Bioconductor RankProd Package Vignette

Fangxin Hong, fhong@salk.edu
Francesco Del Carratore, francescodc87@gmail.com
Ben Wittner, wittner.ben@mgh.harvard.edu
Rainer Breitling, rainer.breitling@manchester.ac.uk
Andris Janckevics, andris.jankevics@gmail.com

October 30, 2018

Contents

1 Introduction

The RankProd package contains all the functions needed to apply the Rank Product (RP) and the Rank Sum (RS) methods (Breitling et al., 2004, *FEBS Letters* **573**:83) to omics datasets.
Both methods are a non-parametric statistical tests, derived from biological reasoning, able to detect variables (e.g. genes or metabolites) that are consistently upregulated (or downregulated) in a number of replicated experiments. In contrast to alternative approaches, both RP and RS are based on relatively weak assumptions:

1. only a minority of all the features measured are upregulated (or downregulated);
2. the measurements are independent between replicate experiments;
3. most of the changes are independent with each other;
4. measurement variance is about equal for all measurements.

While the first three are biologically reasonable assumptions, the latter cannot be considered always true, especially when dealing with metabolomics datasets. For this reason data preprocessing through the use of a variance stabilization method is often required.

Brief description of the RP method. Suppose we have a differential expression data for a total of N genes in K replicated experiments. Let $r_{i,j}$ be the position of the i^{th} gene in the j^{th} replicate experiment in a list ordered according to fold changes (in a decreasing order if we are interested in upreguleted genes, or in a increasing order vice versa). Under the null hypothesis (no differential expressed genes are present in the dataset), the rank of a gene in the list generated, considering a single replicate, comes from an uniform distribution (i.e. $P(r_i = x) = 1/N$, where $x \in \{1, .., N\}$). Considering all the K replicates, one should notice that is extremely unlikely to find the same gene at the top of each list just by chance. In fact the exact probability of a gene being ranked 1 in each replicate is exactly $1/N^K$. The RP statistic for the i^{th} is defined as the geometric mean of all the ranks of the gene obtained in each replicate:

$$RP_i = \left(\prod_{j=1}^{K} r_{i,j} \right)^{1/K} \tag{1}$$

While the RS statistic is defined as the arithmetic mean of all the ranks:

$$RS_i = \frac{1}{K} \sum_{j=1}^{K} r_{i,j} \tag{2}$$

Genes with the smallest RP (or RS) values are the most likely to be upregulated or downregulated (according to the order we chose when ranking the fold changes). The RP is equivalent to calculating the geometric mean rank; replacing the *product* by the *sum* leads to a statistics (RS). This statistic is slightly less sensitive to outliers and puts a higher premium on consistency between the ranks in various lists. This can be useful in some applications as detailed below.
The package is able to analyse different kinds of data, such as: Affymetrix Genechip data, spotted cDNA array data (after normalization) and metabolomics data (after variance stabilization). Both

methods are also able to combine datasets derived from different origins into one analysis, increasing the power of the identification. Since the methods use the ranks of the variables in each replicated experiment (instead of the actual values), it can be flexibly applied to many different situations, such as identifying genes which are down-regulated under one condition while being up-regulated under another condition.

This guide gives a tutorial-style introduction to the main features of RankProd and to the usage of its functions. The presentation focuses on the analysis of Affymetrix array data, cDNA array data and metabolomics data obtained from mass spectrometry.

First, it is necessary to load the package.

```
> library(RankProd)
```

In the following, we use the *Arabidopsis* dataset that is contained in this package to illustrate how the RP and RS methods can be applied.

```
> data(arab)
```

`data(arab)` consists of a 500 $\times$ 10 matrix `arab` containing the expression levels of 500 genes in 10 samples, a vector `arab.cl` containing the class labels of the 10 samples, a vector `arab.origin` containing the origin labels of the 10 samples (data were produced at two different laboratories), and a vector `arab.gnames` containing the names(AffyID) of the 500 genes. The dataset is normalized by `RMA`, thus it is in *log2* scale.

2 Required arguments

In order to run a RP analysis, users need to call either the function `RankProducts` or `RP.advance` (is it also possible to use the two functions `RP` and `RPadvance` which have been kept in the package for backward compatibility). `RankProducts` is a simpler version, which is specialized in handling data sets from a single origin, while `RP.advance` is able to analyse data with single or multiple origins, and also perform some advanced analysis. There are two required arguments for the function `RanksProducts`: `data` and `cl`, which are identical to those required by the function `SAM` contained in the package `siggenes`. The first required argument, `data`, is the matrix (or data frame) containing the gene expression data that should be analysed. Each of its rows corresponds to a gene, and each column corresponds to a sample, which would be obtained, for example, by

```
> Dilution <- ReadAffy()
> data<-exprs(rma(Dilution))
```

The second required argument, `cl`, is a vector of length `ncol(data)` containing the class labels of the samples. In a RP analysis for a datasets containing samples from different origins, there is one more required argument in the function `RP.advance`: `origin`, which is a vector of length `ncol(data)` containing the origin labels of the samples.

One class data. In the one class case, `cl` is expected to be a vector of length `n` containing only 1's, where `n` denotes the number of samples. A label value other than 1 would also be accepted. In the latter case, this value is automatically set to 1. So for n=5, the vector `cl` is given by

```
> n <- 5
> cl <- rep(1,5)
> cl

[1] 1 1 1 1 1
```

Note: for one class data, we usually refer it as the expression ratio of two channels. In the outputs from the package, we call the channel used as the numerator as class 1 and the channel used as denominator as class 2.

