

# Package ‘crossval’

May 8, 2026

**Version** 1.0.5

**Date** 2023-06-08

**Title** Generic Functions for Cross Validation

**Author** Korbinian Strimmer.

**Maintainer** Korbinian Strimmer <strimmerlab@gmail.com>

**Depends** R (>= 3.0.2)

**Imports** stats

**Suggests** MASS, sda, care, binda

**Description** Contains generic functions for performing  
cross validation and for computing diagnostic errors.

**License** GPL (>= 3)

**URL** <https://cran.r-project.org/package=crossval>

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2023-06-08 22:40:02 UTC

## Contents

crossval-package . . . . .	2
confusionMatrix . . . . .	2
crossval . . . . .	3
diagnosticErrors . . . . .	6

<b>Index</b>	<b>8</b>
--------------	----------

---

crossval-package      *The crossval Package*

---

### Description

The "crossval" package implements generic functions for performing cross validation and for computing diagnostic errors.

### Author(s)

Korbinian Strimmer (<https://strimmerlab.github.io/>)

### References

Website: <https://cran.r-project.org/package=crossval>

### See Also

[crossval](#), [confusionMatrix](#), [diagnosticErrors](#).

---

confusionMatrix      *Compute Confusion Matrix*

---

### Description

`confusionMatrix` computes the confusion matrix, i.e. it counts the number of false positives (FP), true positives (TP), true negatives (TN), and false negatives (FN).

Despite its name the functions returns a vector rather than an actual matrix for easier use with the [crossval](#) function.

### Usage

```
confusionMatrix(actual, predicted, negative="control")
```

### Arguments

actual	a vector containing the actual correct labels for each sample (e.g. "cancer" or "control").
predicted	a vector containing the predicted labels.
negative	the label of a negative "null" sample (default: "control").

### Value

`confusionMatrix` returns a vector of length 4 containing the counts for FP, TP, TN, and FN.

**Author(s)**

Korbinian Strimmer (<https://strimmerlab.github.io>).

**See Also**

[diagnosticErrors](#).

**Examples**

```
# load crossval library
library("crossval")

# true labels
a = c("cancer", "cancer", "control", "control", "cancer", "control", "control")

# predicted labels
p = c("cancer", "control", "control", "control", "cancer", "control", "cancer")

# confusion matrix (a vector)
cm = confusionMatrix(a, p, negative="control")
cm
# FP TP TN FN
# 1 2 3 1
# attr(,"negative")
# [1] "control"

# corresponding accuracy, sensitivity etc.
diagnosticErrors(cm)
#      acc      sens      spec      ppv      npv      lor
# 0.7142857 0.6666667 0.7500000 0.6666667 0.7500000 1.7917595
# attr(,"negative")
# [1] "control"
```

---

crossval

*Generic Function for Cross Validation*

---

**Description**

crossval performs K-fold cross validation with B repetitions. If Y is a factor then balanced sampling is used (i.e. in each fold each category is represented in appropriate proportions).

**Usage**

```
crossval(predfun, X, Y, K=10, B=20, verbose=TRUE, ...)
```

**Arguments**

predfun	Prediction function (see details).
X	Matrix of predictors (columns correspond to variables).
Y	Univariate response variable.
K	Number of folds.
B	Number of repetitions.
verbose	If verbose=TRUE then status messages appear during cross validation.
...	optional arguments for predfun

**Details**

The argument predfun must be a function of the form predfun(Xtrain, Ytrain, Xtest, Ytest, ...).

**Value**

crossval returns a list with three entries:

stat.cv: the statistic returned by predfun for each cross validation run.

stat: the statistic returned by predfun averaged over all cross validation runs.

stat.se: the corresponding standard error.

**Author(s)**

Korbinian Strimmer (<https://strimmerlab.github.io>).

**See Also**

[confusionMatrix](#).

