Package 'screenr'

May 8, 2022

Type Package

Title Construction of Binary Test-Screening Rules

Version 0.5.0

Date 2022-05-09

Depends R(>= 3.5.0)

Imports glmpath, pROC, dplyr, scales, stringr, epiR, magrittr

Description Package screenr enables easy development and validation of diagnostic test screening tools. It is designed to enable those with only a basic familiarity with R to develop, validate and implement screening tools for diagnostic tests. Consider the situation where a definitive test for some condition is relatively expensive, and the condition is rare. In that case, universal testing would not be efficient in terms of the yield of postive results per test performed. Now suppose that responses to a set of simple diagnostic questions or observations may be predictive of the definitive test result. Package screenr enables estimation of thresholds for making decisions about when to perform the definitive test on newly observed subjects based on Receiver Operating Characteristics (ROC) estimated from an initial sample. The choice of a particular screening threshold is left to the user, and should be based on careful consideration of application-specific tradeoffs between sensitivity (true positive fraction) and specificity (true negative fraction).

License MIT + file LICENSE

URL https://github.com/sgutreuter/screenr/

Encoding UTF-8

RoxygenNote 7.1.2

BugReports https://github.com/sgutreuter/screenr/issues

LazyData true

Suggests rmarkdown,
knitr

VignetteBuilder knitr

R topics documented:

coef.lasso_screenr — *An S3 Method to Extract Coefficients from* lasso_screenr *Objects*

Description

coef.lasso_screenr returns the regularized logistic model parameter estimates from the AIC- and BIC-best fits from lasso_screenr-class objects.

Usage

```
## S3 method for class 'lasso_screenr'
coef(object, ..., intercept = TRUE, or = FALSE)
```

Arguments

object	an object of class lasso_screenr.
...	optional arguments passed to predict methods.
intercept	(logical) retain (TRUE, default) or drop (FALSE) the intercept coefficients.
or	return odds ratios if TRUE; logit-scale coefficients are the default.

Details

coef.lasso_screenr extracts the estimated coefficients from lasso_screenr objects. Regularization does not support estimation of confidence limits.

Value

coef.lasso_screenr returns a dataframe containing the estimated coefficients (or odds ratios) from the AIC- and BIC-best logistic regression models, where *p* is the number of coefficients.

Examples

```
attach(uniobj1)
coef(uniobj1)
```

coef.logreg_screenr — *An S3 Method to Extract Coefficients from* logreg_screenr *Objects*

Description

coef.logreg_screenr returns the logistic model parameter estimates from and profile-likelihood confidence limits from logreg_screenr-class objects.

Usage

```
## S3 method for class 'logreg_screenr'
coef(object, ..., intercept = TRUE, or = FALSE, conf_level = 0.95, digits = 4)
```

Arguments

`object`	an object of class `logreg_screenr`.
`...`	optional arguments passed to `predict` methods.
`intercept`	(logical) retain (`TRUE`, default) or drop (`FALSE`) the intercept coefficients.
`or`	return odds ratios if `TRUE`. Default: FALSE (returns logit-scale coefficients).
`conf_level`	confidence level for profile-likelihood confidence intervals. Default: 0.95.
`digits`	number of decimal places to be printed. Default: 4.

Details

`coef.logreg_screenr` extracts the estimated coefficients from `logreg_screenr` objects.

Value

`coef.logreg_screenr` returns a dataframe containing the estimated coefficients (or odds ratios) and their profile-likelihood lower and upper confidence limits (lcl and ucl, respectively).

Examples

```
attach(uniobj2)
coef(uniobj2, or = TRUE)
```

---

easy_tool *Simplifying Screening from* `lasso_screenr` *or* `logreg_screenr` *Objects*

Description

`easy_tool` rescales model coefficients to whole numbers ranging from 1 to `max` (`QuestionWeights`). Those rescaled and rounded coefficients can be used as weights for each screening question in a simplified model-based screening tool. The test screening score is the sum of the weights for each subject.