Two class data. In this case, the function expects a vector `cl` consisting only of 0's and 1's, where all the samples with class label '0' belong to the first group (e.g. the control group) and the samples with class label '1' belong to the second group (e.g. the treatment group). For example, the first `n1=5` columns belong to the first group, and the next `n2=4` columns belong to the second group, the `cl` is given by

```
> n1 <- 5
> n2 <- 4
> cl <- rep(c(0,1),c(n1,n2))
> cl

[1] 0 0 0 0 0 1 1 1 1
```

Identically to the behaviour of the `SAM` analysis, the function also accepts others values. In that case, the smaller value is set to 0 to be the first class and the larger value to 1 as the second class.

Single origin: If the data were generated under identical or very similar conditions except the factor of interest (e.g. control and treatment), it is considered to be data with a single origin. This is the most common case of array analysis. In this case, the function `RP.advance` (and RPadvance) expects a vector `origin` of length `n` with only 1's. For example, for 9 samples generated at one time in one laboratories, the first 5 columns in the data are class 1, and the next 4 are class 2, the `cl` and `origin` are given by

```
> n1 <- 5
> n2 <- 4
> cl <- rep(c(0,1),c(n1,n2))
> cl
```

```
[1] 0 0 0 0 0 1 1 1 1
```

```
> origin <- rep(1, n1+n2)
> origin
```

```
[1] 1 1 1 1 1 1 1 1 1
```

If 9 samples are from one class, the `cl` and `origin` vectors are given by:

```
> n <- 9
> cl <- rep(1,n)
> cl
```

```
[1] 1 1 1 1 1 1 1 1 1
```

```
> origin <- rep(1, n)
> origin
```

```
[1] 1 1 1 1 1 1 1 1 1
```

Multiple origins: Sometimes happens that different laboratories conduct a very similar experiment to study the effect of the same treatment (e.g. application of a certain drug). Datasets generated by different laboratories are considered as data with different origin, as it is known that the resulting data are not directly comparable. The RP can combine these datasets together to perform an overall analysis. In this case, the vector `origin` should consist numbers 1 to L, where L is the number of different origins. For example, if 3 laboratories performed the same study using respectively 6, 4 and 8 samples, the `origin` vector is given by

```
> origin <- c(rep(1, 6), rep(2,4), rep(3,8))
> origin
```

```
 [1] 1 1 1 1 1 1 2 2 2 2 3 3 3 3 3 3 3 3
```

The function also accepts other values in the origin labels. In that case, samples with the same origin label will be treated as having the same origin.

Example: For the dataset `arab` which is included in the package, 6 samples are from laboratory 1, and another 4 are from laboratory 2. Both laboratories compare wild type *Arabidopsis* plants with and without treatment (i.e. brassinosteroid).

```
> colnames(arab)
```

```
 [1] "Chory_mock_1" "Chory_mock_2" "Chory_mock_3" "Chory_BL_1"   "Chory_BL_2"
 [6] "Chory_BL_3"   "Goda_mock_1"  "Goda_mock_2"  "Goda_BL_1"    "Goda_BL_2"
```

```
> arab.cl

 [1] 0 0 0 1 1 1 0 0 1 1

> arab.origin

 [1] 1 1 1 1 1 1 2 2 2 2
```

3 Identification of differentially expressed genes – Affymetrix array

In this section, we show how the RP method can be applied to the dataset `arab`. One should notice that RP identifies differentially expressed genes in two separate lists, up- and down-regulated genes separately. For each variable, a pfp (percentage of false prediction) is computed and used to select the differentially expressed variables. Alternatively, the p-values estimated by the function can be used with the same purpose after a multiple test correction is performed (e.g. Benjamini-Hochberg).

3.1 Data with single origin

Here, we perform the analysis for the samples from the same origin. A subset data matrix is extracted by selecting columns whose origin label is 1.

```
> arab.sub <- arab[,which(arab.origin==1)]
> arab.cl.sub <- arab.cl[which(arab.origin==1)]
> arab.origin.sub <- arab.origin[which(arab.origin==1)]
```

The RP analysis for single-origin data can be performed by either `RankProducts` or `RP.advance`(and also by the two functions kept for backward compatibility `RP` and `RPadvance`). Initially, we use the function `RankProducts` to look for differentially expressed genes between class 2 (class label=1)and class 1 (class label=0).

```
> RP.out <- RankProducts(arab.sub,arab.cl.sub, logged=TRUE,
+ na.rm=FALSE,plot=FALSE,  rand=123)

Rank Product analysis for unpaired case



 done
```

Data in `arab` are already log-transformed, otherwise one should set `logged=FALSE`. The argument `plot=FALSE` will prevent the graphical display of the estimated pfp *vs.* number of identified genes. The argument `rand` sets the random seed number to `123` allowing the function to produce reproducible results. Since some of the function parameters have a default value, we can use this function by simply typing:

```
> RP.out <- RankProducts(arab.sub,arab.cl.sub,gene.names=arab.gnames,rand=123)
```

The same results could also be obtained by

```
> RP.out <- RP.advance(arab.sub, arab.cl.sub, arab.origin.sub,
+     logged = TRUE, na.rm = FALSE, gene.names = arab.gnames, plot = FALSE,
+     rand = 123)
```

or

```
> RP.out=RP.advance(arab.sub,arab.cl.sub,arab.origin.sub,gene.names=arab.gnames,rand=123)
```

