

# Package ‘WilcoxCV’

February 14, 2012

**Version** 1.0-2

**Date** 2009-05-29

**Title** Wilcoxon-based variable selection in cross-validation

**Author** Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>.

**Maintainer** Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

**Depends** R (>= 2.0.0)

**Suggests**

**Description** This package provides functions to perform fast variable selection based on the Wilcoxon rank sum test in the cross-validation or Monte-Carlo cross-validation settings, for use in microarray-based binary classification.

**License** GPL (>= 2)

**URL** <http://cran.r-project.org/web/packages/WilcoxCV/index.html>

**Repository** CRAN

**Date/Publication** 2009-05-29 11:43:26

## R topics documented:

generate.cv . . . . .	2
generate.split . . . . .	3
wilcox.selection.split . . . . .	4
wilcox.split . . . . .	5
<b>Index</b>	<b>8</b>

---

`generate.cv`*Generating groups for cross-validation*

---

**Description**

The function `generate.cv` generates randomly  $m$  groups for  $m$ -fold cross-validation.

**Usage**

```
generate.cv(n,m)
```

**Arguments**

<code>n</code>	The total number of observations in the data set.
<code>m</code>	The desired number of groups.

**Details**

Leave-one-out cross-validation is a special case of cross-validation, with  $m=n$ .

**Value**

A  $m \times \text{ceiling}(n/m)$  matrix giving the indices of the observations included in each group. The  $i$ -th row gives the indices of observations included in the  $i$ -th group. If the  $m$  groups are not perfectly equally sized, the last column includes one or several zero(s).

**Author(s)**

Anne-Laure Boulesteix ([http://www.ibe.med.uni-muenchen.de/organisation/mitarbeiter/020\\_professuren/boulesteix/index.html](http://www.ibe.med.uni-muenchen.de/organisation/mitarbeiter/020_professuren/boulesteix/index.html))

**References**

A. L. Boulesteix (2007). WilcoxCV: an R package for fast variable selection in cross-validation. *Bioinformatics* 23:1702-1704.

**See Also**

[generate.split](#), [wilcox.split](#), [wilcox.selection.split](#)

**Examples**

```
# load WilcoxCV library
library(WilcoxCV)

# Generate 10 groups for a data set of size 95.
my.cv<-generate.cv(n=95,m=10)
```

---

generate.split      *Generating random splittings into learning and test data sets*

---

### Description

The function `generate.split` generates `niter` random splittings into learning and test data sets for use in Monte-Carlo cross-validation (MCCV).

### Usage

```
generate.split(niter,n,ntest)
```

### Arguments

<code>niter</code>	The number of iterations (number of splits into learning and split sets).
<code>n</code>	The total number of observations in the data set.
<code>ntest</code>	The number of observations in the test sets.

### Details

This function is meant for use in Monte-Carlo cross-validation (MCCV).

### Value

A `niter` x `ntest` matrix giving the indices of the observations included in the test sets. The *i*-th row gives the indices of the `ntest` observations included in the test set for the *i*-th MCCV iteration.

### Author(s)

Anne-Laure Boulesteix ([http://www.ibe.med.uni-muenchen.de/organisation/mitarbeiter/020\\_professuren/boulesteix/index.html](http://www.ibe.med.uni-muenchen.de/organisation/mitarbeiter/020_professuren/boulesteix/index.html))

### References

A. L. Boulesteix (2007). *WilcoxCV: an R package for fast variable selection in cross-validation*. *Bioinformatics* 23:1702-1704.

### See Also

[generate.cv](#), [wilcox.split](#), [wilcox.selection.split](#)

### Examples

```
# load WilcoxCV library
library(WilcoxCV)

# Generate 50 splits with ratio 2:1 for a data set including 90 observations
my.split<-generate.split(niter=50,n=90,ntest=30)
```

wilcox.selection.split

*Wilcoxon-based variable selection in cross-validation (CV) and Monte-Carlo cross-validation (MCCV)*

---

### Description

The function `wilcox.selection.split` performs variable ordering based on the Wilcoxon rank sum test for all `niter` CV or MCCV iterations.

### Usage

```
wilcox.selection.split(x,y,split,algo="new",pvalue=FALSE)
```

### Arguments

<code>x</code>	a matrix or a data frame of size $n \times p$ giving the expression levels of the $p$ variables (genes) for the $n$ observations (arrays). Variables correspond to columns, observations to rows.
<code>y</code>	a vector of length $n$ giving the class membership for the $n$ observations (arrays). <code>y</code> can be either a factor or a numeric and must be coded as 0,1.
<code>split</code>	A $niter \times ntest$ matrix giving the indices of the $ntest$ observations included in each of the $niter$ test sets, as generated by the functions <a href="#">generate.split</a> or <a href="#">generate.cv</a> . The $i$ -th row of <code>split</code> gives the indices of the observations included in the test data set for the $i$ -th random splitting iteration.
<code>algo</code>	either "new" or "naive". If <code>type="new"</code> , the new fast method described in Boulesteix (2007) is used. If <code>type="naive"</code> , results are obtained by running the function <code>wilcox.test</code> <code>niter</code> times.
<code>pvalue</code>	Logical. Should p-values be returned?

### Details

The Wilcoxon rank sum statistic is defined as the sum of the  $X$ -ranks of the observations with  $y=0$ . The Wilcoxon rank sum test is equivalent to the Mann-Whitney test. It is implemented in the function `wilcox.test`.