Usage

```
easy_tool(object, max = 3, model = c("minAIC", "minBIC"), crossval = TRUE, ...)
```

Arguments

object	an object of class lasso_screenr or logreg_screenr.
max	(numeric) the desired maximum value for the response weights (default is 3).
model	(for lasso_screenr objects only) the desired basis model. Valid options are "minAIC" (the default) and "minBIC".
crossval	a (logical) indicator for cross-validated (TRUE) or in-sample (FALSE) performance evaluation.
...	additional arguments passed to coef.lasso_screenr or coef.logreg_screenr

Details

The QuestionWeights (see Value, below) are the foundation for easy screening. For example, the screening tool could consist of a simple questionnaire followed by the weight for each question, expressed as a small whole number (1, ..., max) and/or an equal number of open circles. The person doing the screening need only circle the numerical weight and/or fill in the circles if and only if the subject provides a "yes" response to a particular question. The person doing the screening then obtains the final score for that subject by adding up the circled numbers or counting the total number of filled-in circles. Testing is mandatory for consenting subjects for whom that final score equals or exceeds the chosen threshold based on the receiver-operating characteristics of CVresults.

The value chosen for max involves a trade-off between the ease of manual scoring and the degree to which the ROC from the re-scaling matches the ROC from the model. Small values of max make manual scoring easy, and sufficiently large values will match the screening performance of the model fit. A value of 3 may be a reasonable compromise. It is prudent to compare the ROCs from a few values of max with the ROC from the model and base the final choice on the trade-off between ease of manual scoring and the desired combination of sensitivity and specificity.

Value

easy_tool returns (invisibly) an object of class easy_tool containing:

Call The call to easy_tool.

varname The names of the response and predictor variables.

QuestionWeights Weights for the screening questions obtained by rescaling the non-zero-valued logistic regression coefficients to whole numbers ranging from 1 to max.

Type The type of test performance evaluaion ("cross-validated" or "in-sample").

Scores A data frame containing the testing outcomes (response) and cross-validated scores obtained as the sums of the weighted responses to the set of screening questions (score).

ROC An object of class roc containing the receiver-operating characteristic produced by `pROC::roc`.

Note

Execute methods(class = "easy_tool") to see available methods.

See Also

rescale_to_int, ntpp.easy_tool, plot.easy_tool, print.easy_tool and summary.easy_tool

Examples

```
attach(uniobj1)
tool <- easy_tool(uniobj1, max = 3)
class(tool)
```

---

get_what *S3 Methods for Extraction of Object Components*

Description

`get_what` extracts components from objects.

Usage

```
get_what(from, what, ...)
```

Arguments

`from`	an object from which to extract `what`.
`what`	the element to extract from `from`.
`...`	additional arguments.

Value

`get_what` returns the object specified by `what`.

See Also

`get_what.easy_tool`, `get_what.lasso_screenr`, `get_what.logreg_screenr` and `get_what.simple_screenr`.

---

get_what.easy_tool *An S3 Method for Extraction of Components from* `easy_tool` *Objects*

Description

`get_what.easy_tool` extracts components from `easy_tool`-class objects.

Usage

```
## S3 method for class 'easy_tool'
get_what(
  from = NULL,
  what = NULL,
  ...,
  bootreps = 4000,
  conf_level = 0.95,
  se_min = 0.8
)
```

Arguments

from	the easy_tool-class object from which to extract the component.
what	the (character) name of the component to extract. Valid values are "Call", "QuestionWeights", "ROCci", "ROC" and "Scores".
...	optional arguments to get_what methods.
bootreps	the number of bootstrap replications for estimation of confidence intervals for what = "ROCci". Default: 4000.
conf_level	(optional) confidence level for what = ROCci
se_min	minimum value of sensitivity printed for what = ROCci. Default: 0.8.

Details

get_what is provided to enable easy extraction of components that are not provided by the plot, predict, print or summary methods.

Valid values of what are:

"Call" returns the function call that created from.

"QuestionWeights" returns the screening question weights, which are the re-scaled logistic-regression coefficients.

ROCci returns a data frame containing sensitivities, specificities and their confidence limits, and thresholds

"Scores" returns the screening scores for each subject, which are the sums of the products of the binary question responses and their QuestionWeights

"ROC" returns the receiver-operating characteristic for the Scores

Value

get_what.easy_tool returns (invisibly) the object specified by what.