or

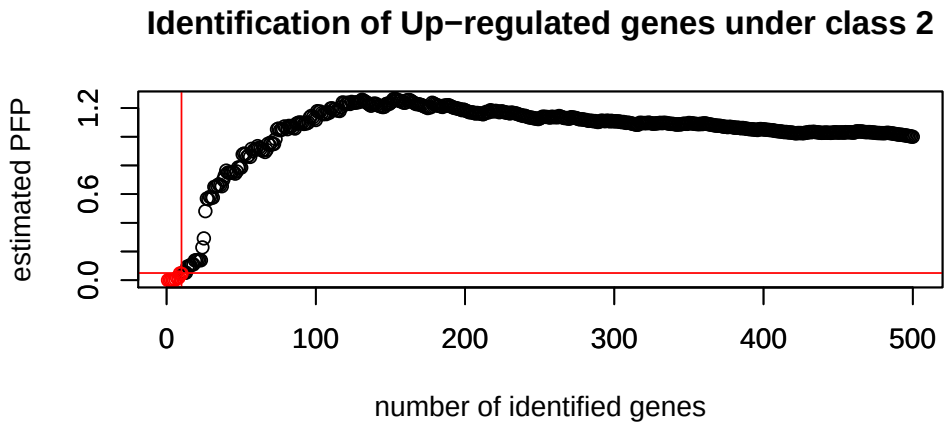
```
> RP.out <- RP(arab.sub,arab.cl.sub,gene.names=arab.gnames,rand=123)
```

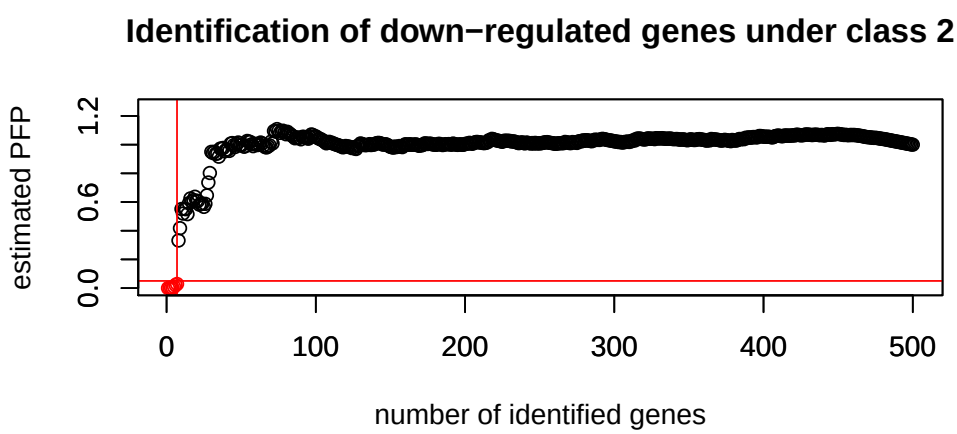
or

```
> RP.out=RPadvance(arab.sub,arab.cl.sub,arab.origin.sub,gene.names=arab.gnames,
+                  rand=123)
```

The function `plotRP` can be used to plot a graphical display of the estimated pfp *vs.* number of identified genes using the output from `RankProducts` or `RP.advance` (also for `RP` and `RPadvance`). If `cutoff` (the maximum accepted pfp) is specified, identified genes are marked in red (see figure 1). Note that the estimated pfps are not necessarily smaller than 1, but they will converge to 1. Two plots will be generated on current graphic display, for identification of up- and down-regulated genes under class 2, respectively.

```
> plotRP(RP.out, cutoff=0.05)
```





The function `topGene` generates a table of the identified genes based on user-specified selection criteria. One of the required argument is the output object from `RankProducts` or `RP.advance` (also for `RP` and `RPadvance`). The user also needs to specify either the `cutoff` (the pfp or p value threshold) or `num.gene` (the number of top genes identified), otherwise a error message will be printed and the function will stop. If `cutoff` is specified, the function also requests user to select either `pfp` (percentage of false prediction) or `pval` (p value) which is used to select genes. `pfp` is the default setting.

```
> topGene(RP.out,gene.names=arab.gnames)
Error in topGene(RP.out, gene.names = arab.gnames) :
No selection criteria is input, please input either cutoff or num.gene
```

```
> topGene(RP.out,cutoff=0.05,method="pfp",
+ logged=TRUE,logbase=2,gene.names=arab.gnames)
```

```
Table1: Genes called significant under class1 < class2

Table2: Genes called significant under class1 > class2

$Table1
          gene.index RP/Rsum FC:(class1/class2)       pfp   P.value
245244_at        344   1.587             0.4327 4.875e-05 9.749e-08
245336_at        436   2.884             0.4773 4.159e-04 1.663e-06
245119_at        219   3.107             0.4783 3.840e-04 2.304e-06
245304_at        404   3.302             0.5011 3.746e-04 2.996e-06
245176_at        276   3.684             0.5038 4.776e-04 4.776e-06
245196_at        296   7.114             0.6035 5.636e-03 6.763e-05
245254_at        354   9.967             0.6469 1.652e-02 2.312e-04
245262_at        362  10.670             0.6667 1.830e-02 2.928e-04
245141_at        241  14.720             0.6971 4.734e-02 8.520e-04
245334_at        434  14.890             0.6994 4.415e-02 8.830e-04

$Table2
          gene.index RP/Rsum FC:(class1/class2)       pfp   P.value
245362_at        462   1.000              2.594 0.0000040 8.000e-09
245136_at        236   3.175              1.718 0.0006324 2.530e-06
245277_at        377   4.762              1.564 0.0023000 1.380e-05
245296_at        396   5.130              1.550 0.0023290 1.863e-05
245276_at        376   7.830              1.479 0.0097900 9.790e-05
245229_at        329  10.320              1.458 0.0217500 2.610e-04
245075_at        175  11.740              1.394 0.0288600 4.040e-04
```

Here the user can choose variables shown by controlling $pfp < 0.05$. If `gene.names` is provided the output will also show the names of the selected genes. Since data set `arab` is in log based 2 scale, we specified `logged=TRUE` and `logbase=2`, which are the default values.

