Package ‘precrec’

August 23, 2017

Type Package

Title Calculate Accurate Precision-Recall and ROC (Receiver Operator Characteristics) Curves

Version 0.9.1

Date 2017-08-23

Description Accurate calculations and visualization of precision-recall and ROC (Receiver Operator Characteristics) curves.

URL http://takayasaito.github.io/precrec,
    https://github.com/takayasaito/precrec

BugReports https://github.com/takayasaito/precrec/issues

Depends R (>= 3.2.1)

License GPL-3

LazyData TRUE

Suggests testthat (>= 0.11.0), knitr (>= 1.11), rmarkdown (>= 0.8.1)

LinkingTo Rcpp

Imports Rcpp (>= 0.12.2), ggplot2 (>= 2.1.0), assertthat (>= 0.1),
    grid, gridExtra (>= 2.0.0), methods, data.table (>= 1.10.4)

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation yes

Author Takaya Saito [aut, cre],
    Marc Rehmsmeier [aut]

Maintainer Takaya Saito <takaya.saito@outlook.com>

Repository CRAN

Date/Publication 2017-08-23 07:21:31 UTC
R topics documented:

- as.data.frame .................................................. 2
- auc ...................................................................... 7
- autoplot .................................................................. 9
- B1000 ................................................................. 16
- B500 ................................................................. 16
- create_sim_samples ................................................. 17
- evalmod .............................................................. 18
- format_nfold ...................................................... 23
- fortify ............................................................... 25
- IB1000 ............................................................... 30
- IB500 ............................................................... 31
- join_labels .......................................................... 32
- join_scores ........................................................ 33
- M2N50F5 ............................................................ 34
- mmdata .............................................................. 35
- P10N10 .............................................................. 38
- part ................................................................. 39
- pauc ............................................................... 42
- plot ................................................................. 44
- precocr ............................................................ 49

Index .............................................................. 51

as.data.frame .............................. Convert a curves and points object to a data frame

Description

The as.data.frame function converts an S3 object generated by evalmod to a data frame.

Usage

```r
## S3 method for class 'sscurves'
as.data.frame(x, row.names = NULL, optional = FALSE,
              raw_curves = NULL, ...)

## S3 method for class 'mscurves'
as.data.frame(x, row.names = NULL, optional = FALSE,
              raw_curves = NULL, ...)

## S3 method for class 'smcurves'
as.data.frame(x, row.names = NULL, optional = FALSE,
              raw_curves = NULL, ...)

## S3 method for class 'mmcurves'
as.data.frame(x, row.names = NULL, optional = FALSE,
```
raw_curves = NULL, ...)

## S3 method for class 'sspoints'
as.data.frame(x, row.names = NULL, optional = FALSE,
               raw_curves = NULL, ...)

## S3 method for class 'mspoints'
as.data.frame(x, row.names = NULL, optional = FALSE,
               raw_curves = NULL, ...)

## S3 method for class 'smpoints'
as.data.frame(x, row.names = NULL, optional = FALSE,
               raw_curves = NULL, ...)

## S3 method for class 'mmpoints'
as.data.frame(x, row.names = NULL, optional = FALSE,
               raw_curves = NULL, ...)

## S3 method for class 'aucroc'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

Arguments

x

An S3 object generated by evalmod. The as.data.frame function takes one of the following S3 objects.

1. ROC and Precision-Recall curves (mode = "rocprc")

   **S3 object**  **# of models**  **# of test datasets**
   sscurves   single    single
   mscurves   multiple  single
   smcurves   single    multiple
   mmcurves   multiple  multiple

2. Basic evaluation measures (mode = "basic")

   **S3 object**  **# of models**  **# of test datasets**
   sspoints    single    single
   mspoints    multiple  single
   smpoints    single    multiple
   mmpoints    multiple  multiple

3. Fast AUC (ROC) calculation with the U statistic (mode = "aucroc")

   **S3 object**  **# of models**  **# of test datasets**
   aucroc     -         -

See the **Value** section of evalmod for more details.

row.names

Not used by this method.
optional: Not used by this method.

raw_curves: A Boolean value to specify whether raw curves are shown instead of the average curve. It is effective only when raw_curves is set to TRUE of the evalmod function.

... Not used by this method.

Value

The `as.data.frame` function returns a data frame.

See Also

evalmod for generating S3 objects with performance evaluation measures.

Examples

```r
## Not run:
#########################################################
### Single model & single test dataset
###
## Load a dataset with 10 positives and 10 negatives
data(P10N10)

## Generate an sscurve object that contains ROC and Precision-Recall curves
sscurves <- evalmod(scores = P10N10$scores, labels = P10N10$labels)

## Convert sscurves to a data frame
sscurves.df <- as.data.frame(sscurves)

## Show data frame
head(sscurves.df)

## Generate an sspoints object that contains basic evaluation measures
sspoints <- evalmod(mode = "basic", scores = P10N10$scores, labels = P10N10$labels)

## Convert sspoints to a data frame
sspoints.df <- as.data.frame(sspoints)

## Show data frame
head(sspoints.df)

#########################################################
### Multiple models & single test dataset
###
## Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(1, 100, 100, "all")
mdat <- mmdata(samps[['scores']], samps[['labels']],
               modnames = samps[['modnames']])
```
## Generate an mscurve object that contains ROC and Precision-Recall curves

```r
curves <- evalmod(mdat)
```

## Convert mscurves to a data frame

```r
curves.df <- as.data.frame(curves)
```

## Show data frame

```r
head(curves.df)
```

## Generate an mspoints object that contains basic evaluation measures

```r
curves <- evalmod(mdat, mode = "basic")
```

## Convert mspoints to a data frame

```r
points.df <- as.data.frame(curves)
```

## Show data frame

```r
head(points.df)
```

# Single model & multiple test datasets

## Create sample datasets with 100 positives and 100 negatives

```r
samps <- create_sim_samples(10, 100, 100, "good_er")
mdat <- mmdata(samps["scores"], samps["labels"],
    modnames = samps["modnames"],
    dsids = samps["dsids"])
```

## Generate an smcurve object that contains ROC and Precision-Recall curves

```r
curves <- evalmod(mdat, raw_curves = TRUE)
```

## Convert smcurves to a data frame

```r
curves.df <- as.data.frame(curves)
```

## Show data frame

```r
head(curves.df)
```

## Generate an smpoints object that contains basic evaluation measures

```r
points <- evalmod(mdat, mode = "basic")
```

## Convert smpoints to a data frame

```r
points.df <- as.data.frame(points)
```

## Show data frame

```r
head(points.df)
```

# Multiple models & multiple test datasets

```r
```
## Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(10, 100, 100, "all")
mdat <- mmdata(samps["scores"], samps["labels"],
               modnames = samps["modnames"],
               dsids = samps["dsids"])

## Generate an mscurve object that contains ROC and Precision-Recall curves
mmcurves <- evalmod(mdat, raw_curves = TRUE)

## Convert mmcurves to a data frame
mmcurves.df <- as.data.frame(mmcurves)

## Show data frame
head(mmcurves.df)

## Generate an mmpoints object that contains basic evaluation measures
mmpoints <- evalmod(mdat, mode = "basic")

## Convert mmpoints to a data frame
mmpoints.df <- as.data.frame(mmpoints)

## Show data frame
head(mmpoints.df)

# N-fold cross validation datasets

## Load test data
data(M2N50F5)

cvdat <- mmdata(nfold_df = M2N50F5, score_cols = c(1, 2),
                 lab_col = 3, fold_col = 4,
                 modnames = c("m1", "m2"), dsids = 1:5)

## Generate an mmcurve object that contains ROC and Precision-Recall curves
cvcurves <- evalmod(cvdat)

## Convert mmcurves to a data frame
cvcurves.df <- as.data.frame(cvcurves)

## Show data frame
head(cvcurves.df)

## Generate an mmpoints object that contains basic evaluation measures
cvpoints <- evalmod(cvdat, mode = "basic")

## Convert mmpoints to a data frame
cvpoints.df <- as.data.frame(cvpoints)

## Show data frame
head(cvpoints.df)

########################################################################
### AUC with the U statistic
###

## mode = "aucroc"
data(P10N10)
uauc1 <- evalmod(scores = P10N10$scores, labels = P10N10$labels,  
mode="aucroc")

# as.data.frame 'aucroc'
as.data.frame(uauc1)

## mode = "aucroc"
samps <- create_sim_samples(10, 100, 100, "all")
mdat <- mmdata(samps[["scores"]], samps[["labels"]],  
modnames = samps[["modnames"]],
dsids = samps[["dsids"]])
uauc2 <- evalmod(mdat, mode="aucroc")

# as.data.frame 'aucroc'
head(as.data.frame(uauc2))

## End(Not run)

---

auc Retrieve a data frame of AUC scores

Description

The auc function takes an S3 object generated by evalmod and retrieves a data frame with the Area Under the Curve (AUC) scores of ROC and Precision-Recall curves.