**Examples**

```
# load "crossval" package
library("crossval")

# classification examples

# set up lda prediction function
predfun.lda = function(train.x, train.y, test.x, test.y, negative)
{
  require("MASS") # for lda function

  lda.fit = lda(train.x, grouping=train.y)
  ynew = predict(lda.fit, test.x)$class

  # count TP, FP etc.
  out = confusionMatrix(test.y, ynew, negative=negative)

  return( out )
}
```

```
}

# Student's Sleep Data
data(sleep)
X = as.matrix(sleep[,1, drop=FALSE]) # increase in hours of sleep
Y = sleep[,2] # drug given
plot(X ~ Y)
levels(Y) # "1" "2"
dim(X) # 20 1

set.seed(12345)
cv.out = crossval(predfun.lda, X, Y, K=5, B=20, negative="1")

cv.out$stat
diagnosticErrors(cv.out$stat)

# linear regression example

data("attitude")
y = attitude[,1] # rating variable
x = attitude[,-1] # data frame with the remaining variables
is.factor(y) # FALSE

summary( lm(y ~ . , data=x) )

# set up lm prediction function
predfun.lm = function(train.x, train.y, test.x, test.y)
{
  lm.fit = lm(train.y ~ . , data=train.x)
  ynew = predict(lm.fit, test.x )

  # compute squared error risk (MSE)
  out = mean( (ynew - test.y)^2 )

  return( out )
}

# prediction MSE using all variables
set.seed(12345)
cv.out = crossval(predfun.lm, x, y, K=5, B=20)
c(cv.out$stat, cv.out$stat.se)

# and only two variables
cv.out = crossval(predfun.lm, x[,c(1,3)], y, K=5, B=20)
c(cv.out$stat, cv.out$stat.se)

# for more examples (e.g. using cross validation in a regression or classification context)
# see the R packages "sda", "care", or "binda".
```

---

diagnosticErrors	<i>Compute Diagnostic Errors: Accuracy, Sensitivity, Specificity, Positive Predictive Value, Negative Predictive Value, Log Odds Ratio</i>
------------------	--

---

### Description

diagnosticErrors computes various diagnostic errors useful for evaluating the performance of a diagnostic test or a classifier: accuracy (acc), sensitivity (sens), specificity (spec), positive predictive value (ppv), negative predictive value (npv), and log-odds ratio (lor).

### Usage

```
diagnosticErrors(cm)
```

### Arguments

cm                    a vector containing the true positives, false positives etc, as computed by [confusionMatrix](#).

### Details

The diagnostic errors are computed as follows:

$$\text{acc} = (\text{TP} + \text{TN}) / (\text{FP} + \text{TN} + \text{TP} + \text{FN})$$

$$\text{sens} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{spec} = \text{TN} / (\text{FP} + \text{TN})$$

$$\text{ppv} = \text{TP} / (\text{FP} + \text{TP})$$

$$\text{npv} = \text{TN} / (\text{TN} + \text{FN})$$

$$\text{lor} = \log(\text{TP} * \text{TN} / (\text{FN} * \text{FP}))$$

### Value

diagnostic errors returns a vector containing various diagnostic errors.

### Author(s)

Korbinian Strimmer (<https://strimmerlab.github.io>).

### See Also

[confusionMatrix](#).

**Examples**

```
# load crossval library
library("crossval")

# true labels
a = c("cancer", "cancer", "control", "control", "cancer", "control", "control")

# predicted labels
p = c("cancer", "control", "control", "control", "cancer", "control", "cancer")

# confusion matrix (a vector)
cm = confusionMatrix(a, p, negative="control")
cm
# FP TP TN FN
# 1 2 3 1
# attr(,"negative")
# [1] "control"

# corresponding accuracy, sensitivity etc.
diagnosticErrors(cm)
#      acc      sens      spec      ppv      npv      lor
# 0.7142857 0.6666667 0.7500000 0.6666667 0.7500000 1.7917595
# attr(,"negative")
# [1] "control"
```

# Index

\* **multivariate**

crossval, [3](#)

crossval-package, [2](#)

\* **univar**

confusionMatrix, [2](#)

diagnosticErrors, [6](#)

confusionMatrix, [2](#), [2](#), [4](#), [6](#)

crossval, [2](#), [3](#)

crossval-package, [2](#)

diagnosticErrors, [2](#), [3](#), [6](#)