The output consists of two tables, listing selected up- (Table1: class 1 < class 2) and down- (Table2: class 1 > class 2) regulated genes. In the tables, there are 5 columns, the first one `gene.index` contains the gene indexes; the second `RP/Rsum` contains the computed Rank Product (or RS) statistics; the third `FC:(class1/class2)` contains the computed fold change of the average expression levels under two conditions, which would be converted to the original scale using input `logbase` (default value is 2) if `logged=TRUE` is specified; the fourth `pfp` contains the estimated `pfp` value for each gene in the list; the last `P.value` contained the estimated P-values for each gene. If the user wants to use a less stringent criterion, a cutoff on the p-value ($pvalue < 0.05$) can be specified as:

```
> topGene(RP.out,cutoff=0.05,method="pval",logged=TRUE,logbase=2,
```

```
+          gene.names=arab.gnames)
```

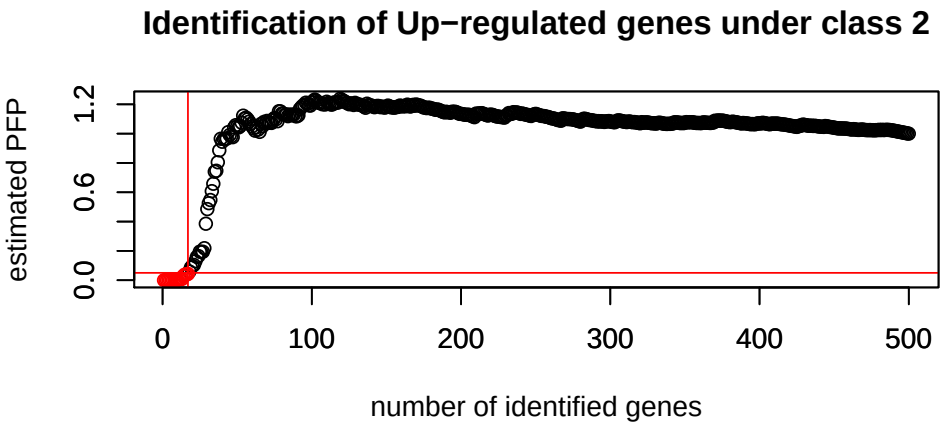
If the user is interested in the top 50 genes, he/she can type

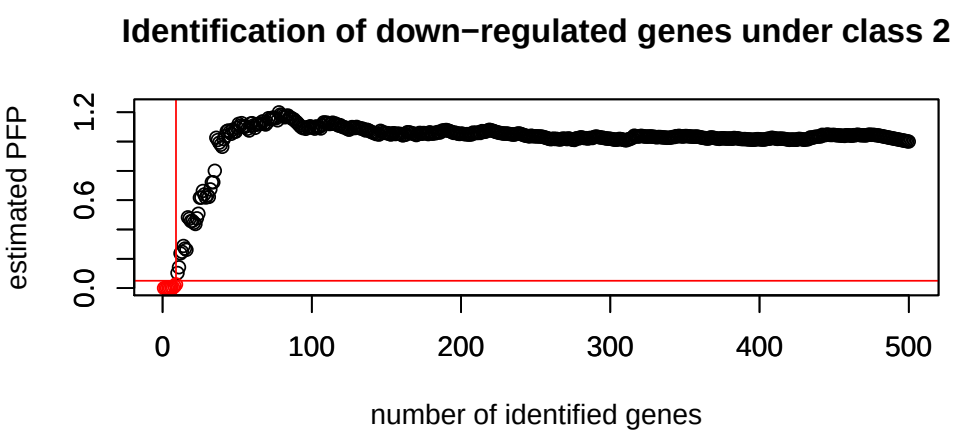
```
> topGene(RP.out,num.gene=50,gene.names=arab.gnames)
```

3.2 Data with multiple origins

In this section, we will illustrate how the RP method can be applied to datasets containing samples from multiple origins using the built-in data set `arab`. As mentioned before, `arab` consists of array data measured by two different laboratories. Both laboratories measured gene expression under the same two conditions.
Given the lack of experimental standards for microarray experiments, direct comparison is not feasible. Instead of using actual expression data, our approach combines the gene rank from different origins together (for details refer to Breitling et al. (2004)).

```
> ##identify differentially expressed  genes
> RP.adv.out <-  RP.advance(arab,arab.cl,arab.origin,
+ logged=TRUE,gene.names=arab.gnames,rand=123)

 The data is from  2 different origins

Rank Product analysis for two-class case

Rank Product analysis for unpaired case

Rank Product analysis for unpaired case


 done

> #The last command can also be written using the old syntax
> #RP.adv.out <-  RPadvance(arab,arab.cl,arab.origin,
> #logged=TRUE,gene.names=arab.gnames,rand=123)

> plotRP(RP.adv.out, cutoff=0.05)
```





