed, and all these results are returned in a list. This approach has been proposed and is described in *Abeel et al.* [7].

`selectFeatures()` takes an `EListRaw` or `EList` object, group-specific sample numbers, group labels and parameters choosing and setting up a univariate feature pre-selection method as well as a multivariate feature selection method (frequency-based or ensemble feature selection) to select a panel of differential features. When an output path is defined (via `output.path`) results will be saved on the hard disk and when `verbose` is TRUE additional information will be printed to the console. Depending on the selection method, one of two different results lists will be returned:

1. If `method` is `"frequency"`, the results list contains the following elements:
    - accuracy: average k-fold cross validation accuracy.
    - sensitivity: average k-fold cross validation sensitivity.
    - specificity: average k-fold cross validation specificity.
    - features: selected feature panel.
    - all.results: complete cross validation results.
2. If `method` is `"ensemble"`, the results list contains the following elements:
    - accuracy: average accuracy regarding all subsamples.
    - sensitivity: average sensitivity regarding all subsamples.
    - specificity: average specificity regarding all subsamples.
    - features: selected feature panel.
    - all.results: all feature ranking results.
    - stability: stability of the feature panel (i.e., Kuncheva index for the subrun-specific panels).

In the following two code chunks first *"frequency-based"* feature selection and then *"ensemble"* feature selection is demonstrated.

```
> selectFeatures.results <- selectFeatures(elist,n1=20,n2=20,label1="AD",
+     label2="NDC",selection.method="rf.rfe",subruns=2,candidate.number=1000,
+     method="frequency")
```

```
> selectFeatures.results <- selectFeatures(elist,n1=20,n2=20,label1="AD",
+     label2="NDC",selection.method="rf.rfe",subsamples=10,bootstraps=10,
+     method="ensemble")
```

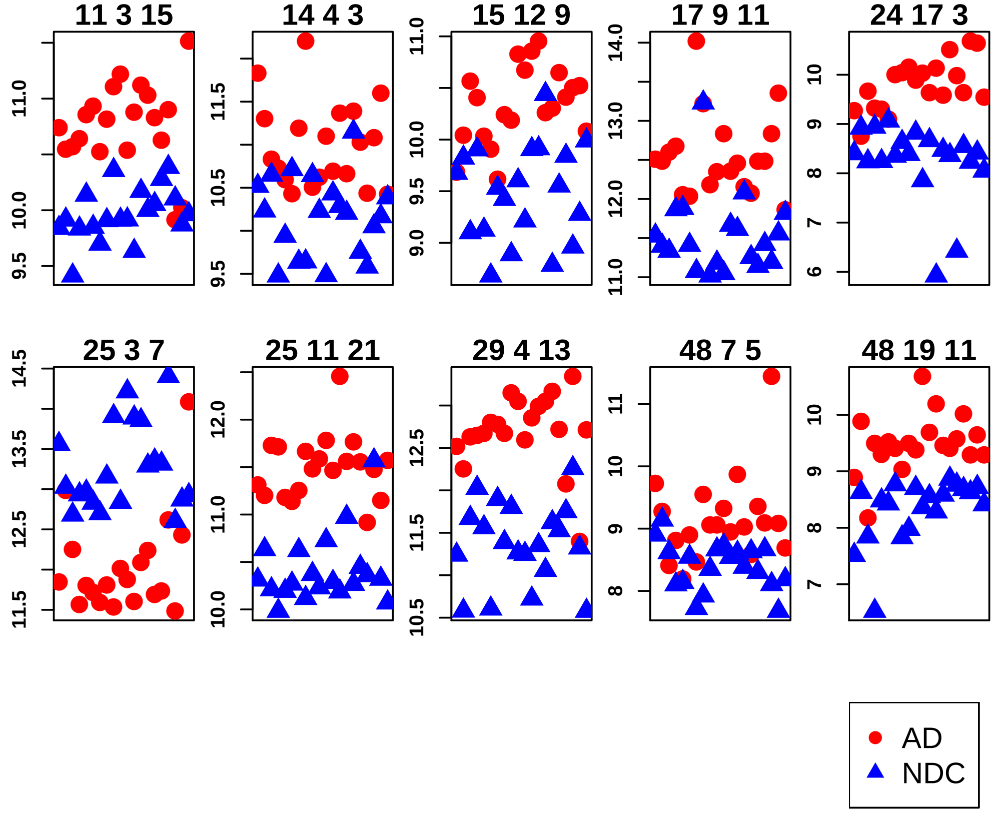
Because runtimes would take too long for this vignette *PAA* comes with pre-computated `selectFeatures.results` objects stored in '.RData' files. These objects can be loaded as follows:

```
> # results of frequency-based feature selection:
> load(paste(cwd, "/extdata/selectFeaturesResultsFreq.RData", sep=""))
> # or results of ensemble feature selection:
> load(paste(cwd, "/extdata/selectFeaturesResultsEns.RData", sep=""))
```