Usage

auc(curves)

## S3 method for class 'aucs'
auc(curves)

Arguments

curves An S3 object generated by evalmod. The auc function accepts the following S3 objects.

S3 object # of models # of test datasets
The auc function returns a data frame with AUC scores.

See Also

evalmod for generating S3 objects with performance evaluation measures. pauc for retrieving a dataset of pAUCs.

Examples

# Single model & single test dataset

### Load a dataset with 10 positives and 10 negatives
data(P10N10)

### Generate an sscurve object that contains ROC and Precision-Recall curves
sscurves <- evalmod(scores = P10N10$scores, labels = P10N10$labels)

### Shows AUCs
auc(sscurves)

# Multiple models & single test dataset

### Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(1, 100, 100, "all")
mdat <- mmdata(samps[['scores']], samps[['labels']],
               modnames = samps[['modnames']])

### Generate an mscurve object that contains ROC and Precision-Recall curves
mscurves <- evalmod(mdat)

### Shows AUCs
auc(mscurves)

# Single model & multiple test datasets

###
## Create sample datasets with 100 positives and 100 negatives

```r
samps <- create_sim_samples(4, 100, 100, "good_er")
mdat <- mmdat(samps[['scores']], samps[['labels']],
              modnames = samps[['modnames']],
              dsids = samps[['dsids']])
```

## Generate an smcurve object that contains ROC and Precision-Recall curves

```r
smcurves <- evalmod(mdat, raw_curves = TRUE)
```

## Get AUCs

```r
sm_aucs <- auc(smcurves)
```

## Shows AUCs

```r
sm_aucs
```

## Get AUCs of Precision-Recall

```r
sm_aucs_prc <- subset(sm_aucs, curvetypes == "PRC")
```

## Shows AUCs

```r
sm_aucs_prc
```

## Multiple models & multiple test datasets

## Create sample datasets with 100 positives and 100 negatives

```r
samps <- create_sim_samples(4, 100, 100, "all")
mdat <- mmdat(samps[['scores']], samps[['labels']],
              modnames = samps[['modnames']],
              dsids = samps[['dsids']])
```

## Generate an mcurve object that contains ROC and Precision-Recall curves

```r
mmcurves <- evalmod(mdat, raw_curves = TRUE)
```

## Get AUCs

```r
mm_aucs <- auc(mmcurves)
```

## Shows AUCs

```r
mm_aucs
```

## Get AUCs of Precision-Recall

```r
mm_aucs_prc <- subset(mm_aucs, curvetypes == "PRC")
```

## Shows AUCs

```r
mm_aucs_prc
```
Description

The autoplot function plots performance evaluation measures by using ggplot2 instead of the general R plot.

Usage

```r
## S3 method for class 'sscurves'
autoplot(object, curvetype = c("ROC", "PRC"), ...)

## S3 method for class 'mscurves'
autoplot(object, curvetype = c("ROC", "PRC"), ...)

## S3 method for class 'smcurves'
autoplot(object, curvetype = c("ROC", "PRC"), ...)

## S3 method for class 'mmcurves'
autoplot(object, curvetype = c("ROC", "PRC"), ...)

## S3 method for class 'sspoints'
autoplot(object, curvetype = .get_metric_names("basic"), ...)

## S3 method for class 'mspoints'
autoplot(object, curvetype = .get_metric_names("basic"), ...)

## S3 method for class 'smpoints'
autoplot(object, curvetype = .get_metric_names("basic"), ...)

## S3 method for class 'mmpoints'
autoplot(object, curvetype = .get_metric_names("basic"), ...)
```

Arguments

- `object` An S3 object generated by `evalmod`. The autoplot function accepts the following codeS3 objects for two different modes, "rocprc" and "basic".

1. ROC and Precision-Recall curves (mode = "rocprc")

<table>
<thead>
<tr>
<th>S3 object</th>
<th># of models</th>
<th># of test datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>sscurves</td>
<td>single</td>
<td>single</td>
</tr>
<tr>
<td>mscurves</td>
<td>multiple</td>
<td>single</td>
</tr>
<tr>
<td>smcurves</td>
<td>single</td>
<td>multiple</td>
</tr>
<tr>
<td>mmcurves</td>
<td>multiple</td>
<td>multiple</td>
</tr>
</tbody>
</table>

2. Basic evaluation measures (mode = "basic")

<table>
<thead>
<tr>
<th>S3 object</th>
<th># of models</th>
<th># of test datasets</th>
</tr>
</thead>
</table>
sspoints single single
mspoints multiple single
smpoints single multiple
mmpoints multiple multiple

See the Value section of evalmod for more details.

curvetype A character vector with the following curve types.

1. ROC and Precision-Recall curves (mode = "rocprc")

<table>
<thead>
<tr>
<th>curvetype</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROC</td>
<td>ROC curve</td>
</tr>
<tr>
<td>PRC</td>
<td>Precision-Recall curve</td>
</tr>
</tbody>
</table>

Multiple curvetype can be combined, such as c("ROC", "PRC").

2. Basic evaluation measures (mode = "basic")

<table>
<thead>
<tr>
<th>curvetype</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>Normalized ranks vs. error rate</td>
</tr>
<tr>
<td>accuracy</td>
<td>Normalized ranks vs. accuracy</td>
</tr>
<tr>
<td>specificity</td>
<td>Normalized ranks vs. specificity</td>
</tr>
<tr>
<td>sensitivity</td>
<td>Normalized ranks vs. sensitivity</td>
</tr>
<tr>
<td>precision</td>
<td>Normalized ranks vs. precision</td>
</tr>
<tr>
<td>mcc</td>
<td>Normalized ranks vs. Matthews correlation coefficient</td>
</tr>
<tr>
<td>fscore</td>
<td>Normalized ranks vs. F-score</td>
</tr>
</tbody>
</table>

Multiple curvetype can be combined, such as c("precision", "sensitivity").

Following additional arguments can be specified.

type A character to specify the line type as follows.
"l" lines
"p" points
"b" both lines and points

show_cb A Boolean value to specify whether point-wise confidence bounds are drawn. It is effective only when calc_avg of the evalmod function is set to TRUE.

raw_curves A Boolean value to specify whether raw curves are shown instead of the average curve. It is effective only when raw_curves of the evalmod function is set to TRUE.

show_legend A Boolean value to specify whether the legend is shown.

ret_grob A logical value to indicate whether autoplot returns a grob object.
The grob object is internally generated by arrangeGrob. The grid.draw function takes a grob object and shows a plot. It is effective only when a multiple-panel plot is generated, for example, when curvetype is c("ROC", "PRC").

reduce_points A Boolean value to decide whether the points should be reduced when mode = "rocprc". The points are reduced according to x_bins of the evalmod function. The default values is TRUE.
The autoplot function returns a ggplot object for a single-panel plot and a frame-grob object for a multiple-panel plot.

See Also

evalmod for generating an S3 object. fortify for converting a curves and points object to a data frame. plot for plotting the equivalent curves with the general R plot.

Examples

```r
## Not run:

## Load libraries
library(ggplot2)
library(grid)

#############################################################
## Single model & single test dataset
##
## Load a dataset with 10 positives and 10 negatives
data(P10N10)

## Generate an sscurve object that contains ROC and Precision-Recall curves
sscurves <- evalmod(scores = P10N10$scores, labels = P10N10$labels)

## Plot both ROC and Precision-Recall curves
autoplot(sscurves)

## Reduced/Full supporting points
sampss <- create_sim_samples(1, 50000, 50000)
evalss <- evalmod(scores = sampss$scores, labels = sampss$labels)

# Reduced supporting point
system.time(autoplot(evalss))

# Full supporting points
system.time(autoplot(evalss, reduce_points = FALSE))

## Get a grob object for multiple plots
pp1 <- autoplot(sscurves, ret_grob = TRUE)
plot.new()
grid.draw(pp1)

## A ROC curve
autoplot(sscurves, curvetype = "ROC")

## A Precision-Recall curve
autoplot(sscurves, curvetype = "PRC")

## Generate an sspoints object that contains basic evaluation measures
```r
sspoints <- evalmod(mode = "basic", scores = P10N10$scores, labels = P10N10$labels)

## Normalized ranks vs. basic evaluation measures
autoplot(sspoints)

## Normalized ranks vs. precision
autoplot(sspoints, curvetype = "precision")

#########################################################################
### Multiple models & single test dataset
###
### Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(1, 100, 100, "all")
mdat <- mmdata(samps$scores, samps$labels, modnames = samps$modnames)

## Generate an mscurve object that contains ROC and Precision-Recall curves
mscurves <- evalmod(mdat)

## ROC and Precision-Recall curves
autoplot(mscurves)

## Reduced/Full supporting points
sampms <- create_sim_samples(5, 50000, 50000)
 evalms <- evalmod(scores = sampms$scores, labels = sampms$labels)