By combining data from different origins, the power of the statistical test increases leading to an higher number of selected genes, as shown in the following table.

```
> topGene(RP.adv.out,cutoff=0.05,method="pfp",logged=TRUE,logbase=2,
+         gene.names=arab.gnames)

Table1: Genes called significant under class1 < class2

Table2: Genes called significant under class1 > class2

$Table1
          gene.index RP/Rsum FC:(class1/class2)       pfp   P.value
245244_at        344   3.493             0.4968 1.469e-06 2.938e-09
245336_at        436   3.845             0.4860 1.483e-06 5.930e-09
```

```
245176_at        276   7.082              0.5630 6.100e-05 3.660e-07
245304_at        404  10.750              0.6286 5.684e-04 4.547e-06
245334_at        434  11.440              0.6085 6.480e-04 6.480e-06
245262_at        362  11.730              0.6292 6.227e-04 7.472e-06
245119_at        219  13.580              0.6609 1.207e-03 1.690e-05
245329_at        429  14.680              0.5018 1.614e-03 2.582e-05
245196_at        296  16.150              0.6775 2.393e-03 4.307e-05
245265_at        365  16.830              0.5769 2.673e-03 5.346e-05
245112_at        212  20.860              0.7120 7.275e-03 1.600e-04
245193_at        293  22.080              0.6884 8.841e-03 2.122e-04
245141_at        241  22.290              0.7203 8.559e-03 2.225e-04
245254_at        354  29.290              0.7845 2.915e-02 8.162e-04
245041_at        141  31.620              0.7600 3.861e-02 1.158e-03
245383_at        483  32.020              0.7652 3.826e-02 1.224e-03
245052_at        152  33.330              0.7524 4.310e-02 1.466e-03

$Table2
          gene.index RP/Rsum FC:(class1/class2)       pfp   P.value
245362_at        462   1.320              2.507 3.512e-10 7.024e-13
245296_at        396   6.166              1.654 3.769e-05 1.508e-07
245136_at        236   6.964              1.598 5.484e-05 3.291e-07
245276_at        376   9.364              1.582 2.530e-04 2.024e-06
245277_at        377  10.270              1.480 3.494e-04 3.494e-06
245229_at        329  16.710              1.391 4.286e-03 5.143e-05
245075_at        175  17.120              1.391 4.172e-03 5.841e-05
245319_at        419  22.030              1.421 1.312e-02 2.100e-04
245307_at        407  25.970              1.400 2.584e-02 4.651e-04
```

4 Identification of differentially expressed genes – cDNA array

When dealing with cDNA array data, the usage of the RP method has to change since gene expressions of two conditions are measured from a single spot. Furthermore, the RP implementation will also change according to the experimental design. The two most commonly encountered experimental design are:

- **common reference design**, where two RNA samples are compared via a common reference;
- **direct two-color design**, where two RNA samples are directly compared without a common reference.

4.1 Common Reference Design

This type of analysis is very similar to the analysis of Affymetrix Genechips. As an example, we will have a look at the data `lymphoma` copied from the package `vsn`.

```
> data(lymphoma)

> pData(lymphoma)
      name    sample
1  lc7b047 reference
2  lc7b047    CLL-13
3  lc7b048 reference
4  lc7b048    CLL-13
5  lc7b069 reference
6  lc7b069    CLL-52
7  lc7b070 reference
8  lc7b070    CLL-39
9  lc7b019 reference
10 lc7b019 DLCL-0032
11 lc7b056 reference
12 lc7b056 DLCL-0024
13 lc7b057 reference
14 lc7b057 DLCL-0029
15 lc7b058 reference
16 lc7b058 DLCL-0023
```

As shown in the table, the 16 columns of the `lymphoma` object contain the red and green intensities of the 8 slides. Thus, the Ch1 intensities are in columns 1,3,...,15, while the Ch2 intensities are in columns 2,4,...,16. We can call `vsn` to normalize all of them at once.

```
> library(vsn)
> lym.vsn <- vsn(lymphoma)
> lym.exp <- exprs(lym.vsn)
```

Next, we can obtain the log-ratios for each slide by subtracting the common reference intensities from the 8 samples. After a class label vector is created, the `RankProducts` function can be called to perform a two-classes analysis.

```
> refrs <- (1:8)*2-1
> samps <- (1:8)*2
> M <- lym.exp[,samps]-lym.exp[,refrs]
> colnames(M)
```

```
[1] "CLL-13"    "CLL-13"    "CLL-52"    "CLL-39"    "DLCL-0032" "DLCL-0024"
[7] "DLCL-0029" "DLCL-0023"
```

```
> cl <- c(rep(0,4),rep(1,4))
> cl  #"CLL" is class 1, and "DLCL" is class 2
```

```
[1] 0 0 0 0 1 1 1 1
```

```
> RP.out <- RankProducts(M,cl, logged=TRUE, rand=123)
```

```
Rank Product analysis for unpaired case



 done
```

```
> #The last command can also be written using the old syntax
> # RP.out <- RP(M,cl,logged=TRUE,rand=123)
```

```
> topGene(RP.out,cutoff=0.05,logged=TRUE,logbase=exp(1))
```

Note that `vsn` normalized data is in log base e.

4.2 Direct two-color design

In this case, the gene expression ratio of the two dyes (classes) is measured for each spot. Here we consider and experiment where two wild type (class1) and two mutant mice (class2) are compared. The targets might be:

File name	Cy3	Cy5	Ratio=wt/mu
File 1	wt	mu	Cy3/Cy5
File 2	mu	wt	Cy5/Cy3
File 3	wt	mu	Cy3/Cy5
File 4	mu	wt	Cy5/Cy3

The first required argument for the the `RankProducts`, `data`, is the matrix (or data frame) containing the gene expression ratios. The rows correspond to the genes, while each column corresponds to the ratio of one chip. Since the input `data` is already log-ratios, the second required argument, `cl`, has to be a the vector of length `ncol(data)` containing only 1's. Finally, a one-class RP analysis can be performed in order to identify up- or down-regulated genes.

```
> cl=rep(1,4)
> RankProducts(data,cl, logged=TRUE, rand=123)
>#  or using the old syntax
>#  RP(data,cl, logged=TRUE, rand=123)
```

It should be noticed that for the direct two-color design, the RP will not distinguish the details of different designs as done by `limma` (for example see the special designs discussed in section 9 of the `limma` vignette including simple comparison and dye swaps). Furthermore, the differences among biological or technical replicates are not an issue in the RP analysis.

