Bioconductor RankProd Package Vignette

Fangxin Hong fhong@salk.edu
Ben Wittner wittner.ben@mgh.harvard.edu

October 13, 2015

Contents

1 Introduction

The RankProd package contains functions for the analysis of gene expression microarray data, in particular the identification of differentially expressed genes. RankProd utilizes the so called rank product non-parametric method (Breitling et al., 2004, *FEBS Letters* **573**:83) to identify up-regulated or down-regulated genes under one condition against another condition, e.g. two different treatments, two different tissue types, etc.
Rank Product is a non-parametric statistic that detects items that are consistently highly ranked in a number of lists, for example genes that are consistently found among the most strongly upregulated genes in a number of replicate experiments. It is based on the assumption that under the null hypothesis that the order of all items is random the probability of finding a specific item among the top r of n items in a list is $p = \frac{r}{n}$. Multiplying these probabilities leads to the definition of the rank product $RP = \prod_i \frac{r_i}{n_i}$, where r_i is the rank of the item in the i-th list and n_i is the total number

of items in the i-th list. The smaller the RP value, the smaller the probability that the observed placement of the item at the top of the lists is due to chance. The rank product is equivalent to calculating the geometric mean rank; replacing the *product* by the *sum* leads to a statistics (average rank) that is slightly more sensitive to outlier data and puts a higher premium on consistency between the ranks in various lists. This can be useful in some applications as detailed below.
A list of up- or down-regulated genes are selected based on the estimated percentage of false positive predictions (pfp), which is also known as false discovery rate (FDR). The package is able to analyze both Affymetrix Genechip data as well as spotted cDNA array data after normalization. Another attraction of this method is its ability to combine data sets from different origins into one analysis to increase the power of the identification. In practice, this makes it possible for data sets generated at different laboratories or under different environments to be combined for study. Since the method utilizes the rank of genes in each array instead of the actual expression value, it can be flexibly applied to many different questions, 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 shows how these functions can be used. The presentation focuses on the analysis of Affymetrix array data, but the usage for cDNA array analysis will be illustrated briefly as well.

First, it is necessary to load the package.

```
> library(RankProd)
```

In the following, we use the *Arabidopsis* data set that is contained in this package to illustrate how rank product method analyses can be performed.

```
> data(arab)
```

`data(arab)` consists of a 500×10 matrix `arab` containing the expression levels of 500 genes and 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 500 genes. The data set is normalized by `RMA`, thus it is in *log2* scale.

2 Required arguments

In order to run a rank product analysis, users need to call either the function `RP` or `RPadvance`. `RP` is a simpler version which is specialized in handling data sets from a single origin, while `RPadvance` is able to analyze data with single or multiple origins, and also perform some advanced analysis. There are two required arguments for the function `RP`: `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 analyzed. 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 the vector of length `ncol(data)` containing the class labels of the samples. In a rank product analysis for data sets from different origin, there is one more required argument in the function `RPadvance`, `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 class, the function expect 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 behavior of the `SAM` analysis, the function also accepts others values than 0 and 1. 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 done. In this case, the function `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` 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: It sometimes happens that different laboratories conducted similar/same experiments to study the effect of the same treatment (e.g., application of a certain drug). Data sets generated at different laboratories are considered as data with different origins, as it is known that the resulting data are not directly comparable. Rank products can combine these data sets 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 there are 3 labs that did the same study, and used 6 samples, 4 samples and 8 samples, respectively, 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 others 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 data set `arab` which is included in the package, 6 samples are from lab 1, and another 4 are from lab 2. Both labs compare wild type *Arabidopsis* plants with and without treatment (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 rank product method can be applied to the sample data set `arab`. One should notice that rank products identify differentially expressed genes in two separate lists, up- and down-regulated genes separately. For each identification, one pfp (percentage of false prediction) is computed and used to select genes.