7 Results inspection

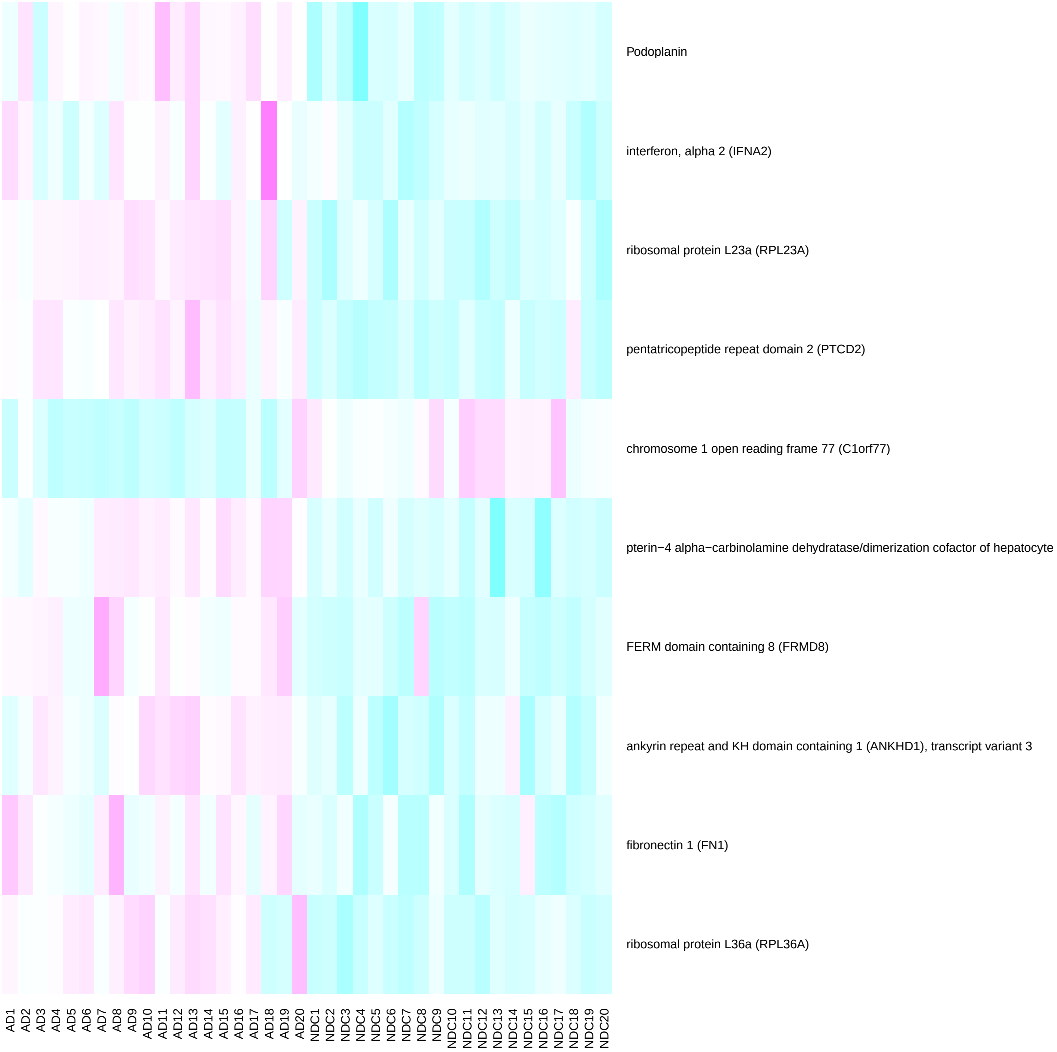
After the selection of a feature panel, these features should be validated by manual inspection and evaluation for further research. To aid results inspection, *PAA* provides several functions. The function `plotFeatures()` plots the intensities of all features (represented by BRC-IDs) that have been selected by `selectFeatures()` (one sub-plot per feature) in group-specific colors. All sub-plots are aggregated in one figure. If `output.path` is not NULL, this figure will be saved in a 'tiff' file in `output.path`.

```
> plotFeatures(features=selectFeatures.results$features, elist=elist, n1=20,
+     n2=20, group1="AD", group2="NDC")
```



