# Outline of SPRINT Functions

Parallel Computing with R using SPRINT
on post-genomic data

# apply() ➔ papply()

This function allows you to apply any function (existing or defined by you) to each of the rows or each of the columns of a data matrix.

Example use:

- Compute an error estimate for each gene's fold change on a microarray
- Compute a count statistic for each sequence in a RNA-seq run

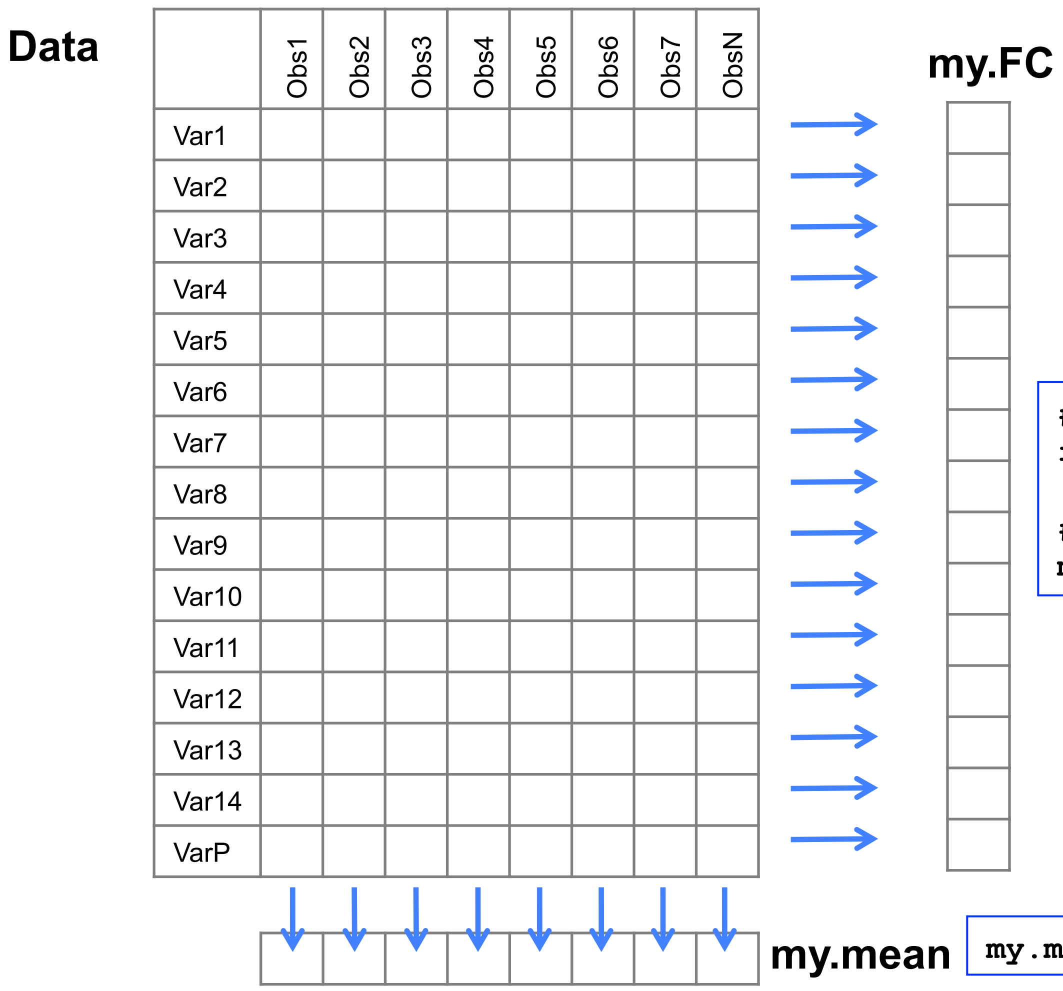
Notes:

- You will not need the parallelised option if you are applying a 'simple' function (t test etc.) to all the genes on a microarray…unless you are trying to do this repeatedly for research purposes
- However, apply() is highly generic so uses (including computationally challenging ones) are plentiful

# {base} apply() papply()

**For each row (or each column) in a data matrix, apply a function of your choice or making**

Input: Data matrix or array
Output: A calculated result for every row or column
Use: Quicker replacement for “for-loops”



```
#make a function to calculate fold-change
f.FC  <- function(x) log2(x[1:4] / [5:8])

#apply function to each row
my.FC <- apply(Data, 1, f.FC)
```

```
my.mean <- apply(Data, 1, mean)
```