3.1 Data with single origin

First, we perform the analysis for data from a single 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 rank product analysis for single-origin data can be performed by either `RP` or `RPadvance`. We first use function `RP` to look for differentially expressed genes between class 2 (class lable=1)and class 1 (class lable=0).

```
> RP.out <- RP(arab.sub,arab.cl.sub, num.perm=100, logged=TRUE,
+ na.rm=FALSE,plot=FALSE,  rand=123)

Rank Product analysis for two-class case


Starting 100 permutations...
Computing pfp ..
Outputing the results ..
```

In this case, the data in `arab` are already log-transformed, otherwise one should set `logged=FALSE`. The default number of permutations is 100, this can be set to higher values by the user to obtain more precise estimates of the pfp. The argument `plot=FALSE` will prevent the graphical display of the estimated pfp *vs.* number of identified genes. The argument `rand` sets the random number seed to `123` to make the results of `RP` reproducible. Since we chose some default values in function `RP` , we would simply type

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

The same analysis could also be done by

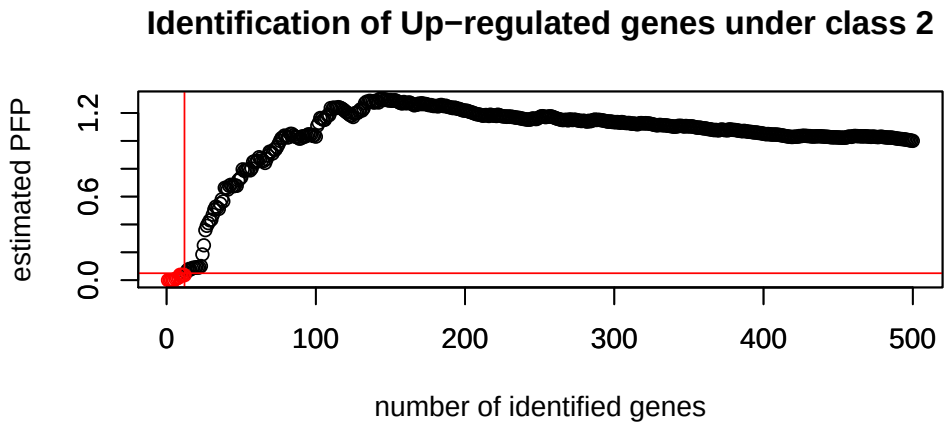
```
> RP.out <- RPadvance(arab.sub, arab.cl.sub, arab.origin.sub, num.perm = 100,
+     logged = TRUE, na.rm = FALSE, gene.names = arab.gnames, plot = FALSE,
+     rand = 123)
 The data is from  1 different origins


Rank Product analysis for two-class case


Starting  100 permutations...
Computing pfp...
```

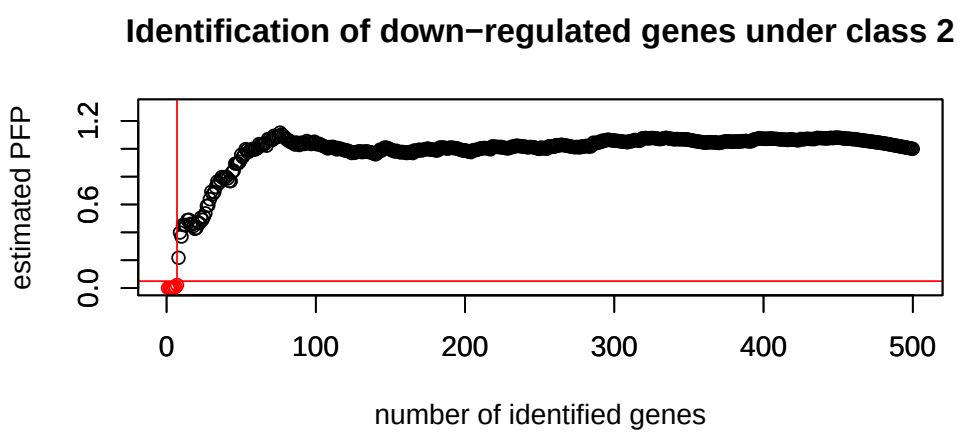
or

```
> 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 `RP` or `RPadvance` If `cutoff` (the maximum accepted pfp) is specified, identified genes are marked in red (see figure 1). Note that the estimated pfp is not necessarily smaller than 1, but will converge to 1 in the tail. 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` is used to output a table of the identified genes based on user-specified selection criteria. The required argument is the output object from function `RP` or `RPadvance`. The user also needs to specify either the `cutoff` (the desired significance of the identification) or `num.gene` (the number of top genes identified), otherwise a error message will be printed and the function will be stopped. If `cutoff` is specified, the function also requests user to select either `pfp` (percentage of false prediciton) or `pval` (pvalue) which is used to select genes. `pfp` is the default setting, which is selected when no selection is made by user.