Alternatively, the function `plotFeaturesHeatmap()` plots intensities of all features given in the vector `features` (represented by BRC-IDs) as a heatmap. If `description` is `TRUE` (default: `FALSE`), features will be described via protein names instead of uniprot accessions. Again, if `output.path` is not `NULL`, the heatmap will be saved as a 'tiff' file in `output.path`.

```
> plotFeaturesHeatmap(features=selectFeatures.results$features, elist=elist,
+     n1=20, n2=20, description=TRUE)
```



Finally, the function `printFeatures()` creates a table containing the selected biomarker candidate panel as well as additional information for results inspection. If `output.path` is defined, this table will be saved in a 'txt' file ('candidates.txt').

```
> printFeatures(features=selectFeatures.results$features, elist=elist.unlog)[,-2]
        BRC              AD1              AD2              AD3              AD4
1   11 3 15 1707.54872174741 1497.38674689786 1518.50548665977 1595.48520781193
2    14 4 3 3647.88566818139 2525.02827878682 1822.99945555957  1693.2902270162
3   15 12 9 821.899619465949 1053.14353763607 1517.30660391224 1358.10268235527
4   17 9 11 5841.31407989913 5741.40278219462 6210.83563636406 6537.74627659327
5   24 17 3 616.111573789642 430.507783526307 810.540240335788 640.802478533241
6    25 3 7  3684.0612493095 8097.25857845326  4873.0973765184 3029.75212058219
7  25 11 21 2540.94438380917 2354.59829803531  3390.1103019931  3353.4424765703
8   29 4 13 5874.42547230903 4891.00677037413 6347.91201462793 6413.66758809412
9    48 7 5 845.714240769082 620.144401936303 340.749562634254 449.474219299969
10 48 19 11 476.178151266904 943.905600292773  288.80435763445 720.297395448979
                AD5              AD6              AD7              AD8              AD9
1  1851.16995801815 1954.34250591408 1474.58808187167 1800.72128180473 2203.41666536497
2  1545.93055668745 1382.00035799687 2342.34218157286 4715.17557251856 1453.15299797786
3  1049.37102695368 959.981281958013 784.029298909076 1211.40462403156 1168.05152698586
```

```
4  4260.17714734055 4202.60187600583  16612.026517109 9541.51053863024 4658.85886329244
5  632.069400444692 548.763100507301 1023.42880381066 1052.79856077292 1141.40761228464
6  3567.87822120149 3361.12781744627 3083.52734553143  3582.1118182747 2964.79454811552
7  2318.92221694186 2253.41013303309 2440.77082029899 3254.85907322304 2857.61042907188
8  6536.80850077035  7127.7489779757 7039.47858075824 6525.73661581403 9104.56738330486
9  293.288590818698 478.504192919502 353.200964938416 750.950212843092 534.827154135722
10 628.686647544034  736.02264034872 677.197314077355  523.36451490189 723.686113405867
               AD10             AD11             AD12             AD13             AD14
1   2386.4640344815 1487.73326737896 1883.45531898605 2229.51382781522 2089.56380522874
2  1576.31726776973 2192.54630997844  1657.0714848759  2643.2065805943 1622.60970111714
3  1817.67145801659 1632.14697480898 1852.35247893045 1983.67962329568 1225.49775172499
4  5223.88824469703 7296.75112793942 5235.96698390632  5625.4231554349  4562.0703840908
5  948.492518532536 1046.88437502487 793.938411158226 1128.04497935666 769.658352739003
6  4130.59006500485 3768.62900951339 3117.05276658082 4362.44872185672 4826.64287027529
7  3076.07852248741 3518.44620770815 2828.55863681368 5617.81938427671 3021.46387470037
8  8470.83889644293 6197.37542856778 7401.90131165533 8122.56344668599 8489.87868607183
9  534.374903350582 640.036360803543 494.014987949289 936.872390455275 521.727313868972
10 668.009686426334 1640.47500006764 825.616986915424 1170.89751921497 702.412305275128
               AD15             AD16             AD17             AD18             AD19
1  1816.81480837384 1581.63855404132 1913.63049498383 964.097897606955  1041.3186966281
2  2680.75754761936 2088.89985475576 1386.72270539493 2163.21598235924 3100.93907875794
3     1264.90132813 1606.12803588602 1363.54480704359 1451.21232699926 1472.85585097183
4  4313.49722704221 5726.17932727699 5729.03626045874 7322.01008188909 10463.4326694858
5  1461.33398384393 1014.53577155993 796.654471494409 1643.68188498968 1600.00902244824
6  3305.55291661661 3411.90331260226 6299.56837933058 2867.60878558195 5513.99669518694
7   3482.2145207653 3007.38413310604 1935.38202306179 2846.67803253688 2274.68712494846