## boot() ➔ pboot()

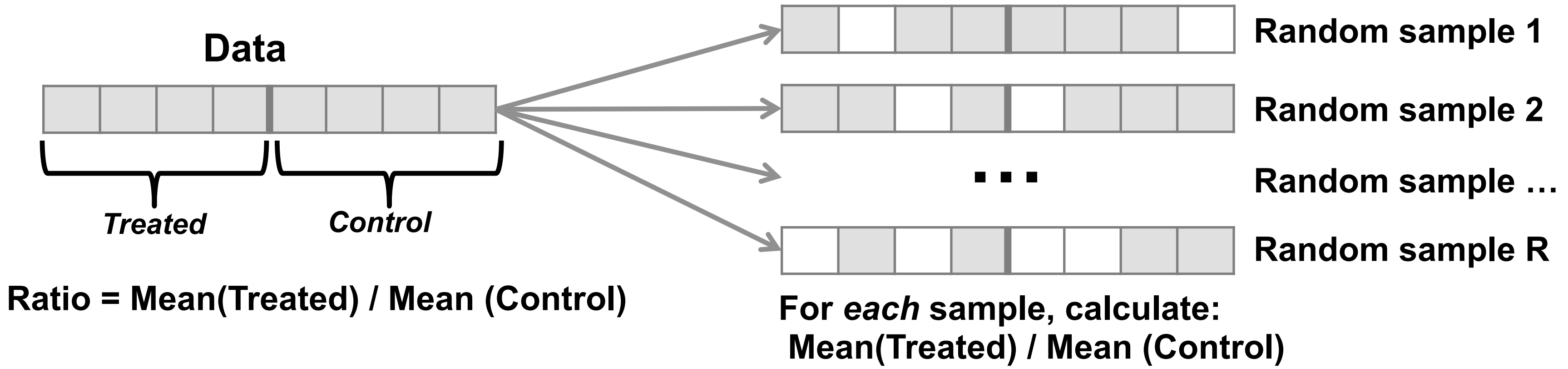
This function allows you to 'bootstrap' any function, i.e. to repeatedly apply the same function to resampled sets of your input data with the intention of estimating properties of your function.

Example use:

- Compute the standard error of a ratio (e.g. gene fold change) for a gene
- Compute a Pearson correlation matrix on repeatedly resampled microarray data to obtain a Null distribution for Pearson's correlation coefficient

# {boot} boot() pboot()

Input: Data vector or matrix
Output: Bootstrapped metric of your choice
Use: Obtaining a robust statistic with fewer assumptions


Data
Treated
Control
Ratio = Mean(Treated) / Mean (Control)
Random sample 1
Random sample 2
. . .
Random sample …
Random sample R
For *each* sample, calculate:
Mean(Treated) / Mean (Control)

```
#make a function to calculate ratio of two means
f.ratio  <- function(x,w) sum(x[1:4]*w) / sum(x[5:8]*w)

#bootstrap this expression ratio function to obtain standard error of ratio
#this outputs the ratio of means and its standard error
boot.ratio <- boot(Data, f.ratio, R=100, stype="w")
```

## cor() ➔ pcor()

This function computes Pearson correlation coefficients between any two sets of numbers, or between all rows (or columns) in a data matrix

Example use:

- Compute an adjacency or "guilt-by-association" matrix, pairwise for all genes in a microarray data set

**{stats}**
**cor()**
**pcor()**

## Correlation between all possible pairs of rows (or columns) in a data matrix.

Input: Data matrix
Output: Matrix of all possible correlation coefficients
Use: Input for network graphs, clustering,…

| | Obs1 | Obs2 | Obs3 | Obs4 | Obs5 | Obs6 | Obs7 | ObsN |
|---|---|---|---|---|---|---|---|---|
| Var1 | | | | | | | | |
| Var2 | | | | | | | | |
| Var3 | | | | | | | | |
| Var4 | | | | | | | | |
| Var5 | | | | | | | | |
| Var6 | | | | | | | | |
| Var7 | | | | | | | | |
| Var8 | | | | | | | | |
| Var9 | | | | | | | | |
| Var10 | | | | | | | | |
| Var11 | | | | | | | | |
| Var12 | | | | | | | | |
| Var13 | | | | | | | | |
| Var14 | | | | | | | | |
| VarP | | | | | | | | |