5 Identification of differentially expressed metabolites - LC/MS based metabolomics experiment

As mentioned before, our methods (especially the RS) can be successfully used to analyse metabolomics datasets. Testing biomarker selection methods on real data is problematic. In fact, we usually do not know the "True" biomarkers a priori. In order to cope with this problem, a publicly available UPLC-MS spike-in metabolomics dataset has been used (Franceschi P., et al. (2012)). This dataset has been obtained from twenty apples, ten of which have been spiked with known compounds that naturally occur in apples. The raw have been pre-processed allowing us to work with a data matrix containing the basepeaks intensities of the identified metabolites. The dataset used in this example is contained this package, but it can also be found in the `BioMark` package (Wehrens R., et al. (2011)). The dataset can be loaded as follows:

```
> data(Apples)
```

The list of the features associated to the spiked-in biomarkers is contained in the `Biom` vector:

```
> Biom
```

```
  463.09/414   227.07/418 273.08/434.1 435.13/434.1 447.09/438.3
         118          120          140          142          152
```

While the `apples.cl` vector, contains the class labels of the samples. As mentioned before, it is necessary to apply variance stabilization and normalization to the data. This can be easily done with the `vsn` funciton.

```
> library(vsn)
> apples.data.exp<-vsn(apples.data)
> apples.data.vsn<-exprs(apples.data.exp)
```

In order to put an higher premium on consistency between replicates, the RS analysis is preferred here.

```
> RS.apples<-RankProducts(apples.data.vsn, apples.cl,
+                         gene.names = rownames(apples.data.vsn),
+                         calculateProduct = FALSE, rand=123)
```

```
Rank Sum analysis for unpaired case


 done
```

As shown in previous examples, the `topGene` function can be used in order to show the variables presenting a `pfp` values smaller or equal than 0.05. It is also possible to store the indexes of the selected variables in a vector called `selected`.

```
> topGene(RS.apples,cutoff = 0.05, method = "pfp",
+         gene.names = rownames(apples.data.vsn))

Table1: Genes called significant under class1 < class2

No genes called significant under class1 > class2

$Table1
             gene.index RP/Rsum FC:(class1/class2)       pfp   P.value
227.07/418          120     4.2             0.4789 3.293e-12 1.671e-14
273.08/434.1        140    18.0             0.6633 8.535e-06 8.665e-08
435.13/434.1        142    20.9             0.6803 2.627e-05 4.000e-07
447.09/438.3        152    25.2             0.7060 1.329e-04 2.698e-06
869.21/433.6        137    37.4             0.6773 5.820e-03 1.477e-04

$Table2
NULL

> selected <- which(RS.apples$pfp[,1]<= 0.05)
```

Comparing the variables selected by out method with the list of the real biomarkers, it is easy to see that the method is able to identify 4 true biomarkers out of 5, while finding only one False positive.

```
> selected %in% Biom

[1]  TRUE FALSE  TRUE  TRUE  TRUE
```

6 Advanced usage of the package

Since the RP method uses ranks instead of actual expression to identify genes, the method can be generally used in many other cases beside the simple two-class comparison. Evaluating the RP is

equivalent to calculating the geometric mean of Rank. Replacing the *product* with the *sum* (i.e. replacing the geometric mean with the average) leads to the RS. Which is a statistic that is slightly less sensitive to outliers and puts a higher premium on consistency between the ranks in various lists. The RS analysis can be performed both by the `RankProducts` and `RP.advance` functions (the latter can also cope with the multiple origins case). In fact, setting `calculateProduct=FALSE` the functions will perform the RS instead of the RP. In order to guarantee the backward compatibility with the previous version of the package, the function `RSadvance` has been kept allowing to perform the RS analysis with the old syntax.

6.1 Identify genes with consistent down- or up-regulation upon drug-treatment

The following example has been inspired by a question posted in the BioC mailing-list. Suppose that a comparative study (control against treated) has been performed 3 different times with a different dosage of the same drug. The aim of such study is to investigate genes that are consistently up- or down- regulated by the drug when compared to controls. Regardless the difference in the drug dosage, one will expect that the genes up-regulated (down-regulated) by the drug will consistently show an high (low) rank in all studies. Treating the 3 studies as 3 different origins (as shown in section 3.2), the RS method can be successfully performed. The identified genes will be good candidates for consistent down- or up-regulation under various conditions.

6.2 Simultaneously identify genes up-regulated under one condition and down-regulated under another condition

Usually, in a microarray study that considers the responses in two different conditions, two lists of genes are identified independently:

- up-regulated genes under condition 1;
- down-regulated genes under condition 2.

Genes appearing in both lists are considered as the candidates. The rank-based method can be used to identify the desired list of genes in a single analysis. This is another advantage of the rank-based methods.
In fact, one can rank genes in ascending order under the first condition and in descending order under the second one. The two lists, can be considered together as in a 2-origin study in order to identify the candidate genes. Using the data `arab`, we now show a practical example. Suppose that we want to verify the consistency of the datasets generated in two different laboratories. Specifically, we want to look for genes that have been detected as up-regulated in class 2 at laboratory 1, but down-regulated in class 2 at laboratory 2. This can be achieved switching class labels for laboratory 2. Thus, for laboratory 2 the hypothetical class1 represents the real class2.

```
> arab.cl2 <- arab.cl
> arab.cl2[arab.cl==0 &arab.origin==2] <- 1
> arab.cl2[arab.cl==1 &arab.origin==2] <- 0
> arab.cl2
```

```
 [1] 0 0 0 1 1 1 1 1 0 0
```