8  9182.19304833043 6731.10498346007  4314.2359530565 10389.9308274541 2697.28622852138
9  385.232741137473 654.526774345012 547.249319019936 2785.35671953301 543.590666369536
10 679.003582388026 763.190819480562 1035.66536310852 628.178037013298 799.090556916176
               AD20             NDC1             NDC2             NDC3             NDC4
1  2926.42411753133 916.310255562943 965.297890850339 682.054562088021 912.950704810335
2  1377.25850041096 1467.91207671002  1203.8501226154 1605.84699431214  713.29858767954
3  1084.48003919483 817.847716743106 906.146982416229 548.946861729746 957.882043011667
4  3724.16200320832 2957.78866872176 2703.03269812678 2583.57431962805 3735.99667897856
5  744.952853092939 338.540631709147 484.806316565156    302.279573333 491.625200407971
6  17381.9520086747 12078.4380584589 8356.16477086818 6570.94940407822 7820.46972808687
7  3041.02403868926 1271.70744537939 1587.81230232421 1184.55950308558 1012.47570612392
8  6708.44188234722 2418.89204876834 1531.79738605767 3265.91990969846 4176.04046007776
9  414.318514283765 478.046723028933  564.95991771095 393.118448520894 274.765462066997
10 626.652122277331 183.129536672825 396.287218246346 229.444162809801 92.2033511239309
               NDC5             NDC6             NDC7             NDC8             NDC9
1  1126.38247141383 922.659886109752 830.938986253438 961.606341894624 1312.65211553596
2  981.752307876405  1679.4377370426 796.743380621798 798.991549046085 1600.45415383023
3  558.713037627192 410.597277093105 739.751575990402  686.46640843809 473.595891923411
4  3777.33539556943 2722.26177707606 2158.43944120213 9622.39214229564 2075.93149199792
5   303.11706552559 536.157510885367 326.678547811936 396.881767196437 334.843022317674
6  8007.97121254096  7280.3648822549 6646.32014204791 9120.48431398314 15347.6266865224
7  1171.43537550505 1234.76041510299 1577.46172386673 1112.00298493664 1325.10396521861
8  3025.96547287932 1557.58885760089 3812.53215134227 2681.25160743592 3572.53949683196
9  282.289985043169 373.757162655128 211.309445291505 242.786477581689 326.622127353144
10 357.247510556275 344.052731115064 435.198927146948 227.203421006674 252.122607579667
              NDC10            NDC11            NDC12            NDC13            NDC14
1  964.547283576607 965.925455596193  794.50575819912 1153.04044563743 1024.41205626166
2  1199.79776944229 714.828087360538  1381.1081734664 1247.85077230902 1181.44529164994
```

```
3  778.128372428111 594.239115574482 959.255211480257 964.787743665106 1388.66379613292
4  2326.72562790659 2119.09515194109 3252.01291938107 3134.98527115822 4336.07141691116
5  452.275401873753 231.547622778681 406.947836547413 60.6294493022838 354.768459059881
6  7335.85579632345 19002.2930164706 15171.4608698065 14820.5186810995 10027.6534437619
7   1201.2468321564 1695.59724444137  1250.2503997022 1165.90626401451 2011.78693289676
8  2463.80654651343 2434.45300087441 1689.94804150111 2615.46312616177   2135.879656365
9  405.535678033701 423.906146123596 373.155329687154 390.503241154264  334.16573913948
10 416.622316198428 328.548938140964 376.120270496834 311.854059551955  383.56958236256
              NDC15            NDC16            NDC17            NDC18            NDC19
1  1062.74220857402 1241.54257236707 1338.10713313902 1100.43290991954 936.367518922351
2  2271.31764943593 862.867035440617 766.574994407203 1060.18779683921 1149.13869569307
3  440.992612465669 751.178591098328 917.357004718087 498.687370891938 621.105094349133
4  2443.16333785854 2258.87688670425 2743.14047789461 2340.83240144534 3014.32852730183
5  329.753797565592 85.9622199864307 374.766024317675 300.196996312642 341.950351299535
6  10473.5885640035 10195.7433739506 21655.7767725149 6220.93753625089 7480.52915420352
7  1232.22526054351 1395.71156412759 1313.52593772082 3038.53221162451 1279.57569704464
8  3154.40516192703 2952.69278321962 3448.73707209159 4903.95735927491 2564.88063392483
9  397.001712962493 315.579611644118 405.872950164318 275.519666925895 204.822065149205
10 466.530753677284 433.525387889441 410.710097186053 394.782278488945 421.419441238489
              NDC20
1  997.772012656477
2  1338.41875267022
3  1016.73416943207
4  3615.51895537293
5   265.14717496664
6  7771.28134001044
7   1075.7820418653
8  1529.55534039402
9  291.125728322968
10 340.338492661384
```