Examples

```
## Not run:
attach(uniobj1)
tool <- easy_tool(uniobj1, max = 3, crossval = TRUE)
## Get and print sensitivities and specificities at thresholds for the
##   local maxima of the ROC curve
ROCci <- get_what(from = tool, what = "ROCci")
print(ROCci)

## End(Not run)
```

`get_what.lasso_screenr` *An S3 Method for Extraction of Components from* `lasso_screenr` *Objects*

Description

`get_what.lasso_screenr` extracts components from `lasso_screenr`-class objects.

Usage

```
## S3 method for class 'lasso_screenr'
get_what(
  from = NULL,
  what = c("glmpathObj", "ROCci", "cvROC", "isROC"),
  ...,
  model = c("minAIC", "minBIC"),
  conf_level = 0.95,
  bootreps = 4000,
  se_min = 0.8
)
```

Arguments

`from`	the `lasso_screenr`-class object from which to extract the component.
`what`	the character-valued name of the component to extract. Valid values are `"glmpathObj"`, `"ROCci"`, `"cvROC"` and `"isROC"`.
`...`	optional arguments to `get_what` methods.
`model`	the character-valued name of the model for which the component is desired. Valid values are `"minAIC"` and `"minBIC"`. Default: `"minAIC"`.
`conf_level`	confidence level for `what = "ROCci"`. Default: 0.95.
`bootreps`	the number of bootstrap replications for estimation of confidence intervals for `what = "ROCci"`. Default: 4000.
`se_min`	minimum value of sensitivity printed for `what = ROCci`. Default: 0.8.

Details

`get_what` is provided to enable easy extraction of components that are not provided by the `coef`, `plot`, `predict`, `print` or `summary` methods.

The following values of `what` return:

`"glmpathObj"` the entire `glmpath`-class object produced by by `glmpath`.

`ROCci` a data frame containing cross-validated sensitivities, specificities and their confidence limits, and thresholds.

"cvROC" the roc-class object produced by roc containing the k-fold cross-validated receiver-operating characteristic.

"isROC" the roc-class object produced by roc containing the in-sample (overly optimistic) receiver-operating characteristic.

Value

get_what.lasso_screenr returns (invisibly) the object specified by what.

Examples

```
## Not run:
attach(uniobj1)
## Plot the coefficient paths
pathobj <- get_what(from = uniobj1, what = "glmpathObj", model = "minAIC")
plot(pathobj)
## Get and print cross-validated sensitivities and specificities at
##   thresholds for the local maxima of the ROC curve
cvROCci <- get_what(from = uniobj1,  what = "ROCci", model = "minBIC")
print(cvROCci)

## End(Not run)
```

---

get_what.logreg_screenr *An S3 Method for Extraction of Components from* logreg_screenr *Objects*

Description

get_what.logreg_screenr extracts components from logreg_screenr-class objects.

Usage

```
## S3 method for class 'logreg_screenr'
get_what(
  from = NULL,
  what = c("ModelFit", "ROCci", "cvROC", "isROC"),
  ...,
  conf_level = 0.95,
  bootreps = 4000,
  se_min = 0.8
)
```

Arguments

from	the logreg_screenr-class object from which to extract the component.
what	the (character) name of the component to extract. Valid values are "ModelFit", "ROCci", "cvROC" and "isROC".
...	optional arguments to get_what methods.
conf_level	(optional) confidence level for what = "ROCci". Default: 0.95.
bootreps	the number of bootstrap replications for estimation of confidence intervals for what = "ROCci". Default: 4000.
se_min	minimum value of sensitivity printed for what = ROCci. Default: 0.8.

Details

get_what is provided to enable easy extraction of components for those who wish to perform computations that are not provided by the coef, plot, predict, print or summary methods.

The following values of what return:

"ModelFit" the entire glm-class object produced by by glm.

ROCci a data frame containing cross-validated sensitivities, specificities and their confidence limits, and thresholds.

"cvROC" the roc-class object produced by roc containing the *k*-fold cross-validated receiver-operating characteristic.

"isROC" the roc-class object produced by roc containing the in-sample (overly optimistic) receiver-operating characteristic.

Value

get_what.logreg_screenr returns (invisibly) the object specified by what.

Examples

```
## Not run:
attach(uniobj2)
## Get and print cross-validated sensitivities and specificities at
##   thresholds for the local maxima of the ROC curve
myROCci <- get_what(from = uniobj2, what = "ROCci")
print(myROCci)

## End(Not run)
```

---

get_what.simple_screenr *An S3 Method for Extraction of Components from* simple_screenr *Objects*

Description

get_what.simple_screenr extracts components from simple_screenr-class objects.