```
> 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  1.4891             0.4327 0.0000   0e+00
245336_at        436  2.4231             0.4773 0.0000   0e+00
245119_at        219  3.0932             0.4783 0.0000   0e+00
245176_at        276  3.3955             0.5038 0.0000   0e+00
245304_at        404  3.7745             0.5011 0.0000   0e+00
245196_at        296  7.9884             0.6035 0.0100   1e-04
245254_at        354  9.4769             0.6469 0.0143   2e-04
245262_at        362 11.0043             0.6667 0.0188   3e-04
245334_at        434 14.9425             0.6994 0.0389   7e-04
245141_at        241 15.2589             0.6971 0.0380   8e-04
245265_at        365 15.7394             0.6888 0.0391   9e-04
245112_at        212 15.7589             0.7112 0.0358   9e-04

$Table2
          gene.index RP/Rsum FC:(class1/class2)    pfp P.value
245362_at        462  1.0000             2.5935 0.0000   0e+00
245136_at        236  2.8470             1.7180 0.0000   0e+00
245277_at        377  4.9437             1.5636 0.0000   0e+00
245296_at        396  5.6434             1.5505 0.0000   0e+00
245276_at        376  8.5368             1.4795 0.0060   1e-04
245229_at        329  9.1905             1.4577 0.0083   1e-04
245075_at        175 11.7001             1.3937 0.0229   3e-04
```

Here the user chose to output the identified genes by controlling $pfp < 0.05$. `gene.names` are provided and thus are output with the table of selected genes. Since data set `arab` is in log based 2 scale, we specified `logged=TRUE`, `logbase=2`, which are the default values.

Two tables are output, listing identified up- (Table1: class 1 $<$ class 2) and down- (Table2: class 1 $>$ class 2) regulated genes. There are 5 columns in the table, the first one `gene.index` is the gene index in the original data set; the second `RP/Rsum` is the computed rank product (or rank sum in section 5) for each gene; the third column `FC:(class1/class2)` is 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 4th column `pfp` is the estimated `pfp` value for each gene in the list if that gene serves as the cutoff point; the last column `P.value` is

the associated P-values for each gene. If user want to use less stringent criterion (without adjust for multiple comparison), $pvalue < 0.05$ can be specified as

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

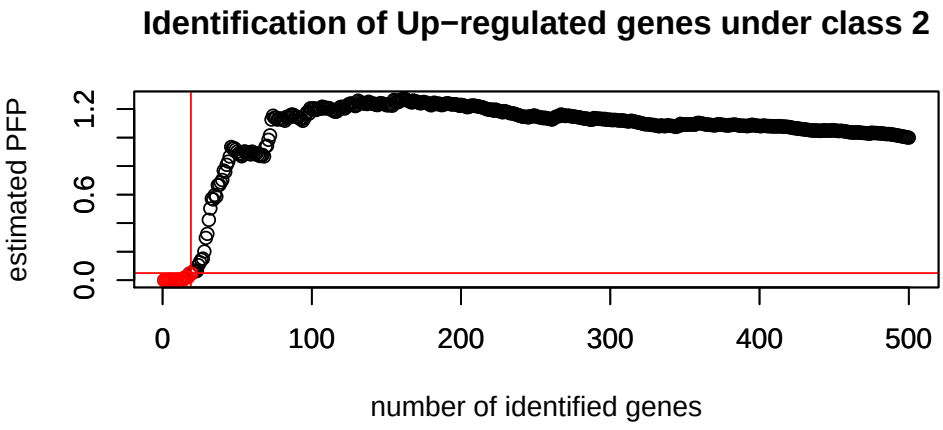
If the user is interested in the top, say, 50 genes, one can type

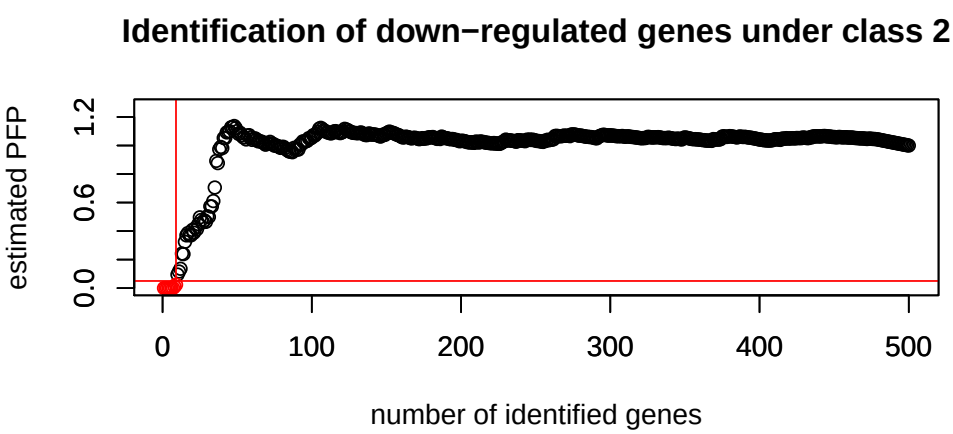
```
> topGene(RP.out,num.gene=50,gene.names=arab.gnames)
```

3.2 Data with multiple origins

In this section, we will illustrate how the rank product analysis can be applied to data sets from multiple origins using the built-in data set `arab`. As introduced above, `arab` consists of array data sets measured at two different laboratories. Both labs measured gene expression under two classes with similar condition.
The lack of experimental standards for microarray experiments leads to heterogeneous data sets for which direct comparison is not possible. Instead of using actual expression data, the present approach combines the gene rank from different origins together to select genes (for details refer to Breitling et al. (2004)).