References

[1] Love B: The Analysis of Protein Arrays. In: Functional Protein Microarrays in Drug Discovery. CRC Press; 2007: 381-402.

[2] Nagele E, Han M, Demarshall C, Belinka B, Nagele R (2011): Diagnosis of Alzheimer's disease based on disease-specific autoantibody profiles in human sera. PLoS One 6: e23112.

[3] Sboner A. et al., Robust-linear-model normalization to reduce technical variability in functional protein microarrays. J Proteome Res 2009, 8(12):5451-5464.

[4] Johnson WE, Li C, and Rabinovic A (2007) Adjusting batch effects in microarray expression data using empirical Bayes methods. Biostatistics 8:118-27.

[5] Baek S, Tsai CA, Chen JJ.: Development of biomarker classifiers from high- dimensional data. Brief Bioinform. 2009 Sep;10(5):537-46.

[6] Kuncheva, LI: A stability index for feature selection. Proceedings of the IASTED International Conference on Artificial Intelligence and Applications. February 12-14, 2007. Pages: 390-395.

[7] Abeel T, Helleputte T, Van de Peer Y, Dupont P, Saeys Y: Robust biomarker identification for cancer diagnosis with ensemble feature selection methods. Bioinformatics. 2010 Feb 1;26(3):392-8.