# Reduced supporting point
system.time(autoplot(evalms))

# Full supporting points
system.time(autoplot(evalms, reduce_points = FALSE))

## Hide the legend
autoplot(mscurves, show_legend = FALSE)

## Generate an mpoints object that contains basic evaluation measures
mspoints <- evalmod(mdat, mode = "basic")

## Normalized ranks vs. basic evaluation measures
autoplot(mspoints)

## Hide the legend
autoplot(mspoints, show_legend = FALSE)

#########################################################################
### Single model & multiple test datasets
###
### Create sample datasets with 100 positives and 100 negatives
```
samps <- create_sim_samples(10, 100, 100, "good_er")
mdat <- mmdata(samps["scores"], samps["labels"],
               modnames = samps["modnames"],
               dsids = samps["dsids"])

## Generate an smcurve object that contains ROC and Precision-Recall curves
smcurves <- evalmod(mdat, raw_curves = TRUE)

## Average ROC and Precision-Recall curves
autoplot(smcurves, raw_curves = FALSE)

## Hide confidence bounds
autoplot(smcurves, raw_curves = FALSE, show_cb = FALSE)

## Raw ROC and Precision-Recall curves
autoplot(smcurves, raw_curves = TRUE, show_cb = FALSE)

## Reduced/Full supporting points
samps <- create_sim_samples(4, 5000, 5000)
mdatsm <- mmdata(samps$scores, samps$labels, expd_first = "dsids")
evalsm <- evalmod(mdatsm, raw_curves = TRUE)

# Reduced supporting point
system.time(autoplot(evalsm, raw_curves = TRUE))

# Full supporting points
system.time(autoplot(evalsm, raw_curves = TRUE, reduce_points = FALSE))

## Generate an smpoints object that contains basic evaluation measures
smpoints <- evalmod(mdat, mode = "basic")

## Normalized ranks vs. average basic evaluation measures
autoplot(smpoints)

########################################################################
### Multiple models & multiple test datasets
###
## Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(10, 100, 100, "all")
mdat <- mmdata(samps["scores"], samps["labels"],
               modnames = samps["modnames"],
               dsids = samps["dsids"])

## Generate an smcurve object that contains ROC and Precision-Recall curves
mmcurves <- evalmod(mdat, raw_curves = TRUE)

## Average ROC and Precision-Recall curves
autoplot(mmcurves, raw_curves = FALSE)

## Show confidence bounds
autoplot(mmcurves, raw_curves = FALSE, show_cb = TRUE)
```r
## Raw ROC and Precision-Recall curves
autoplot(mmcures, raw_curves = TRUE)

## Reduced/Full supporting points
sampmm <- create_sim_samples(4, 5000, 5000)
mdatmm <- mmdata(sampmm$scores, sampmm$labels, modnames = c("m1", "m2"),
                 dsids = c(1, 2), expd_first = "modnames")
evalmm <- evalmod(mdatmm, raw_curves = TRUE)

# Reduced supporting point
system.time(autoplot(evalmm, raw_curves = TRUE))

# Full supporting points
system.time(autoplot(evalmm, raw_curves = TRUE, reduce_points = FALSE))

## Generate an mmpoints object that contains basic evaluation measures
mmpoints <- evalmod(mdat, mode = "basic")

## Normalized ranks vs. average basic evaluation measures
autoplot(mmpoints)

###############################################################
### N-fold cross validation datasets
###
## Load test data
data(M2N50F5)

## Specify necessary columns to create mdat
cvdat <- mmdata(nfold_df = M2N50F5, score_cols = c(1, 2),
                 lab_col = 3, fold_col = 4,
                 modnames = c("m1", "m2"), dsids = 1:5)

## Generate an mmcurve object that contains ROC and Precision-Recall curves
cvcures <- evalmod(cvdat)

## Average ROC and Precision-Recall curves
autoplot(cvcurves)

## Show confidence bounds
autoplot(cvcurves, show_cb = TRUE)

## Generate an mmpoints object that contains basic evaluation measures
cvpoints <- evalmod(cvdat, mode = "basic")

## Normalized ranks vs. average basic evaluation measures
autoplot(cvpoints)

## End(Not run)
```
B1000

Balanced data with 1000 positives and 1000 negatives.

Description

A list contains labels and scores of five different performance levels. All scores were randomly generated.

Usage

data(B1000)

Format

A list with 8 items.

- `np` number of positives: 1000
- `nn` number of negatives: 1000
- `labels` labels of observed data
- `random_scores` scores of a random performance level
- `poor_er_scores` scores of a poor early retrieval level
- `good_er_scores` scores of a good early retrieval level
- `excel_scores` scores of an excellent level
- `perf_scores` scores of the perfect level

B500

Balanced data with 500 positives and 500 negatives.

Description

A list contains labels and scores of five different performance levels. All scores were randomly generated.

Usage

data(B500)
**create_sim_samples**

Create random samples for simulations

**Format**

A list with 8 items.

- **np** number of positives: 500
- **nn** number of negatives: 500
- **labels** labels of observed data
- **random_scores** scores of a random performance level
- **poor_er_scores** scores of a poor early retrieval level
- **good_er_scores** scores of a good early retrieval level
- **excel_scores** scores of an excellent level
- **perf_scores** scores of the perfect level

**Description**

The `create_sim_samples` function generates random samples with different performance levels.

**Usage**

```r
create_sim_samples(n_repeat, np, nn, score_names = "random")
```

**Arguments**

- **n_repeat** The number of iterations to make samples.
- **np** The number of positives in a sample.
- **nn** The number of negatives in a sample.
- **score_names** A character vector for the names of the following performance levels.
  - "random" Random
  - "poor_er" Poor early retrieval
  - "good_er" Good early retrieval
  - "excel" Excellent
  - "perf" Perfect
  - "all" All of the above

**Value**

The `create_sim_samples` function returns a list with the following items.

- scores: a list of numeric vectors
- labels: an integer vector
- modnames: a character vector of the model names
- dsids: a character vector of the dataset IDs
evalmod

Evaluate models and calculate performance evaluation measures

description

The evalmod function calculates ROC and Precision-Recall curves for specified prediction scores and binary labels. It also calculates several basic performance evaluation measures, such as accuracy, error rate, and precision, by specifying mode as "basic".

Usage

```r
evalmod(mdat, mode = NULL, scores = NULL, labels = NULL, modnames = NULL, dsids = NULL, posclass = NULL, na_worst = TRUE,
```
evalmod

ties_method = "equiv", calc_avg = TRUE, cb_alpha = 0.05, raw_curves = FALSE, x_bins = 1000, ...

Arguments

mdat  An S3 object created by the mmdat function. It contains formatted scores and labels. The evalmod function ignores the following arguments when mdat is specified.

  • scores
  • labels
  • modnames
  • dsids
  • posclass
  • na_worst
  • ties_method

These arguments are internally passed to the mmdat function when mdat is unspecified. In that case, both scores and labels must be at least specified.

mode  A string that specifies the types of evaluation measures that the evalmod function calculates.

  "rocrec" ROC and Precision-Recall curves
  "prcrec" Same as above
  "basic" Normalized ranks vs. accuracy, error rate, specificity, sensitivity, precision, Matthews correlation coefficient, and F-score.
  "aucroc" Fast AUC(ROC) calculation with the U statistic

scores  A numeric dataset of predicted scores. It can be a vector, a matrix, an array, a data frame, or a list. The join_scores function can be useful to make scores with multiple datasets.

labels  A numeric, character, logical, or factor dataset of observed labels. It can be a vector, a matrix, an array, a data frame, or a list. The join_labels function can be useful to make labels with multiple datasets.

modnames  A character vector for the names of the models. The evalmod function automatically generates default names as "m1", "m2", "m3", and so on when it is NULL.

dsids  A numeric vector for test dataset IDs. The evalmod function automatically generates the default ID as 1 when it is NULL.

posclass  A scalar value to specify the label of positives in labels. It must be the same data type as labels. For example, posclass = -1 changes the positive label from 1 to -1 when labels contains 1 and -1. The positive label will be automatically detected when posclass is NULL.

na_worst  A Boolean value for controlling the treatment of NAs in scores.

  TRUE  All NAs are treated as the worst scores
  FALSE  All NAs are treated as the best scores

ties_method  A string for controlling ties in scores.
"equiv"  Ties are equivalently ranked
"first"  Ties are ranked in an increasing order as appeared
"random"  Ties are ranked in random order

calc_avg  A logical value to specify whether average curves should be calculated. It is effective only when dsids contains multiple dataset IDs. For instance, the function calculates the average for the model "m1" when modnames is c("m1", "m1", "m1") and dsids is c(1, 2, 3). The calculation points are defined by x_bins.

cb_alpha  A numeric value with range [0, 1] to specify the alpha value of the point-wise confidence bounds calculation. It is effective only when calc_avg is set to TRUE. For example, it should be 0.05 for the 95% confidence level. The calculation points are defined by x_bins.

raw_curves  A logical value to specify whether all raw curves should be discarded after the average curves are calculated. It is effective only when calc_avg is set to TRUE.

x_bins  An integer value to specify the number of minimum bins on the x-axis. It is then used to define supporting points. For instance, the x-values of the supporting points will be c(0, 0.5, 1) and c(0, 0.25, 0.5, 0.75, 1) when x_bins = 2 and x_bins = 4, respectively. All corresponding y-values of the supporting points are calculated.

...  These additional arguments are passed to mmdata for data preparation.

Value

The evalmod function returns an S3 object that contains performance evaluation measures. The number of models and the number of datasets can be controlled by modnames and dsids. For example, the number of models is "single" and the number of test datasets is "multiple" when modnames = c("m1", "m1", "m1") and dsids = c(1, 2, 3) are specified.

Different S3 objects have different default behaviors of S3 generics, such as plot, autoplot, and fortify.

1. The evalmod function returns one of the following S3 objects when mode is "prcroc". The objects contain ROC and Precision-Recall curves.

<table>
<thead>
<tr>
<th>S3 object</th>
<th># of models</th>
<th># of test datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>sscurves</td>
<td>single</td>
<td>single</td>
</tr>
<tr>
<td>mscurves</td>
<td>multiple</td>
<td>single</td>
</tr>
<tr>
<td>smcurves</td>
<td>single</td>
<td>multiple</td>
</tr>
<tr>
<td>mmcurves</td>
<td>multiple</td>
<td>multiple</td>
</tr>
</tbody>
</table>

2. The evalmod function returns one of the following S3 objects when mode is "basic". They contain five different basic evaluation measures; error rate, accuracy, specificity, sensitivity, and precision.

<table>
<thead>
<tr>
<th>S3 object</th>
<th># of models</th>
<th># of test datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>sspoints</td>
<td>single</td>
<td>single</td>
</tr>
<tr>
<td>mspoints</td>
<td>multiple</td>
<td>single</td>
</tr>
<tr>
<td>spoints</td>
<td>single</td>
<td>multiple</td>
</tr>
<tr>
<td>mmpoints</td>
<td>multiple</td>
<td>multiple</td>
</tr>
</tbody>
</table>
3. The evalmod function returns the aucroc S3 object when mode is "aucroc", which can be used with 'print' and 'as.data.frame'.

See Also

plot for plotting curves with the general R plot. autoplot and fortify for plotting curves with ggplot2. mndata for formatting input data. join_scores and join_labels for formatting scores and labels with multiple datasets. format_nfold for creating n-fold cross validation dataset from data frame. create_sim_samples for generating random samples for simulations.

Examples