Usage

```
## S3 method for class 'simple_screenr'
get_what(
  from = NULL,
  what = c("ROCci", "isROC"),
  ...,
  conf_level = 0.95,
  bootreps = 4000,
  se_min = 0.6
)
```

Arguments

from	the simple_screenr-class object from which to extract the component.
what	the (character) name of the component to extract. Valid values are "ROCci" and "isROC".
...	optional arguments to get_what methods.
conf_level	(optional) confidence level for what = "ROCci". Default: 09.5.
bootreps	the number of bootstrap replications for estimation of confidence intervals for what = "ROCci". Default: 4000.
se_min	minimum value of sensitivity printed for what = ROCci. Default: 0.6.

Details

get_what is provided to enable easy extraction of components for those who wish to perform computations that are not provided by the plot, predict, print or summary methods.

The following values of what return:

"isROC" the roc-class object produced by roc containing the in-sample (overly optimistic) receiver-operating characteristic.

"ROCci" a data frame containing cross-validated sensitivities, specificities and their confidence limits, and thresholds.

Value

get_what.simple_screenr returns (invisibly) the object specified by what.

Examples

```
## Not run:
data(unicorns)
too_simple <- simple_screenr(testresult ~ Q1 + Q2 + Q3 + Q4 + Q5 + Q6 + Q7,
                             data = unicorns)
too_simple_roc <- get_what(from = too_simple, what = "isROC" )
plot(too_simple_roc)

## End(Not run)
```

---

inverse_link *Compute the Inverses of Binomial Link Functions*

Description

`inverse_link` returns the inverse of logit, cloglog and probit link functions for a linear predictor

Usage

```
inverse_link(lp = NULL, link = c("logit", "cloglog", "probit"))
```

Arguments

`lp`	numeric vector containing the estimated link.
`link`	(character) name of the link function (one of "`logit`", "`cloglog`" or "`probit`").

Details

`inverse_link` returns the inverses of logit, cloglog and probit link functions, and is provided as a (laborious) way to compute predicted values from the `ModelFit` component of `logreg_screenr`-class objects. The `predict` methods are a better way to obtain predicted values.

Value

`inverse_link` returns a numeric vector containing the inverse of the link function for the linear predictor.

See Also

`predict.logreg_screenr`

Examples

```
## Make predictions of probability of infection from new observations
attach(uniobj2)
new_corns <- data.frame(ID = c("Alice D.", "Bernie P."),
                        testresult = c(NA, NA), Q1 = c(0, 0), Q2 = c(0, 0),
                        Q3 = c(0, 0), Q4 = c(0, 0), Q5 = c(0, 1), Q6 = c(0, 1 ),
                        Q7 = c(0, 1))
mfit <- get_what(from = uniobj2 , what = "ModelFit")
coefs <- mfit$coefficients
lp <- as.matrix(cbind(rep(1, nrow(new_corns)), new_corns[, 3:9])) %*%
           as.matrix(coefs, ncol =  1)
(preds <- inverse_link(lp, link = "logit"))
## Note that only the predicted values are returned.
```

`lasso_screenr` *Fitting Screening Tools Using Lasso-Like Regularization of Logistic Regression*

Description

`lasso_screenr` is a convenience function which combines logistic regression using $L1$ regularization, k-fold cross-validation, and estimation of the receiver-operating characteristic (ROC). The in-sample and out-of-sample performance is estimated from the models which produced the minimum AIC and minimum BIC. Execute `methods(class = "lasso_screenr")` to identify available methods.

Usage

```
lasso_screenr(
  formula,
  data = NULL,
  Nfolds = 10,
  L2 = TRUE,
  partial_auc = c(0.8, 1),
  partial_auc_focus = "sensitivity",
  partial_auc_correct = TRUE,
  boot_n = 4000,
  conf_level = 0.95,
  standardize = FALSE,
  seed = Sys.time(),
  ...
)
```

Arguments

`formula` an object of class `stats::formula` defining the testing outcome and predictor variables.