```
> ##identify differentially expressed  genes
> RP.adv.out <-  RPadvance(arab,arab.cl,arab.origin,num.perm=100,
+ logged=TRUE,gene.names=arab.gnames,rand=123)

 The data is from  2 different origins

Rank Product analysis for two-class case

Starting  100 permutations...
Computing pfp...

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






```
##Table of identified genes  by controlling pfp (FDR)=0.05,
> topGene(RP.adv.out,cutoff=0.05,method="pfp",logged=TRUE,logbase=2,gene.names=arab.gnames)
```

Note: By combining data sets from different origins together, the test gets increased power, which leads to more identified genes.

4 Identification of differentially expressed genes – cDNA array

For cDNA array data, the usage of the rank product method is different from that for Affymetrix arrays, since gene expressions of two conditions are measured from one spot. The usage is different depending on the experimental design. Two types of design are regularly encountered: Common reference designs in which two type RNA samples are compared via a common reference, or direct two-color designs in which two types of RNA samples are directly compared without a common reference.

4.1 Common Reference Design

In this case, different RNA samples are compared with a common reference for each array. 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
```

The 16 columns of the `lymphoma` object contain the red and green intensities, respectively, from the 8 slides, as shown in the table. Thus, the Ch1 intensities are in column 1,3,...,15, the Ch2 intensities in column 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)
```

We can then obtain the log-ratios for each slide, by subtracting the common reference intensities from those for the 8 samples. A class label vector is created for these 8 samples, and function `RP` is called to perform a two-class 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 <- RP(M,cl, logged=TRUE, rand=123)
```

```
Rank Product analysis for two-class case


Starting 100 permutations...
Computing pfp ..
Outputing the results ..
```

```
> 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 order to use the rank product method, the gene expression ratio for the two dyes (classes) is calculated for each spot, ratio=expression of class 1/expression of class 2. Suppose there is an experiment in which two wild type (class 1) and two mutant mice (class 2) are compared using two arrays. 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, `data`, is the matrix (or data frame) containing the gene expression ratio that one intends to analyze. Each of its rows corresponds to a gene, and each column corresponds to the ratio of one chip. Since the input `data` is already log-ratios, the second required argument, `cl`, should be set to the vector of length `ncol(data)` with all 1's, and a one-class rank product analysis performed to identify up- or down-regulated genes .

```
> cl=rep(1,4)
> RP(data,cl, logged=TRUE, rand=123)
```

One should notice, for the direct two-color design rank products will not distinguish the details of different designs in the way `limma` does (for example see the special designs discussed in section 9 of the `limma` vignette including simple comparison and dye swaps). And the problems caused by the difference of biological replicates or technical replicate are not an issue in the rank products analysis, either.

5 Advanced usage of the package

Since the rank product 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. As what mentioned in the introduction, the rank product is equivalent to calculating the geometric mean rank which is robust to the outliers. However, replacing the *product* by the *sum* leads to a statistics (average rank) that is slightly more sensitive to individual data value and puts a higher premium on consistency between the ranks in various lists (Breitling et al., submitted). There is a function `RSadvance` in the `RankProd` library which can be used to perform a rank sum analysis. Since the rank sum method hasn't been published yet, we would only recommend it to the advanced users, and we will not guarantee the performance. The following examples show the potential usage.

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

This example was inspired by a question posted in the BioC mailing-list. There are 3 studies and 4 to 5 doses of a drug within each study (the doses are not the same in each study). Genes that are consistently up- or down- regulated by drug compared to control are of interests. Rank sum method can be potentially used towards this end by treating 3 studies as 3 origins in a multi-origin study introduced in section 3.2.