**Data**
**(*N* rows)**

**Perform correlation on all possible pairs of variables.**

| | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Var1 | **1** | | | | | | | | | | | | | | |
| Var2 | | **1** | | | | | | | | | | | | | |
| Var3 | | | **1** | | | | | | | | | | | | |
| Var4 | | | | **1** | | | | | | | | | | | |
| Var5 | | | | | **1** | | | | | | | | | | |
| Var6 | | | | | | **1** | | | | | | | | | |
| Var7 | | | | | | | **1** | | | | | | | | |
| Var8 | | | | | | | | **1** | | | | | | | |
| Var9 | | | | | | | | | **1** | | | | | | |
| Var10 | | | | | | | | | | **1** | | | | | |
| Var11 | | | | | | | | | | | **1** | | | | |
| Var12 | | | | | | | | | | | | **1** | | | |
| Var13 | | | | | | | | | | | | | **1** | | |
| Var14 | | | | | | | | | | | | | | **1** | |
| VarP | | | | | | | | | | | | | | | **1** |

**Correlation/similarity/association matrix**
**(1-correlation matrix = distance matrix)**
**(*N*$^2$ correlation coefficients)**

```
my.cor <- cor(t(Data))
```

# stringdistmatrix() ➔ pstringdistmatrix()

This function computes the distance between all pairs of string vectors in a data set (for now, Hamming distance only)

Example use:

- Compute similarity matrix of nucleotide sequences in an RNA-seq run

## {strindistmatrix} stringdistmatrix() pstringdistmatrix()

**Similar to cor(), but for nominal or categorical data**

Input: Vector of strings
Output: Matrix of alignment/overlap coefficients
Use: Input for network graphs, clustering,…

**Data= string vector**

***(N strings)***

**Calculate string alignment on all possible string pairs.**

| | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 | Var1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Var1 | **0** | | | | | | | | | | | | | | |
| Var2 | | **0** | | | | | | | | | | | | | |
| Var3 | | | **0** | | | | | | | | | | | | |
| Var4 | | | | **0** | | | | | | | | | | | |
| Var5 | | | | | **0** | | | | | | | | | | |
| Var6 | | | | | | **0** | | | | | | | | | |
| Var7 | | | | | | | **0** | | | | | | | | |
| Var8 | | | | | | | | **0** | | | | | | | |
| Var9 | | | | | | | | | **0** | | | | | | |
| Var10 | | | | | | | | | | **0** | | | | | |
| Var11 | | | | | | | | | | | **0** | | | | |
| Var12 | | | | | | | | | | | | **0** | | | |
| Var13 | | | | | | | | | | | | | **0** | | |
| Var14 | | | | | | | | | | | | | | **0** | |
| VarP | | | | | | | | | | | | | | | **0** |

**Distance matrix**

**($N^2$ alignment scores)**

```
my.sdist <- stringdistmatrix(Data)
```

## pam() ➔ ppam()

The clustering function Partitioning-Around-Medoids groups numerical vectors by their similarity (this is the non-parametric equivalent to K-means clustering).

Example use:

- Grouping of genes by similarity of their expression vectors across samples

**{cluster}**
**pam()**
**ppam()**

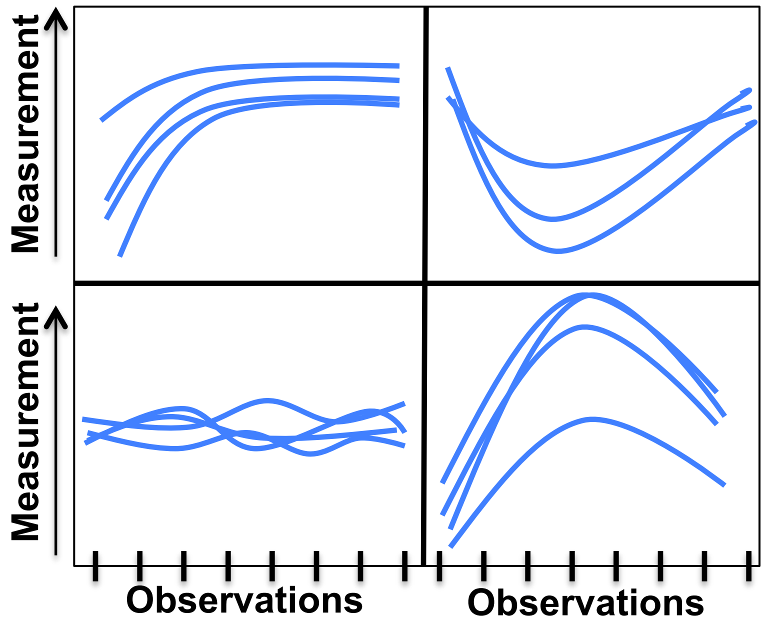
## Without prior information, identify similar sets (clusters) of variables or observations in data