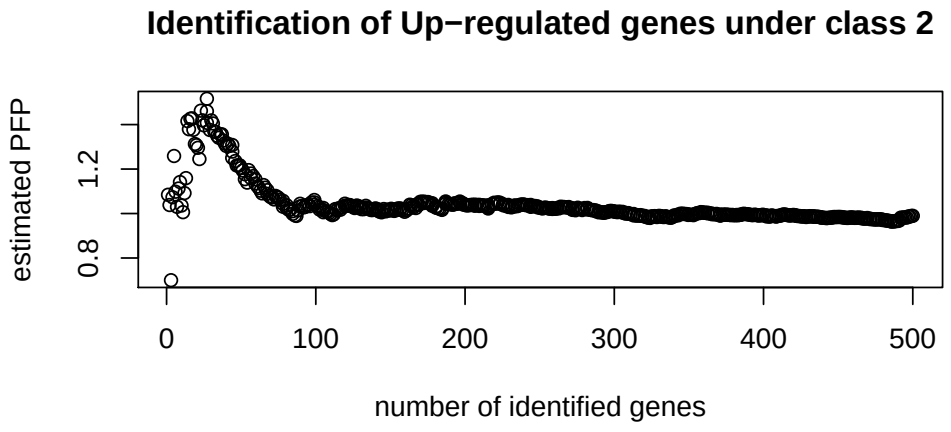
If the measurements in the two laboratories are consistent, the genes will have very different ranks in the two origins. The RS analysis is preferred here, in order to emphasise consistency for the candidate genes. In the following example, we used only the first 500 genes to perform a fast analysis.

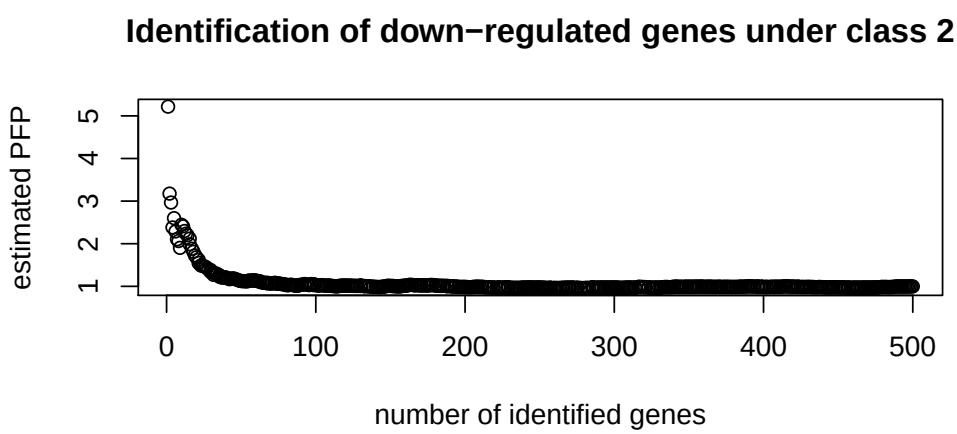
```
> Rsum.adv.out <- RP.advance(arab,arab.cl2,arab.origin,calculateProduct=FALSE,
+ logged=TRUE,gene.names=arab.gnames,rand=123)

 The data is from  2 different origins

Rank Sum analysis for two-class case

Rank Sum analysis for unpaired case

Rank Sum analysis for unpaired case


 done

> # also the old syntax can be used
> #Rsum.adv.out <- RSadvance(arab,arab.cl2,arab.origin,
> #logged=TRUE,gene.names=arab.gnames,rand=123)
> topGene(Rsum.adv.out,cutoff=0.05,gene.names=arab.gnames)

No genes called significant under class1 < class2

No genes called significant under class1 > class2

$Table1
NULL

$Table2
NULL
```

No gene was found to be differentially expressed (FDR=0.05), indicating a relative good consistency of the experiments conducted by the two laboratories. Looking at the top 10 genes in the lists, it easy to realise that they are indeed very similar.

```
> topGene(Rsum.adv.out,num.gene=10,gene.names=arab.gnames)
```

```
Table1: Genes called significant under class1 < class2

Table2: Genes called significant under class1 > class2

$Table1
            gene.index RP/Rsum FC:(class1/class2)    pfp  P.value
245392_at          492    76.8             0.8302 1.0850 0.002169
245233_at          333    87.4             0.8589 1.0380 0.004153
245305_at          405    87.6             0.8603 0.7002 0.004201
245269_at          369   101.0             0.8857 1.0730 0.008586
244951_s_at         51   109.0             0.8757 1.2590 0.012590
245381_at          481   110.0             0.8788 1.0980 0.013180
245181_at          281   112.0             0.8253 1.0300 0.014420
245380_at          480   116.8             0.8924 1.1130 0.017800
245254_at          354   120.2             0.8246 1.1420 0.020550
245082_at          182   120.4             0.8583 1.0360 0.020720

$Table2
          gene.index RP/Rsum FC:(class1/class2)   pfp P.value
245252_at        352   105.0              1.184 5.217 0.01043
245338_at        438   109.2              1.166 3.176 0.01270
245259_at        359   116.8              1.130 2.967 0.01780
245343_at        443   118.4              1.124 2.382 0.01906
245080_at        180   126.0              1.120 2.600 0.02600
244999_at         99   127.4              1.190 2.289 0.02747
245021_at        121   129.2              1.205 2.104 0.02946
244986_at         86   132.2              1.153 2.063 0.03301
245038_at        138   133.2              1.204 1.903 0.03426
245063_at        163   143.2              1.138 2.444 0.04888

> plotRP(Rsum.adv.out,cutoff=0.05)

No genes found using the input cutoff class 1 < class 2
No genes found using the input cutoff: class 1 > class 2
```