Although the drug doses are different in each study, people do expect genes with high rank in ascending order (treatment *vs.* control) across different studies to be consistently down-regulated, genes with high rank in descending order (treatment *vs.* control) to be consistently up-regulated. Since we would expect candidate genes having relative consistent high rank in all studies, we prefer to use `RSadvance` to perform a three-origin analysis treating each study as one origin. The identified genes would be good candidates for consistent down- or up-regulation under various conditions

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

Normally, when people conduct a microarray study that studies responses in two different (and opposing) conditions, two lists of genes will be identified independently: up-regulated genes under condition 1, down-regulated genes under condition 2. Genes appearing in both lists will be called as the candidate genes. However, the rank-based method can be used to identify the desired list of genes in a single analysis. In other words, one significance is controlled for the identification. This is another advantage of the rank-based method compared to other methods.

To perform this analysis, one can rank genes in ascending order under one condition and descending order under another condition, then put all ranks together as in a 2-origin study to identify the desired candidate genes. Since we would expect consistent ranks for the candidate genes under both conditions, we would prefer `RSadvance` instead of rank product. The practical application is a bit complicated, the data `arab` is again used to illustrate the usage. Suppose we want to check the consistence of the data sets generated in two different labs. For example, we would look for genes that were measured to be up-regulated in class 2 at lab 1, but down-regulated in class 2 at lab 2. In stead of changing the function to rank genes in different order under two conditions, we switched

class labels for lab 2. Thus the resulted "class 2" is the real class 2 in lab 1 and the real class 1 in lab 2, and the resulted "class 1" is the real class 1 in lab1 and the real class in lab2. We call the resulted classes as hypothetical class 1 or 2.

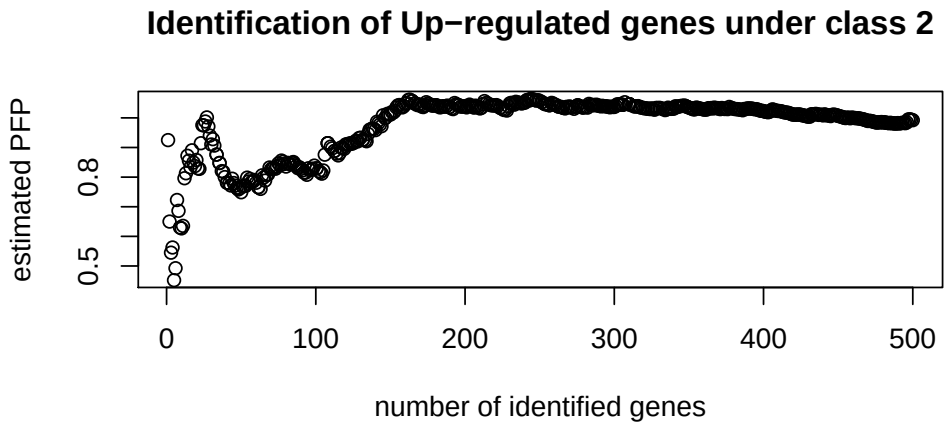
```
> arab.cl2 <- arab.cl
> arab.cl2[arab.cl==0 &arab.origin==2] <- 1
> arab.cl2[arab.cl==1 &arab.origin==2] <- 0
> arab.cl2

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

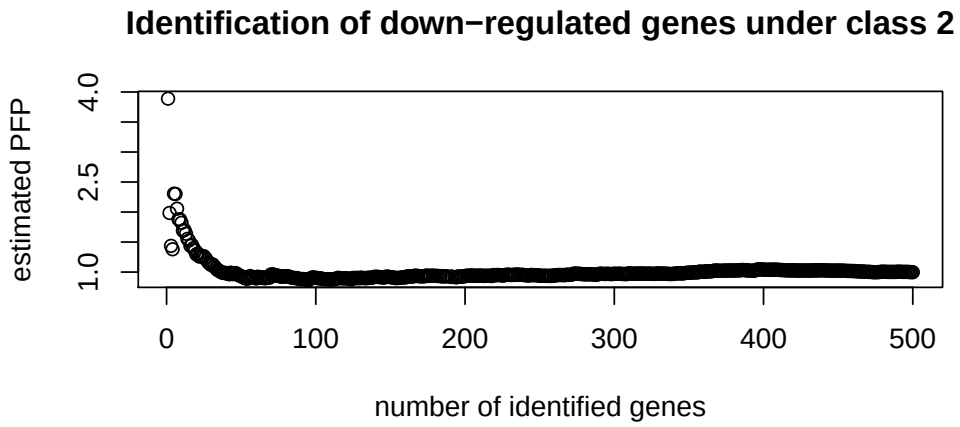
Looking for genes that are consistently up-regulated under hypothetical class 2 is equivalent as looking for genes that are up-regulated under class 2 in lab 1 and down-regulated under class 2 in lab2. Ideally, we would expect genes having very different ranks from two labs after switching class labels in lab 2, therefore a rank sum analysis is preferred to emphasis on consistency on the candidate genes. As illustration purpose, we used only first 500 genes to perform a fast analysis