Input: Data matrix
Output: Rows or column sorted into clusters of similarity
Use: Finding patterns in time series or multi-condition data

**Data**

| | Obs1 | Obs2 | Obs3 | Obs4 | Obs5 | Obs6 | Obs7 | ObsN |
|---|---|---|---|---|---|---|---|---|
| Var1 | | | | | | | | |
| Var2 | | | | | | | | |
| Var3 | | | | | | | | |
| Var4 | | | | | | | | |
| Var5 | | | | | | | | |
| Var6 | | | | | | | | |
| Var7 | | | | | | | | |
| Var8 | | | | | | | | |
| Var9 | | | | | | | | |
| Var10 | | | | | | | | |
| Var11 | | | | | | | | |
| Var12 | | | | | | | | |
| Var13 | | | | | | | | |
| Var14 | | | | | | | | |
| VarP | | | | | | | | |

**Compute distance between all possible pairs of variables, then partition into sets of maximal distinction**



**4 clusters**

```
my.pam <- pam(t(Data))
```

# randomForest() ➔ prandomForest()

This classification function allows the identification of new/unknown observations on the basis of a training on a set of known observations. It also identifies which variables (e.g. genes) are most useful in classification.

Example use:

- Train classifier on a set of known cancer status samples and classify new biological microarray samples as cancerous or healthy
- Each 'tree' is a different bootstrap sample of the data, each uses different gene sets to decide if a given sample is 'cancer' or 'healthy'. Aggregate data from thousands of such trees make up the "random forest" classifier.

Notes: for an introduction of random forests on microarray data, see:
"Gene selection and classification of microarray data using random forest", Diaz-Uriarte et al 2006, Bioinformatics.

## {randomForest}
## randomForest()
## prandomForest()

**Based on known observations, predict class membership of a new observation**

Input: Data matrix
Output: Predictions of which class a new sample belongs to
Use: Classify new data based on known data

**Data**

| | Obs1 | Obs2 | Obs3 | Obs4 | Obs5 | Obs6 | Obs7 | ObsN |
|---|---|---|---|---|---|---|---|---|
| Var1 | | | | | | | | |
| Var2 | | | | | | | | |
| Var3 | | | | | | | | |
| Var4 | | | | | | | | |
| Var5 | | | | | | | | |
| Var6 | | | | | | | | |
| Var7 | | | | | | | | |
| Var8 | | | | | | | | |
| Var9 | | | | | | | | |
| Var10 | | | | | | | | |
| Var11 | | | | | | | | |
| Var12 | | | | | | | | |
| Var13 | | | | | | | | |
| Var14 | | | | | | | | |
| VarP | | | | | | | | |

**Class:** *Treated* *Control*

**Construct 'forest' of decision trees, where each decision tree is based on a bootstrap sample of the input data set. Splitting variables (genes) within each decision tree are chosen randomly. Trees are aggregated by majority vot.**

**Predicted class for new observations**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Treated | Treated | Control | Control | Treated | Control | Treated | Treated |

**Which variables are best in predicting class**

| |
|---|
| Var5 |
| Var7 |
| Var34 |
| Var100 |
| Var29 |
| Var655 |

```
my.rf <- randomForest(x=Data, y=rep(c(1,0),each=4))
```

# mt.maxT() $\rightarrow$ pmaxT()

Compute permutation multiple-testing-adjusted p-values (Westfall & Young 1993)