```r
# Single model & single test dataset

data(P10N10)

# Generate an sscurve object that contains ROC and Precision-Recall curves
sscurves <- evalmod(scores = P10N10$scores, labels = P10N10$labels)

# Generate an sspoints object that contains basic evaluation measures
sspoints <- evalmod(mode = "basic", scores = P10N10$scores,
                   labels = P10N10$labels)

# Multiple models & single test dataset

# Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(1, 100, 100, "all")
mdat <- mndata(samps["scores"], samps["labels"],
               modnames = samps["modnames"])

# Generate an mscurve object that contains ROC and Precision-Recall curves
mscurves <- evalmod(mdat)

# Generate an mspoints object that contains basic evaluation measures
mspoints <- evalmod(mdat, mode = "basic")

# Single model & multiple test datasets
```
## Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(4, 100, 100, "good_er")
mdat <- mmdata(samps[,"scores"], samps[,"labels"],
  modnames = samps[,"modnames"],
  dsids = samps[,"dsids"])

## Generate an smcurve object that contains ROC and Precision-Recall curves
smcurves <- evalmod(mdat)

## Generate an smpoints object that contains basic evaluation measures
smpoints <- evalmod(mdat, mode = "basic")

## Multiple models & multiple test datasets

## Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(4, 100, 100, "all")
mdat <- mmdata(samps[,"scores"], samps[,"labels"],
  modnames = samps[,"modnames"],
  dsids = samps[,"dsids"])

## Generate an mmcurve object that contains ROC and Precision-Recall curves
mmcurves <- evalmod(mdat)

## Generate an mmpoints object that contains basic evaluation measures
mmpoints <- evalmod(mdat, mode = "basic")

## N-fold cross validation datasets

## Load test data
data(M2N50F5)

cvdat <- mmdata(nfold_df = M2N50F5, score_cols = c(1, 2),
  lab_col = 3, fold_col = 4,
  modnames = c("m1", "m2"), dsids = 1:5)

## Generate an mmcurve object that contains ROC and Precision-Recall curves
cvcurves <- evalmod(cvdat)

## Generate an mmpoints object that contains basic evaluation measures
cvpoints <- evalmod(cvdat, mode = "basic")
```r
## Specify mndata arguments from evalmod
cvcvcurves2 <- evalmod(nfold_df = M2N50F5, score_cols = c(1, 2),
                        lab_col = 3, fold_col = 4,
                        modnames = c("m1", "m2"), dsids = 1:5)
cvcvcurves2

# AUC with the U statistic

## mode = "aucroc" returns 'aucroc' S3 object
data(P10N10)

# 'aucroc' S3 object
uauc1 <- evalmod(scores = P10N10$scores, labels = P10N10$labels,
                  mode="aucroc")

# print 'aucroc'
uauc1

# as.data.frame 'aucroc'
as.data.frame(uauc1)

## It is 2-3 times faster than mode = "rocprc"

## A sample of 100,000
samp1 <- create_sim_samples(1, 50000, 50000)

## a function to test mode = "rocprc"
func_evalmod_rocprc <- function(samp) {
  curves <- evalmod(scores = samp$scores, labels = samp$labels)
  aucs <- auc(curves)
}

## a function to test mode = "aucroc"
func_evalmod_aucroc <- function(samp) {
  aucs <- evalmod(scores = samp$scores, labels = samp$labels, mode="aucroc")
  as.data.frame(aucs)
}

## Process time
system.time(res1 <- func_evalmod_rocprc(samp1))

system.time(res2 <- func_evalmod_aucroc(samp1))

## AUCs
res1
res2
```
format_nfold  

Create n-fold cross validation dataset from data frame

Description
The `format_nfold` function takes a data frame with scores, label, and n-fold columns and converts it to a list for `evalmod` and `mmdata`.

Usage

```r
format_nfold(nfold_df, score_cols, lab_col, fold_col)
```

Arguments

- `nfold_df`: A data frame that contains at least one score column, label and fold columns.
- `score_cols`: A character/numeric vector that specifies score columns of `nfold_df`.
- `lab_col`: A number/string that specifies the label column of `nfold_df`.
- `fold_col`: A number/string that specifies the fold column of `nfold_df`.

Value

The `format_nfold` function returns a list that contains multiple scores and labels.

See Also

- `evalmod` for calculation evaluation measures.
- `mmdata` for formatting input data.
- `join_scores` and `join_labels` for formatting scores and labels with multiple datasets.

Examples

```r
# Convert data frame with 2 models and 5-fold datasets
# Load test data
data(M2N50F5)
head(M2N50F5)

# Convert with format_nfold
nfold_list1 = format_nfold(nfold_df = M2N50F5, score_cols = c(1, 2),
lab_col = 3, fold_col = 4)

# Show the list structure
str(nfold_list1)
str(nfold_list1$scores)
str(nfold_list1$labels)
```
### fortify

**Convert a curves and points object to a data frame for ggplot2**

**Description**

The `fortify` function converts an S3 object generated by `evalmod` to a data frame for `ggplot2`.

**Usage**

```r
## S3 method for class 'sscurves'
fortify(model, raw_curves = NULL, reduce_points = FALSE, ...
```

```r
## S3 method for class 'mscurves'
fortify(model, raw_curves = NULL, reduce_points = FALSE, ...
```