In the context of cross-validation (CV) or Monte-Carlo cross-validation (MCCV), `wilcox.selection.split` computes the Wilcoxon rank sum statistic for each iteration, for each variable. At each iteration, a subset of the  $n$  observations is excluded from the data set and considered as test data set. The indices of the observations considered as test set for each of the `niter` iterations are given in the  $niter \times ntest$  matrix `split`.

### Value

A list with the following components:

- `ordering.split` A `niter x p` matrix giving the indices of the genes ordered by `pvalue`. For example, the first column of `ordering.split` gives the index of the variable with lowest `pvalue` in each of the `niter` random splitting iterations, the second column of `ordering.split` gives the index of the variable with the second lowest `pvalue` in each of the `niter` random splitting iterations. For the *i*-th iteration, the indices of the 50 best variables are given in the 50 first columns of row *i*.
- `pvalue.split` Returned only if `pvalue=TRUE`. A `niter x p` matrix of `pvalues`. The element in the *i*-th row and *j*-th column is the `pvalue` of variable *j* in the *i*-th iteration.

**Author(s)**

Anne-Laure Boulesteix ([http://www.ibe.med.uni-muenchen.de/organisation/mitarbeiter/020\\_professuren/boulesteix/index.html](http://www.ibe.med.uni-muenchen.de/organisation/mitarbeiter/020_professuren/boulesteix/index.html))

**References**

A. L. Boulesteix (2007). WilcoxCV: an R package for fast variable selection in cross-validation. *Bioinformatics* 23:1702-1704.

**See Also**

[wilcox.test](#), [generate.split](#), [generate.cv](#), [wilcox.split](#)

**Examples**

```
# load WilcoxCV library
library(WilcoxCV)

# Generate data
x<-matrix(rnorm(1000),100,10)
y<-sample(c(0,1),100,replace=TRUE)

# Generate 50 MCCV splits with ratio 2:1 for a data set including 90 observations
my.split<-generate.split(niter=50,n=90,ntest=30)

# Compute the Wilcoxon rank sum statistic for the 50 iterations.
wilcox.selection.split(x=x,y=y,split=my.split,algo="new",pvalue=TRUE)
```

---

<code>wilcox.split</code>	<i>Wilcoxon rank sum statistic in cross-validation (CV) and Monte-Carlo cross-validation (MCCV)</i>
---------------------------	---

---

**Description**

The function `wilcox.split` computes the Wilcoxon rank sum statistic for all `niter` CV or MCCV iterations defined by the matrix `split`.

**Usage**

```
wilcox.split(x,y,split,algo="new")
```

**Arguments**

x	a numeric vector of length n giving the expression levels of a gene for the n arrays.
y	a vector of length n giving the class membership for the n arrays. y can be either a factor or a numeric and must be coded as 0,1.
split	A niter x ntest matrix giving the indices of the ntest observations included in each of the niter test sets, as generated by the functions <a href="#">generate.split</a> or <a href="#">generate.cv</a> . The i-th row of split gives the indices of the observations included in the test data set for the i-th iteration.
algo	either "new" or "naive". If algo="new", the new fast method described in Boulesteix (2007) is used to compute the Wilcoxon rank statistic. If algo="naive", the Wilcoxon rank sum statistics are obtained by running the function <code>wilcox.test</code> niter times.

**Details**

The Wilcoxon rank sum statistic is defined as the sum of the X-ranks of the observations with  $y=0$ . The Wilcoxon rank sum test is equivalent to the Mann-Whitney test. It is implemented in the function `wilcox.test`.

In the context of cross-validation (CV) or Monte-Carlo cross-validation (MCCV), `wilcox.selection.split` computes the Wilcoxon rank sum statistic for each iteration. At each iteration, a subset of the n observations is excluded from the data set and considered as test data set. The indices of the observations considered as test set for each of the niter iterations are given in the niter x ntest matrix `split`.

**Value**

A list with the following components:

wilcox.split	a numeric vector of length niter whose i-th component gives the Wilcoxon rank sum statistic obtained in the i-th iteration.
--------------	---

**Author(s)**

Anne-Laure Boulesteix ([http://www.ibe.med.uni-muenchen.de/organisation/mitarbeiter/020\\_professuren/boulesteix/index.html](http://www.ibe.med.uni-muenchen.de/organisation/mitarbeiter/020_professuren/boulesteix/index.html))

**References**

A. L. Boulesteix (2007). WilcoxCV: an R package for fast variable selection in cross-validation. *Bioinformatics* 23:1702-1704.

**See Also**

[wilcox.test](#), [generate.split](#), [generate.cv](#), [wilcox.selection.split](#)

**Examples**

```
# load WilcoxCV library
library(WilcoxCV)

# Generate data
x<-rnorm(100)
y<-sample(c(0,1),100,replace=TRUE)

# Generate 50 MCCV splits with ratio 2:1 for a data set including 90 observations
my.split<-generate.split(niter=50,n=90,ntest=30)

# Compute the Wilcoxon rank sum statistic for the 50 iterations.
wilcox.split(x=x,y=y,split=my.split,algo="new")
```

# Index

## \*Topic **hstest**

- generate.cv, 2
- generate.split, 3
- wilcox.selection.split, 4
- wilcox.split, 5

generate.cv, 2, 3–6  
generate.split, 2, 3, 4–6

wilcox.selection.split, 2, 3, 4, 6  
wilcox.split, 2, 3, 5, 5  
wilcox.test, 5, 6