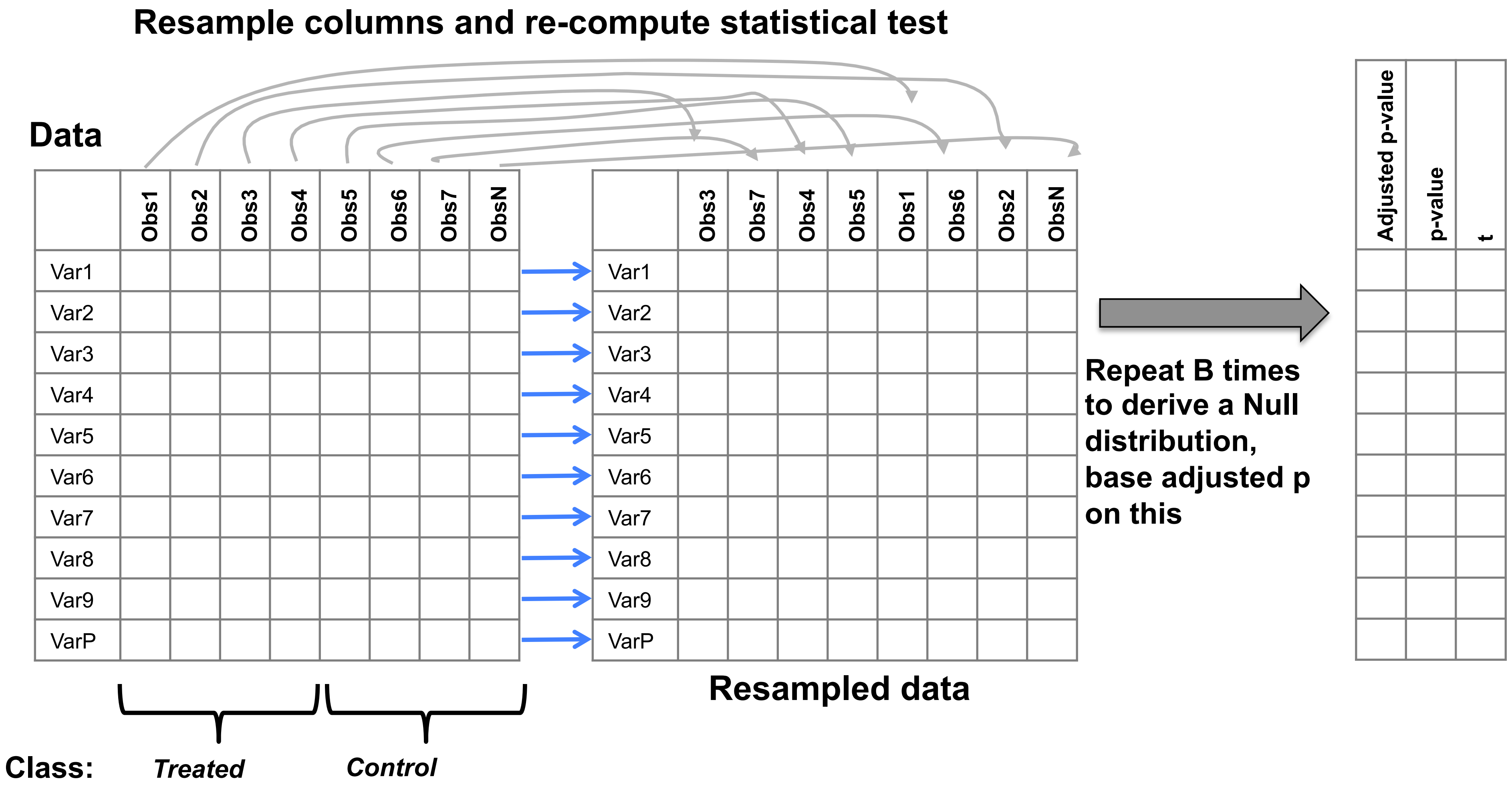
Example use:

- Compute t-tests and adjust p-values for each probe in an Exon microarray study

**{multtest}**
**mt.maxT()**
**pmaxT()**

## Compute permutation-based statistical hypothesis tests for each row (or column) in a data set, also adjusting for multiple-testing issue.

Input: Data matrix
Output: Statistical permutation test result for each row, adjusted for multiple testing
Use: Comparisons of two or more groups where many hypotheses are tested simultaneously



# RP() ➔ pRP()

Computes non-parametric rank-product inference tests for genes

Example use:

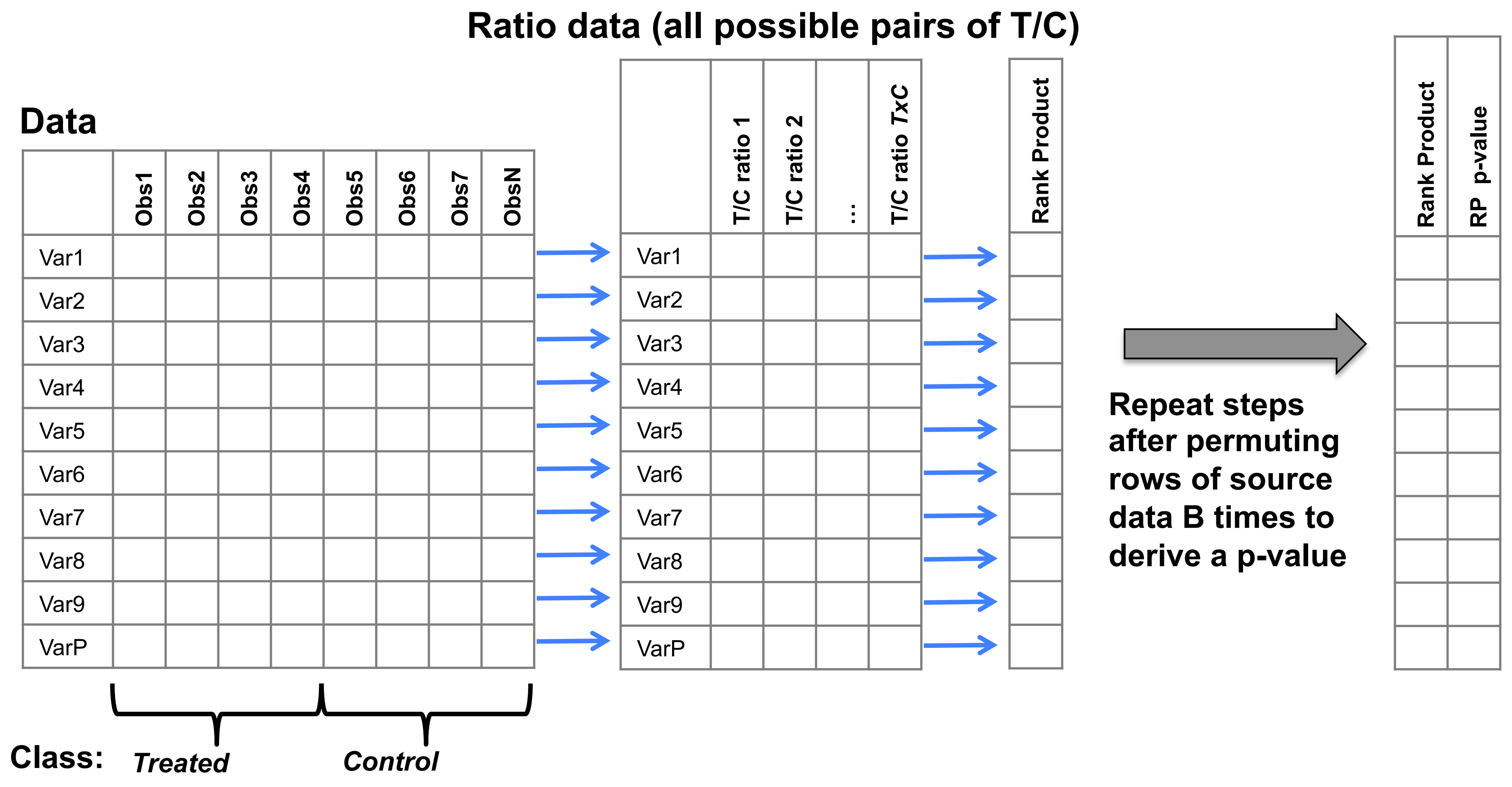
- Compute statistical significance of gene expression fold-changes (rather than mean expression difference)

Notes: the RP() function implementation has been revised recently and is now less computationally challenging in its serial form.

# {RankProd} RP() pRP()

## Statistical test to identify consistently top-ranked variables

Input: Data matrix
Output: Statistical rank product test result for each row
Use: Top-ranking consistency for one group, or differential between two groups, or meta-analysis



Ratio data (all possible pairs of T/C)
Data
Obs1
Obs2
Obs3
Obs4
Obs5
Obs6
Obs7
ObsN
Var1
Var2
Var3
Var4
Var5
Var6
Var7
Var8
Var9
VarP
Class:
Treated
Control
T/C ratio 1
T/C ratio 2
...
T/C ratio TxC
Rank Product
Repeat steps after permuting rows of source data B times to derive a p-value
Rank Product
RP p-value