```r
## S3 method for class 'smcurves'
fortify(model, raw_curves = NULL, reduce_points = FALSE, ...
```
## S3 method for class 'mmcurves'
fortify(model, raw_curves = NULL, reduce_points = FALSE, ...)

## S3 method for class 'sspoints'
fortify(model, raw_curves = NULL, reduce_points = FALSE, ...)

## S3 method for class 'mspoints'
fortify(model, raw_curves = NULL, reduce_points = FALSE, ...)

## S3 method for class 'smpoints'
fortify(model, raw_curves = NULL, reduce_points = FALSE, ...)

## S3 method for class 'mmpoints'
fortify(model, raw_curves = NULL, reduce_points = FALSE, ...)

### Arguments

- **model**: An S3 object generated by `evalmod`. The fortify function takes one of the following S3 objects.
  1. ROC and Precision-Recall curves (mode = "rocprc")

<table>
<thead>
<tr>
<th>S3 object</th>
<th># of models</th>
<th># of test datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>sscures</td>
<td>single</td>
<td>single</td>
</tr>
<tr>
<td>mscures</td>
<td>multiple</td>
<td>single</td>
</tr>
<tr>
<td>smcurves</td>
<td>single</td>
<td>multiple</td>
</tr>
<tr>
<td>mmcurves</td>
<td>multiple</td>
<td>multiple</td>
</tr>
</tbody>
</table>

- 2. Basic evaluation measures (mode = "basic")

<table>
<thead>
<tr>
<th>S3 object</th>
<th># of models</th>
<th># of test datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>sspoints</td>
<td>single</td>
<td>single</td>
</tr>
<tr>
<td>mspoints</td>
<td>multiple</td>
<td>single</td>
</tr>
<tr>
<td>smpoints</td>
<td>single</td>
<td>multiple</td>
</tr>
<tr>
<td>mmpoints</td>
<td>multiple</td>
<td>multiple</td>
</tr>
</tbody>
</table>

See the **Value** section of `evalmod` for more details.

- **raw_curves**: A Boolean value to specify whether raw curves are shown instead of the average curve. It is effective only when raw_curves is set to TRUE of the `evalmod` function.

- **reduce_points**: A Boolean value to decide whether the points should be reduced. The points are reduced according to x_bins of the `evalmod` function. The default values is FALSE.

- ... Not used by this method.
Value

The `fortify` function returns a data frame for `ggplot2`.

See Also

`evalmod` for generating S3 objects with performance evaluation measures. `autoplot` for plotting with `ggplot2`.

Examples

```r
## Not run:

## Load library
library(ggplot2)

# Single model & single test dataset

## Load a dataset with 10 positives and 10 negatives
data(P10N10)

## Generate an sscurve object that contains ROC and Precision-Recall curves
sscurves <- evalmod(scores = P10N10$scores, labels = P10N10$labels)

## Let ggplot internally call fortify
p_rocprc <- ggplot(sscurves, aes(x = x, y = y))
p_rocprc <- p_rocprc + geom_line()
p_rocprc <- p_rocprc + facet_wrap(~curvetype)
p_rocprc

## Explicitly fortify sscurves
ssdf <- fortify(sscurves)

## Plot a ROC curve
p_roc <- ggplot(subset(ssdf, curvetype == "ROC"), aes(x = x, y = y))
p_roc <- p_roc + geom_line()
p_roc

## Plot a Precision-Recall curve
p_prc <- ggplot(subset(ssdf, curvetype == "PRC"), aes(x = x, y = y))
p_prc <- p_prc + geom_line()
p_prc

## Generate an sspoints object that contains basic evaluation measures
sspoints <- evalmod(mode = "basic", scores = P10N10$scores, labels = P10N10$labels)

## Fortify sspoints
ssdf <- fortify(sspoints)

## Plot normalized ranks vs. precision
p_prec <- ggplot(subset(ssdf, curvetype == "precision"), aes(x = x, y = y))
```
```r
p_prec <- p_prec + geom_point()
p_prec

#-----------------------------------------------------------------------------------------
### Multiple models & single test dataset
###

## Create sample datasets with 10 positives and 10 negatives
samps <- create_sim_samples(1, 10, 10, "all")
mdat <- mmdata(samps["scores"], samps["labels"],
               modnames = samps["modnames"])

## Generate an mcurve object that contains ROC and Precision-Recall curves
mscurves <- evalmod(mdat)

## Let ggplot internally call fortify
p_rocprc <- ggplot(mscurves, aes(x = x, y = y, color = modname))
p_rocprc <- p_rocprc + geom_line()
p_rocprc <- p_rocprc + facet_wrap(~curvetype)
p_rocprc

## Explicitly fortify mscurves
msdf <- fortify(mscurves)

## Plot ROC curve
df_roc <- subset(msdf, curvetype == "ROC")
p_roc <- ggplot(df_roc, aes(x = x, y = y, color = modname))
p_roc <- p_roc + geom_line()
p_roc

## Fortified data frame can be used for plotting a Precision-Recall curve
df_prc <- subset(msdf, curvetype == "PRC")
p_prc <- ggplot(df_prc, aes(x = x, y = y, color = modname))
p_prc <- p_prc + geom_line()
p_prc

## Generate an mspoints object that contains basic evaluation measures
mspoints <- evalmod(mdat, mode = "basic")

## Fortify mspoints
msdf <- fortify(mspoints)

## Plot normalized ranks vs. precision
df_prec <- subset(msdf, curvetype == "precision")
p_prec <- ggplot(df_prec, aes(x = x, y = y, color = modname))
p_prec <- p_prec + geom_point()
p_prec
```

#-----------------------------------------------------------------------------------------
### Single model & multiple test datasets
###
## Create sample datasets with 10 positives and 10 negatives

```r
samps <- create_sim_samples(5, 10, 10, "good_er")
mdat <- mndata(samps["scores"], samps["labels"],
               modnames = samps["modnames"],
               dsids = samps["dsids"])
```

## Generate an smcurve object that contains ROC and Precision-Recall curves

```r
smcurves <- evalmod(mdat, raw_curves = TRUE)
```

## Let ggplot internally call fortify

```r
p_rocprc <- ggplot(smcurves, aes(x = x, y = y, group = dsid))
p_rocprc <- p_rocprc + geom_smooth(stat = "identity")
p_rocprc <- p_rocprc + facet_wrap(~curvetype)
p_rocprc
```

## Explicitly fortify smcurves

```r
smdf <- fortify(smcurves, raw_curves = FALSE)
```

## Plot average ROC curve

```r
df_roc <- subset(smdf, curvetype == "ROC")
p_roc <- ggplot(df_roc, aes(x = x, y = y, ymin = ymin, ymax = ymax))
p_roc <- p_roc + geom_smooth(stat = "identity")
p_roc
```

## Plot average Precision-Recall curve

```r
df_prc <- subset(smdf, curvetype == "PRC")
p_prc <- ggplot(df_prc, aes(x = x, y = y, ymin = ymin, ymax = ymax))
p_prc <- p_prc + geom_smooth(stat = "identity")
p_prc
```

## Generate an smpoints object that contains basic evaluation measures

```r
smpoints <- evalmod(mdat, mode = "basic")
```

## Fortify smpoints

```r
smdf <- fortify(smpoints)
```

## Plot normalized ranks vs. precision

```r
df_prc <- subset(smdf, curvetype == "precision")
p_prc <- ggplot(df_prc, aes(x = x, y = y, ymin = ymin, ymax = ymax))
p_prc <- p_prc + geom_ribbon(aes(min = ymin, ymax = ymax),
                             stat = "identity", alpha = 0.25,
                             fill = "grey25")
p_prc <- p_prc + geom_point(aes(x = x, y = y))
p_prc
```

### Multiple models & multiple test datasets

```r
## Create sample datasets with 10 positives and 10 negatives
samps <- create_sim_samples(5, 10, 10, "all")
```
Imbalanced data with 1000 positives and 10000 negatives.
Description
A list contains labels and scores of five different performance levels. All scores were randomly generated.

Usage
data(IB1000)

Format
A list with 8 items.

- np: number of positives: 1000
- nn: number of negatives: 10000
- labels: labels of observed data
- random_scores: scores of a random performance level
- poor_er_scores: scores of a poor early retrieval level
- good_er_scores: scores of a good early retrieval level
- excel_scores: scores of an excellent level
- perf_scores: scores of the perfect level

IB500
Imbalanced data with 500 positives and 5000 negatives.

Description
A list contains labels and scores of five different performance levels. All scores were randomly generated.

Usage
data(IB500)

Format
A list with 8 items.

- np: number of positives: 500
- nn: number of negatives: 5000
- labels: labels of observed data
- random_scores: scores of a random performance level
- poor_er_scores: scores of a poor early retrieval level
- good_er_scores: scores of a good early retrieval level
- excel_scores: scores of an excellent level
- perf_scores: scores of the perfect level
join_labels

Join observed labels of multiple test datasets into a list

Description

join_labels takes observed labels and converts them to a list.

Usage

join_labels(..., byrow = FALSE, chklen = TRUE)

Arguments

... Multiple datasets. They can be vectors, arrays, matrices, data frames, and lists.
byrow A Boolean value to specify whether row vectors are used for matrix, data frame, and array.
chklen A Boolean value to specify whether all list items must be the same lengths.

Value

The join_labels function returns a list that contains all combined label data.

See Also

evalmod for calculation evaluation measures. mmdata for formatting input data. join_scores for formatting scores with multiple datasets.

Examples

###############################
### Add three numeric vectors
###
l1 <- c(1, 0, 1, 1)
l2 <- c(1, 1, 0, 0)
l3 <- c(0, 1, 0, 1)
labels1 <- join_labels(l1, l2, l3)

## Show the list structure
str(labels1)

###############################
### Add a matrix and a numeric vector
###
a1 <- matrix(rep(c(1, 0), 4), 4, 2)
labels2 <- join_labels(a1, l3)

## Show the list structure
join_scores

str(labels2)

# Use byrow
a2 <- matrix(rep(c(1, 0), 4), 2, 4, byrow = TRUE)
labels3 <- join_labels(a2, l3, byrow = TRUE)

# Show the list structure
str(labels3)

# Use chklen
l4 <- c(-1, 0, -1)
l5 <- c(0, -1)
labels4 <- join_labels(l4, l5, chklen = FALSE)

# Show the list structure
str(labels4)

---

join_scores  

**Join scores of multiple models into a list**

**Description**

The `join_scores` function takes predicted scores from multiple models and converts them to a list.

**Usage**