```
> Rsum.adv.out <- RSadvance(arab,arab.cl2,arab.origin,num.perm=100,
+ logged=TRUE,gene.names=arab.gnames,rand=123)

 The data is from  2 different origins

Rank Sum analysis for two-class case

Starting  100 permutations...
Computing pfp...

> 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 at level of FDR=0.05, indicating a relative good consistency of the experiments conducted at two labs. Further study the top 10 genes in the lists indicates that those genes are indeed 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  75.5385             0.8302 0.9250  0.0018
245181_at          281  80.3077             0.8253 0.6500  0.0026
245305_at          405  83.8462             0.8603 0.5450  0.0033
245233_at          333  89.5385             0.8589 0.5625  0.0045
245254_at          354  89.7692             0.8246 0.4520  0.0045
244951_s_at         51  95.0000             0.8757 0.4925  0.0059
245269_at          369 104.6923             0.8857 0.7229  0.0101
245082_at          182 106.5385             0.8583 0.6862  0.0110
245234_at          334 107.2308             0.8627 0.6300  0.0113
245380_at          480 109.4615             0.8924 0.6270  0.0125

$Table2
          gene.index  RP/Rsum FC:(class1/class2)    pfp P.value
245259_at        359  98.5385             1.1298 3.8900  0.0078
245343_at        443  98.9231             1.1236 1.9850  0.0079
245252_at        352 100.5385             1.1839 1.4367  0.0086
245338_at        438 105.4615             1.1656 1.3800  0.0110
245080_at        180 121.6154             1.1203 2.3060  0.0231
245188_at        288 126.3846             1.1088 2.3017  0.0276
245098_at        198 127.2308             1.0786 2.0564  0.0288
244986_at         86 128.1538             1.1526 1.8781  0.0300
245277_at        377 131.3846             1.0567 1.8767  0.0338
245021_at        121 133.3077             1.2048 1.8210  0.0364

> 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 plot of the estimated pfp *vs.* number of identified genes is shown in figure 2. The abnormal patterns in the plot compared with that in figure 1 also indicate a meaningless identification. However, due to the stability of the rank product statistics against outliers, genes would be identified using rank product method

```
> RP.adv.out <- RPadvance(arab,arab.cl2,arab.origin,num.perm=100,
+ logged=TRUE,gene.names=arab.gnames,rand=123)

 The data is from  2 different origins

Rank Product analysis for two-class case

Starting  100 permutations...
Computing pfp...
```

```
> 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  8.8253             0.8710 0.0000   0e+00
245336_at        436 12.4416             0.9822 0.0050   0e+00
245119_at        219 13.3021             0.7238 0.0033   0e+00
245304_at        404 15.5140             0.7973 0.0025   0e+00
245176_at        276 15.5174             0.8950 0.0020   0e+00
245254_at        354 23.9023             0.8246 0.0200   2e-04
245196_at        296 26.8246             0.8908 0.0314   4e-04


$Table2
          gene.index RP/Rsum FC:(class1/class2)    pfp P.value
245362_at        462  6.7647             1.0344 0.0000   0e+00
245136_at        236 13.4622             1.0749 0.0050   0e+00
245277_at        377 19.1517             1.0567 0.0133   1e-04
245296_at        396 22.0083             0.9371 0.0250   2e-04
```

However, the log fold-change indicating non-significant finding, which is also confirmed by the further study of the the ranks under 13 comparisons for one gene(first 9 in lab 1, next 4 in from lab 2).

```
> RP.adv.out$Orirank[[1]][344,]
[1]   3   4   3   1   1   1   1   1   1 495 496 453 492
```

The approach discussed in this section is still in the early stages of development, is not published yet. Hence, it should only be used with caution and with further detailed examination.

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