The abnormal patterns shown in figure 3 (compared with figure 1) reveal a meaningless identification. However, due to its stability, the RP statistics is still able to identify some genes.

```
> RP.adv.out <- RP.advance(arab,arab.cl2,arab.origin,calculateProduct=TRUE,
+ logged=TRUE,gene.names=arab.gnames,rand=123)

 The data is from  2 different origins

Rank Product analysis for two-class case

Rank Product analysis for unpaired case

Rank Product analysis for unpaired case
```

```
 done

> #  also the old syntax can be used
> #RP.adv.out <- RPadvance(arab,arab.cl2,arab.origin,
> #logged=TRUE,gene.names=arab.gnames,rand=123)

> topGene(RP.adv.out,cutoff=0.05,gene.names=arab.gnames)

Table1: Genes called significant under class1 < class2


Table2: Genes called significant under class1 > class2


$Table1
          gene.index RP/Rsum FC:(class1/class2)     pfp   P.value
245244_at        344   16.32             0.8710 0.02270 4.539e-05
245119_at        219   19.48             0.7238 0.02837 1.135e-04
245304_at        404   22.23             0.7973 0.03658 2.195e-04
245336_at        436   22.52             0.9822 0.02927 2.342e-04
245176_at        276   25.91             0.8950 0.04605 4.605e-04


$Table2
          gene.index RP/Rsum FC:(class1/class2)      pfp   P.value
245362_at        462      12              1.034 0.004258 8.517e-06
```

Nevertheless, the log fold-changes show that these findings are not significant. This is also confirmed by the comparison of the ranks under 13 pairings for one gene (first 9 in laboratory 1, next 4 in from laboratory 2).

```
> RP.adv.out$Orirank[[1]][344,]

 [1]   3   4   3   1   1   1   1   1   1 495 496 453 492
```

7 Changes introduced in the last version

In this section changes introduced in the new version of the package are briefly summarized.

7.1 Application to unpaired datasets

Let T and C stand for two experimental conditions (e.g. treatment versus control), while n_T and n_C are the number of replicates in the two conditions. In the old package the RP (RS) analysis for the unpaired case was performed according the ad hoc procedure decribed here:

1. all the possible $K = n_T \times n_C$ pair-wise comparisons are considered and K lists of ratios FC are evaluated;

2. the ratios are ranked within each comparison (r_{gi} is the rank of the gth gene in the ith comparison);

3. the RP for each gene is determined as $RP_g = (\prod_i r_{gi})^{1/K}$;

4. alternatively, the RS is determined as $RP_g = (\sum_i r_{gi})/K$.

Apparently, such approach leads to an increase of the False Discovery Rate. In the new package, a new and more principled method has been developed. This method is described below:

1. the number of pairs ($npairs$) is defined as the number of the samples in the smallest class;

2. if not defined by the user, the number of Random Pairings that will be generated (n_{rp}) is set to $npairs \times npairs$ (if this number is not odd $n_{rp} = npairs \times npairs + 1$);

3. Sampling from the original dataset, n_{rp} new datasets of dimension ($ngenes \times npairs$) are generated;

4. the RP (RS) is evaluated n_{rp} times considering each Random Pairing as a paired experiment;

5. per each gene, the final RP_g (or RS_g) is estimated as the median of the n_{rp} values evaluated in the step before.

7.2 Evaluation of the p-values for the RP

Instead of the permutation approach used in the old version of the package, the pvalues for the RP are now evaluated through the fast algorithm described in *Heskes et al. 2014*, which allows a very accurate approximation of the p-values in a computationally fast manner. This approach significantly speeds up the RP analysis. When considering a typical paired dataset ($N = 1000$ and $K = 10$), the computation time is now reduced by a factor of ~ 500, when compared with the analysis performed with the previous approach (using $10,000$ permutations).

7.3 Evaluation of the p-values for the RS

Also in this case, the permutation approach was abbandoned. We have developed a novel method able to compute the exact p-values for the RS in a fast manner. This method is straightforward and based on a very simple analogy. It is easy to understand that, under the null hypothesis, the probability distribution of the RS, in an experiment with N features and K replicates, is exactly the same as the probability distribution of the sum of the outcomes obtained by rolling K dice with N faces (`http://mathworld.wolfram.com/Dice.html`). The numerical error generated by our fast algorithm increases with the the size of the dataset. For this reason we developed a more accurate implementation of the same algoritm, which is able to cope with extremely large datasets. Unfortunately, this leads to an increase of the computational time. When the size of the dataset is such that the use of the accurate implementation is needed and the time needed to evaluate the exact p-values becomes unacceptable, the new package computes the exact p-values for the smallest

RS values for `tail.time` minutes. The rest of the p-values are approximated with the following gaussian:

$$\mathcal{N}(\mu = \frac{K(N+1)}{2}, \sigma^2 = \frac{K(N^2-1)}{12}) \tag{3}$$

It should be noticed that with such large datasets, this approximation is extremely accurate. Nevertheless, in this case the fuction shows the highest p-values exactly computed, so the user can play with the tail.time parameter if not satisfied. In most of the cases, this approach significantly speeds up the RS analysis. When considering a typical paired dataset ($N = 1000$ and $K = 10$), the computation time is now reduced by a factor of ~ 1200, when compared with the analysis performed with the previous approach (using $10,000$ permutations).

Reference

Breitling, R., Armengaud, P., Amtmann, A., and Herzyk, P.(2004) Rank Products: A simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments. *FEBS Letter*, 57383-92

Nemhauser JL, Mockler TC, Chory J. (2004) Interdependency of brassinosteroid and auxin signaling in Arabidopsis. *PLoS Biol.* 21460

http://arabidopsis.org/info/expression/ATGenExpress.jsp

Heskes, T.,Eisinga, R., and Breitling, R.(2014) A fast algorithm for determining bounds and accurate approximate p-values of the rank product statistic for replicate experiments. *BMC bioinformatics*, 15.1: 367.

Franceschi, P., Masuero, D., Vrhovsek, U., Mattivi, F. and Wehrens R. (2012) A benchmark spike-in data set for biomarker identification in metabolomics. *Journal of chemometrics*, 26(1-2):16-24.

Wehrens, R., Franceschi, P., Vrhovsek, U. and Mattivi, F. Stability-based biomarker selection. *Analytica chimica acta*, 705(1):15-23.