```r
join_scores(..., byrow = FALSE, chklen = TRUE)
```

**Arguments**

- `...`: Multiple datasets. They can be vectors, arrays, matrices, data frames, and lists.
- `byrow`: A Boolean value to specify whether row vectors are used for matrix, data frame, and array.
- `chklen`: A Boolean value to specify whether all list items must be the same lengths.

**Value**

The `join_scores` function returns a list that contains all combined score data.

**See Also**

- `evalmod` for calculation evaluation measures.
- `mmdata` for formatting input data.
- `join_labels` for formatting labels with multiple datasets.
Examples

### Add three numeric vectors
```r
s1 <- c(1, 2, 3, 4)
s2 <- c(5, 6, 7, 8)
s3 <- c(2, 4, 6, 8)
scores1 <- join_scores(s1, s2, s3)
```
## Show the list structure
```r
str(scores1)
```

### Add a matrix and a numeric vector
```r
a1 <- matrix(seq(8), 4, 2)
scores2 <- join_scores(a1, s3)
```
## Show the list structure
```r
str(scores2)
```

### Use byrow
```r
a2 <- matrix(seq(8), 2, 4, byrow = TRUE)
scores3 <- join_scores(a2, s3, byrow = TRUE)
```
## Show the list structure
```r
str(scores3)
```

### Use chklen
```r
s4 <- c(1, 2, 3)
s5 <- c(5, 6, 7, 8)
scores4 <- join_scores(s4, s5, chklen = FALSE)
```
## Show the list structure
```r
str(scores4)
```

5-fold cross validation sample.

Description

A data frame contains labels and scores for 5-fold test sets.
The `mmdata` function takes predicted scores and labels and returns an `mdat` object. The `evalmod` function takes an `mdat` object as input data to calculate evaluation measures.

**Usage**
```
mmdata(scores, labels, modnames = NULL, dsids = NULL, posclass = NULL,
na_worst = TRUE, ties_method = "equiv", expd_first = NULL,
mode = "rocprc", nfold_df = NULL, score_cols = NULL, lab_col = NULL,
fold_col = NULL, ...)  
```

**Arguments**
- `scores` A numeric dataset of predicted scores. It can be a vector, a matrix, an array, a data frame, or a list. The `join_scores` function can be useful to make scores with multiple datasets.
- `labels` A numeric, character, logical, or factor dataset of observed labels. It can be a vector, a matrix, an array, a data frame, or a list. The `join_labels` function can be useful to make labels with multiple datasets.
- `modnames` A character vector for the names of the models. The `evalmod` function automatically generates default names as "m1", "m2", "m3", and so on when it is `NULL`.
- `dsids` A numeric vector for test dataset IDs. The `evalmod` function automatically generates the default ID as 1 when it is `NULL`.
- `posclass` A scalar value to specify the label of positives in `labels`. It must be the same data type as `labels`. For example, `posclass = -1` changes the positive label from 1 to -1 when `labels` contains 1 and -1. The positive label will be automatically detected when `posclass` is `NULL`.
- `na_worst` A Boolean value for controlling the treatment of NAs in `scores`. 

The `mmdata` function creates a data frame with the following columns:
- `score1`: 50 random scores
- `score2`: 50 random scores
- `label`: 50 labels as 'pos' or 'neg'
- `fold`: 50 fold IDs as 1:5

**Description**

Usage data is processed by `mmdata` to create the input data for performance evaluation calculation. The `mdat` object can then be used with the `evalmod` function to calculate evaluation measures.
TRUE  All NAs are treated as the worst scores
FALSE  All NAs are treated as the best scores

ties_method  A string for controlling ties in scores.
"equiv"  Ties are equivalently ranked
"first"  Ties are ranked in an increasing order as appeared
"random"  Ties are ranked in random order

expd_first  A string to indicate which of the two variables - model names or test dataset IDs should be expanded first when they are automatically generated.
"modnames"  Model names are expanded first. For example, The mndata function generates modnames as c("m1", "m2") and dsids as c(1, 1) when two vectors are passed as input, and modnames and dsids are unspecified.
"dsids"  Test dataset IDs are expanded first. For example, The mndata function generates modnames as c("m1", "m1") and dsids as c(1, 2) when two vectors are passed as input, and modnames and dsids are unspecified.

mode  A string that specifies the types of evaluation measures that the evalmod function calculates.
"rocrpc"  ROC and Precision-Recall curves
"prcrpc"  Same as above
"basic"  Normalized ranks vs. accuracy, error rate, specificity, sensitivity, precision, Matthews correlation coefficient, and F-score.
"aucroc"  Fast AUC(ROC) calculation with the U statistic

nfold_df  A data frame that contains at least one score column, label and fold columns.
score_cols  A character/numeric vector that specifies score columns of nfold_df.
lab_col  A number/numeric vector that specifies label column of nfold_df.
fold_col  A number/numeric vector that specifies fold column of nfold_df.

Value

The mndata function returns an mdat object that contains formatted labels and score ranks. The object can be used as input data for the evalmod function.

See Also

evalmod for calculation evaluation measures. join_scores and join_labels for formatting scores and labels with multiple datasets. format_nfold for creating n-fold cross validation dataset from data frame.

Examples

```r
# Single model & single test dataset
```
```r
## Load a dataset with 10 positives and 10 negatives
data(P10N10)

## Generate mdat object
ssmdat1 <- mmdat(P10N10$scores, P10N10$labels)
ssmdat1
ssmdat2 <- mmdat(1:8, sample(c(0, 1), 8, replace = TRUE))
ssmdat2

#########################################################################
### Multiple models & single test dataset
###
## Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(1, 100, 100, "all")

## Multiple models & single test dataset
msmdat1 <- mmdat(samps[["scores"]], samps[["labels"]],
                 modnames = samps[["modnames"]])
msmdat1

## Use join_scores and join_labels
s1 <- c(1, 2, 3, 4)
s2 <- c(5, 6, 7, 8)
scores <- join_scores(s1, s2)
l1 <- c(1, 0, 1, 1)
l2 <- c(1, 0, 1, 1)
labels <- join_labels(l1, l2)

msmdat2 <- mmdat(scores, labels, modnames = c("ms1", "ms2"))
msmdat2

#########################################################################
### Single model & multiple test datasets
###
## Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(10, 100, 100, "good_er")

## Single model & multiple test datasets
smmdat <- mmdat(samps[["scores"]], samps[["labels"]],
                modnames = samps[["modnames"]],
                dsids = samps[["dsids"]])
smmdat

#########################################################################
### Multiple models & multiple test datasets
###
```
## Create sample datasets with 100 positives and 100 negatives

```r
samps <- create_sim_samples(10, 100, 100, "all")
```

## Multiple models & multiple test datasets

```r
mmmdat <- mmdata(samps[["scores"]], samps[["labels"]],
                 modnames = samps[["modnames"]],
                 dsids = samps[["dsids"]])
```

### N-fold cross validation datasets

### Load test data

```r
data(M2N50F5)
head(M2N50F5)
```

### Specify necessary columns to create mdat

```r
cvdat1 <- mmdata(nfold_df = M2N50F5, score_cols = c(1, 2),
                 lab_col = 3, fold_col = 4,
                 modnames = c("m1", "m2"), dsids = 1:5)
cvdat1
```

### Use column names

```r
cvdat2 <- mmdata(nfold_df = M2N50F5, score_cols = c("score1", "score2"),
                 lab_col = "label", fold_col = "fold",
                 modnames = c("m1", "m2"), dsids = 1:5)
cvdat2
```

---

P10N10  

A small example dataset with several tied scores.

---

**Description**

A list contains labels and scores for 10 positives and 10 negatives.

**Usage**

```r
data(P10N10)
```

**Format**

A list with 4 items.

- `np` number of positives: 10
- `nn` number of negatives: 10
- `labels` 20 labels of observed data
- `scores` 20 scores with some ties
Calculate partial AUCs

Description

The `part` function takes an S3 object generated by `evalmod` and calculate partial AUCs and Standardized partial AUCs of ROC and Precision-Recall curves. Standardized pAUCs are standardized to the range between 0 and 1.

Usage

```
part(curves, xlim, ylim, curvetype)
```

```
## S3 method for class 'sscurves'
part(curves, xlim = c(0, 1), ylim = c(0, 1),
     curvetype = c("ROC", "PRC"))

## S3 method for class 'mscurves'
part(curves, xlim = c(0, 1), ylim = c(0, 1),
     curvetype = c("ROC", "PRC"))

## S3 method for class 'smcurves'
part(curves, xlim = c(0, 1), ylim = c(0, 1),
     curvetype = c("ROC", "PRC"))

## S3 method for class 'mmcurves'
part(curves, xlim = c(0, 1), ylim = c(0, 1),
     curvetype = c("ROC", "PRC"))
```

Arguments

- `curves` An S3 object generated by `evalmod`. The `part` function accepts the following S3 objects.

```
S3 object  # of models  # of test datasets
sscurves   single      single
mscurves   multiple    single
smcurves   single      multiple
mmcurves   multiple    multiple
```

See the Value section of `evalmod` for more details.

- `xlim` A numeric vector of length two to specify x range between two points in [0, 1]

- `ylim` A numeric vector of length two to specify y range between two points in [0, 1]

- `curvetype` A character vector with the following curve types.

```
curvetype  description
```
Multiple curvetype can be combined, such as `c("ROC", "PRC")`.

### Value

The `part` function returns the same S3 object specified as input with calculated pAUCs and standardized pAUCs.

### See Also

`evalmod` for generating S3 objects with performance evaluation measures. `pauc` for retrieving a dataset of pAUCs.

### Examples

```r
## Not run:

## Load library
library(ggplot2)

#---------------------------------------------------------------
# Single model & single test dataset
#---------------------------------------------------------------

## Load a dataset with 10 positives and 10 negatives
data(P10N10)

## Generate an sscurve object that contains ROC and Precision-Recall curves
sscurves <- evalmod(scores = P10N10$scores, labels = P10N10$labels)

## Calculate partial AUCs
sscurves.part <- part(sscurves, xlim = c(0.25, 0.75))

## Show AUCs
sscurves.part

## Plot partial curve
plot(sscurves.part)

## Plot partial curve with ggplot
autoplot(sscurves.part)

#---------------------------------------------------------------
# Multiple models & single test dataset
#---------------------------------------------------------------

## Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(1, 100, 100, "all")
mdat <- mmdata(samps["scores"], samps["labels"],
```
modnames = samps["modnames"]

## Generate an mscurve object that contains ROC and Precision-Recall curves
mscurves <- evalmod(mdat)

## Calculate partial AUCs
mscurves.part <- part(mscurves, xlim = c(0, 0.75), ylim = c(0.25, 0.75))

## Show AUCs
mscurves.part

## Plot partial curves
plot(mscurves.part)

## Plot partial curves with ggplot
autoplot(mscurves.part)

#########################################################################
### Single model & multiple test datasets
###
## Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(4, 100, 100, "good_er")
mdat <- mmdata(samps["scores"], samps["labels"],
               modnames = samps["modnames"],
               dsids = samps["dsids"])

## Generate an smcurve object that contains ROC and Precision-Recall curves
smcurves <- evalmod(mdat)

## Calculate partial AUCs
smcurves.part <- part(smcurves, xlim = c(0.25, 0.75))

## Show AUCs
smcurves.part

## Plot partial curve
plot(smcurves.part)

## Plot partial curve with ggplot
autoplot(smcurves.part)

#########################################################################
### Multiple models & multiple test datasets
###
## Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(4, 100, 100, "all")
mdat <- mmdata(samps["scores"], samps["labels"],
               modnames = samps["modnames"],
               dsids = samps["dsids"])
The `auc` function takes an S3 object generated by `part` and `evalmod` and retrieves a data frame with the partial AUC scores of ROC and Precision-Recall curves.

### Usage

```r
pauc(curves)
```

```r
## S3 method for class 'aucs'
pauc(curves)
```

### Arguments

- **curves**: An S3 object generated by `part` and `evalmod`. The `pauc` function accepts the following S3 objects.

<table>
<thead>
<tr>
<th>S3 object</th>
<th># of models</th>
<th># of test datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>sscurves</td>
<td>single</td>
<td>single</td>
</tr>
<tr>
<td>mscurves</td>
<td>multiple</td>
<td>single</td>
</tr>
<tr>
<td>smcurves</td>
<td>single</td>
<td>multiple</td>
</tr>
<tr>
<td>mmcurves</td>
<td>multiple</td>
<td>multiple</td>
</tr>
</tbody>
</table>

See the `Value` section of `evalmod` for more details.
pauc

Value
The auc function returns a data frame with pAUC scores.

See Also
evalmod for generating S3 objects with performance evaluation measures. part for calculation of pAUCs. auc for retrieving a dataset of AUCs.

Examples

# Single model & single test dataset
data(P10N10)
scores = P10N10$scores, labels = P10N10$labels
sscurves.part <- part(sscurves, xlim = c(0.25, 0.75))
pauc(sscurves.part)

# Multiple models & single test dataset
mdat <- mmdat(samps[["scores"]], samps[["labels"]],
modnames = samps[["modnames"]])
mscurves.part <- part(mscurves, xlim = c(0, 0.75), ylim = c(0.25, 0.75))
pauc(mscurves.part)

# Single model & multiple test datasets
mdat <- mmdat(samps[["scores"]], samps[["labels"]],
modnames = samps[["modnames"]])
mscurves.part <- part(mscurves, xlim = c(0, 0.75), ylim = c(0.25, 0.75))
pauc(mscurves.part)
modnames = samps["modnames"],
dsids = samps["dsids"]

## Generate an smcurve object that contains ROC and Precision-Recall curves
smcurves <- evalmod(mdat, raw_curves = TRUE)

## Calculate partial AUCs
smcurves.part <- part(smcurves, xlim = c(0.25, 0.75))

## Shows pAUCs
pauc(smcurves.part)

# Multiple models & multiple test datasets
###
### Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(4, 100, 100, "all")
mdat <- mmdata(samps["scores"], samps["labels"],
               modnames = samps["modnames"],
dsids = samps["dsids"])

## Generate an mcurve object that contains ROC and Precision-Recall curves
mmcurves <- evalmod(mdat, raw_curves = TRUE)

## Calculate partial AUCs
mmcurves.part <- part(mmcurves, xlim = c(0, 0.25))

## Shows pAUCs
pauc(mmcurves.part)

---

**plot**

*Plot performance evaluation measures*

**Description**

The `plot` function creates a plot of performance evaluation measures.

**Usage**

```r
## S3 method for class 'sscurves'
plot(x, y = NULL, ...)
```

```r
## S3 method for class 'mscurves'
plot(x, y = NULL, ...)
```

```r
## S3 method for class 'smcurves'
plot(x, y = NULL, ...)
```
Arguments

x An S3 object generated by `evalmod`. The `plot` function accepts the following S3 objects.

1. ROC and Precision-Recall curves (mode = "rocprc")

<table>
<thead>
<tr>
<th>S3 object</th>
<th># of models</th>
<th># of test datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>sscures</td>
<td>single</td>
<td>single</td>
</tr>
<tr>
<td>mscures</td>
<td>multiple</td>
<td>single</td>
</tr>
<tr>
<td>smcurves</td>
<td>single</td>
<td>multiple</td>
</tr>
<tr>
<td>mmcurves</td>
<td>multiple</td>
<td>multiple</td>
</tr>
</tbody>
</table>

2. Basic evaluation measures (mode = "basic")

<table>
<thead>
<tr>
<th>S3 object</th>
<th># of models</th>
<th># of test datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>sspoints</td>
<td>single</td>
<td>single</td>
</tr>
<tr>
<td>mspoints</td>
<td>multiple</td>
<td>single</td>
</tr>
<tr>
<td>smpoints</td>
<td>single</td>
<td>multiple</td>
</tr>
<tr>
<td>mmpoints</td>
<td>multiple</td>
<td>multiple</td>
</tr>
</tbody>
</table>

See the `Value` section of `evalmod` for more details.

y Equivalent with `curvetype`.

... All the following arguments can be specified.

curvetype 1. ROC and Precision-Recall curves (mode = "rocprc")

<table>
<thead>
<tr>
<th>curvetype</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROC</td>
<td>ROC curve</td>
</tr>
<tr>
<td>PRC</td>
<td>Precision-Recall curve</td>
</tr>
</tbody>
</table>

Multiple `curvetype` can be combined, such as `c("ROC", "PRC")`.

2. Basic evaluation measures (mode = "basic")
**curvetype**  
*description*

- **error**: Normalized ranks vs. error rate
- **accuracy**: Normalized ranks vs. accuracy
- **specificity**: Normalized ranks vs. specificity
- **sensitivity**: Normalized ranks vs. sensitivity
- **precision**: Normalized ranks vs. precision
- **mcc**: Normalized ranks vs. Matthews correlation coefficient
- **fscore**: Normalized ranks vs. F-score

Multiple curvetype can be combined, such as `c("precision", "sensitivity")`.

**type**  
A character to specify the line type as follows.

- "l" lines
- "p" points
- "b" both lines and points

**show_cb**  
A Boolean value to specify whether point-wise confidence bounds are drawn. It is effective only when `calc_avg` of the `evalmod` function is set to TRUE.

**raw_curves**  
A Boolean value to specify whether raw curves are shown instead of the average curve. It is effective only when `raw_curves` of the `evalmod` function is set to TRUE.

**show_legend**  
A Boolean value to specify whether the legend is shown.

**Value**

The `plot` function shows a plot and returns NULL.

**See Also**

- `evalmod` for generating an S3 object. `autoplot` for plotting the equivalent curves with `ggplot2`.

**Examples**

```r
## Not run:

### Single model & single test dataset

## Load a dataset with 10 positives and 10 negatives
data(P10N10)

## Generate an sscurve object that contains ROC and Precision-Recall curves
sscurves <- evalmod(scores = P10N10$scores, labels = P10N10$labels)

## Plot both ROC and Precision-Recall curves
plot(sscurves)

## Plot a ROC curve
plot(sscurves, curvetype = "ROC")
```
## Plot a Precision-Recall curve

```r
plot(sscurves, curvetype = "PRC")
```

## Generate an sspoints object that contains basic evaluation measures

```r
sspoints <- evalmod(mode = "basic", scores = P10N10$scores,
                        labels = P10N10$labels)
```

## Plot normalized ranks vs. basic evaluation measures

```r
plot(sspoints)
```

## Plot normalized ranks vs. precision

```r
plot(sspoints, curvetype = "precision")
```

### Multiple models & single test dataset

```r
## Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(1, 100, 100, "all")
mdat <- mmdata(samps["scores"], samps["labels"],
               modnames = samps["modnames"])

## Generate an mcurve object that contains ROC and Precision-Recall curves
mscurves <- evalmod(mdat)

## Plot both ROC and Precision-Recall curves
plot(mscurves)

## Hide the legend
plot(mscurves, show_legend = FALSE)

## Generate an mpoints object that contains basic evaluation measures
mspoints <- evalmod(mdat, mode = "basic")

## Plot normalized ranks vs. basic evaluation measures
plot(mspoints)

## Hide the legend
plot(mspoints, show_legend = FALSE)

### Single model & multiple test datasets

```r
## Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(10, 100, 100, "good_er")
mdat <- mmdata(samps["scores"], samps["labels"],
               modnames = samps["modnames"],
               dsids = samps["dsids"])
```
## Generate an smcurve object that contains ROC and Precision-Recall curves
smcurves <- evalmod(mdat, raw_curves = TRUE)

## Plot average ROC and Precision-Recall curves
plot(smcurves, raw_curves = FALSE)

## Hide confidence bounds
plot(smcurves, raw_curves = FALSE, show_cb = FALSE)

## Plot raw ROC and Precision-Recall curves
plot(smcurves, raw_curves = TRUE, show_cb = FALSE)

## Generate an smpoints object that contains basic evaluation measures
smpoints <- evalmod(mdat, mode = "basic")

## Plot normalized ranks vs. average basic evaluation measures
plot(smpoints)

### Multiple models & multiple test datasets

## Create sample datasets with 100 positives and 100 negatives
samps <- create_sim_samples(10, 100, 100, "all")
mdat <- mndata(samps["scores"], samps["labels"],
               modnames = samps["modnames"],
               dsids = samps["dsids"])

## Generate an mscurve object that contains ROC and Precision-Recall curves
mmcurves <- evalmod(mdat, raw_curves = TRUE)

## Plot average ROC and Precision-Recall curves
plot(mmcurves, raw_curves = FALSE)

## Show confidence bounds
plot(mmcurves, raw_curves = FALSE, show_cb = TRUE)

## Plot raw ROC and Precision-Recall curves
plot(mmcurves, raw_curves = TRUE)

## Generate an mmpoints object that contains basic evaluation measures
mmpoints <- evalmod(mdat, mode = "basic")

## Plot normalized ranks vs. average basic evaluation measures
plot(mmpoints)

### N-fold cross validation datasets

## Load test data
data(M2N50F5)

### Specify necessary columns to create mdat
cvdat <- mmdata(nfold_df = M2N50F5, score_cols = c(1, 2),
                   lab_col = 3, fold_col = 4,
                   modnames = c("m1", "m2"), dsids = 1:5)

### Generate an mmcurve object that contains ROC and Precision-Recall curves
cvcurves <- evalmod(cvdat)

### Average ROC and Precision-Recall curves
plot(cvcurves)

### Show confidence bounds
plot(cvcurves, show_cb = TRUE)

### Generate an mmpoints object that contains basic evaluation measures
cvpoints <- evalmod(cvdat, mode = "basic")

### Normalized ranks vs. average basic evaluation measures
plot(cvpoints)

### End(Not run)

---

**precrec**  
*

**precrec: A package for computing accurate ROC and Precision-Recall curves**

---

**Description**

The precrec package contains several functions and S3 generics to provide a robust platform for performance evaluation of binary classifiers.

**Functions**

The precrec package provides the following six functions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>evalmod</td>
<td>Main function to calculate evaluation measures</td>
</tr>
<tr>
<td>mmdata</td>
<td>Reformat input data for performance evaluation calculation</td>
</tr>
<tr>
<td>join_scores</td>
<td>Join scores of multiple models into a list</td>
</tr>
<tr>
<td>join_labels</td>
<td>Join observed labels of multiple test datasets into a list</td>
</tr>
<tr>
<td>create_sim_samples</td>
<td>Create random samples for simulations</td>
</tr>
<tr>
<td>format_nfold</td>
<td>Create n-fold cross validation dataset from data frame</td>
</tr>
</tbody>
</table>
### S3 generics

The precrec package provides eight different S3 generics for the S3 objects generated by the `evalmod` function.

<table>
<thead>
<tr>
<th>S3 generic</th>
<th>Library</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>print</code></td>
<td><code>base</code></td>
<td>Print the calculation results and the summary of the test data</td>
</tr>
<tr>
<td><code>as.data.frame</code></td>
<td><code>base</code></td>
<td>Convert a precrec object to a data frame</td>
</tr>
<tr>
<td><code>plot</code></td>
<td><code>graphics</code></td>
<td>Plot performance evaluation measures</td>
</tr>
<tr>
<td><code>autoplot</code></td>
<td><code>ggplot2</code></td>
<td>Plot performance evaluation measures with ggplot2</td>
</tr>
<tr>
<td><code>fortify</code></td>
<td><code>ggplot2</code></td>
<td>Prepare a data frame for ggplot2</td>
</tr>
<tr>
<td><code>auc</code></td>
<td><code>precrec</code></td>
<td>Make a data frame with AUC scores</td>
</tr>
<tr>
<td><code>part</code></td>
<td><code>precrec</code></td>
<td>Calculate partial curves and partial AUC scores</td>
</tr>
<tr>
<td><code>pauc</code></td>
<td><code>precrec</code></td>
<td>Make a data frame with pAUC scores</td>
</tr>
</tbody>
</table>

### Performance measure calculations

The `evalmod` function calculates ROC and Precision-Recall curves and returns an S3 object. The generated S3 object can be used with several different S3 generics, such as `print` and `plot`. The `evalmod` function can also calculate basic evaluation measures - error rate, accuracy, specificity, sensitivity, precision, Matthews correlation coefficient, and F-Score.

### Data preparation

The `mmdata` function creates an input dataset for the `evalmod` function. The generated dataset contains formatted scores and labels. `join_scores` and `join_labels` are helper functions to combine multiple scores and labels. The `create_sim_samples` function creates test datasets with five different performance levels.

### Data visualization

`plot` takes an S3 object generated by `evalmod` as input and plot corresponding curves. `autoplot` uses ggplot to plot curves.

### Result retrieval

`as.data.frame` takes an S3 object generated by `evalmod` as input and and returns a data frame with calculated curve points. `auc` and `pauc` returns a data frame with AUC scores and partial AUC scores, respectively.
Index

*Topic datasets
  B1000, 16
  B500, 16
  IB1000, 30
  IB500, 31
  M2N50F5, 34
  P10N10, 38

arrangeGrob, 11
as.data.frame, 2, 50
auc, 7, 43, 50
autoplot, 9, 20, 21, 27, 46, 50

B1000, 16
B500, 16

create_sim_samples, 17, 21, 49, 50

evalmod, 2–4, 7, 8, 10–12, 18, 18, 24–27, 32, 33, 35, 36, 39, 40, 42, 43, 45, 46, 49, 50

format_nfold, 21, 23, 36, 49

fortify, 12, 20, 21, 25, 50

grid.draw, 11

IB1000, 30
IB500, 31

join_labels, 19, 21, 24, 32, 33, 35, 36, 49, 50

join_scores, 19, 21, 24, 32, 33, 35, 36, 49, 50

M2N50F5, 34

mmdata, 18–21, 24, 32, 33, 35, 49, 50

P10N10, 38

part, 39, 42, 43, 50

pauc, 8, 40, 42, 50

plot, 12, 20, 21, 44, 50

precrec, 49

precrec-package (precrec), 49