- `data` a dataframe containing the variables defined in `formula`. The testing outcome must be binary (0 = no/negative, 1 = yes/positive) or logical (`FALSE`/`TRUE`). The the predictor variables are are typically binary or logical responses to questions which may be predictive of the test result, but numeric variables can also be used.
- `Nfolds` the number of folds used for *k*-fold cross validation. Default = 10; minimum = 2, maximum = 100.
- `L2` (logical) switch controlling penalization using the *L*2 norm of the parameters. Default: `TRUE`).
- `partial_auc` either a logical `FALSE` or a numeric vector of the form `c(left, right)` where left and right are numbers in the interval [0, 1] specifying the endpoints for computation of the partial area under the ROC curve (pAUC). The total AUC is computed if `partial_auc` = `FALSE`. Default: `c(0.8, 1.0)`
- `partial_auc_focus` one of `"sensitivity"` or `specificity`, specifying for which the pAUC should be computed. `partial_auc.focus` is ignored if `partial_auc` = `FALSE`. Default: `"sensitivity"`.
- `partial_auc_correct` logical value indicating whether the pAUC should be transformed the interval from 0.5 to 1.0. `partial_auc_correct` is ignored if `partial_auc` = `FALSE`. Default: `TRUE`).
- `boot_n` number of bootstrap replications for computation of confidence intervals for the (partial)AUC. Default: 4000.
- `conf_level` a number between 0 and 1 specifying the confidence level for confidence intervals for the (partial)AUC. Default: 0.95.
- `standardize` logical; if TRUE predictors are standardized to unit variance. Default: FALSE (sensible for binary and logical predictors).
- `seed` random number generator seed for cross-validation data splitting.
- `...` additional arguments passed to `glmpath`, `roc`, `auc` or `ci` .

Details

`lasso_screenr` uses the *L*1 path regularizer of Park and Hastie (2007), as implemented in the `glmpath` package. Park-Hastie regularization is is similar to the conventional lasso and the elastic net. It differs from the lasso with the inclusion of a very small, *fixed* (`1e-5`) penalty on the *L*2 norm of the parameter vector, and differs from the elastic net in that the *L*2 penalty is fixed. Like the elastic net, the Park-Hastie regularization is robust to highly correlated predictors. The *L*2 penalization can be turned off (`L2 = FALSE`), in which case the regularization is similar to the coventional lasso. Like all *L*1 regularizers, the Park-Hastie algorithm automatically "deletes" covariates by shrinking their parameter estimates to 0.

The coefficients produced by *L*1 regularization are biased toward zero. Therefore one might consider refitting the model selected by regularization using maximum-likelihood estimation as implemented in `logreg_screenr`.

The receiver-operating characteristics are computed using the `pROC` package.

Out-of-sample performance is estimated using *k*-fold cross-validation. For a gentle but Python-centric introduction to *k*-fold cross-validation, see https://machinelearningmastery.com/k-fold-cross-validation/

Value

lasso_screenr returns (invisibly) an object of class lasso_screenr containing the components:

Call The function call.

Prevalence Prevalence of the binary response variable.

glmpathObj An object of class glmpath returned by glmpath::glmpath. See help(glmpath) and methods(class = "glmpath").

Xmat The matrix of predictors.

isResults A list structure containing the results from the two model fits which produced the minimum AIC and BIC values, respectively. The results consist of Coefficients (the logit-scale parameter estimates, including the intercept), isPreds (the in-sample predicted probabilities) and isROC (the in-sample receiver-operating characteristic (ROC) of class roc).

RNG Specification of the random-number generator used for k-fold data splitting.

RNGseed RNG seed.

cvResults A list structure containing the results of k- fold cross-validation estimation of out-of-sample performance.

The list elements of cvResutls are:

Nfolds the number folds k

X_ho the matrix of held-out predictors for each cross-validation fold

minAICcvPreds the held-out responses and out-of-sample predicted probabilities from AIC-best model selection

minAICcvROC the out-of-sample ROC object of class roc from AIC-best model selection

minBICcvPreds the held-out responses and out-of-sample predicted probabilities from BIC-best model selection

minBICcvROC the corresponding out-of-sample predicted probabilities and ROC object from BIC-best model selection

References

Park MY, Hastie T. $L1$-regularization path algorithm for generalized linear models. Journal of the Royal Statistical Society Series B. 2007;69(4):659-677. https://doi.org/10.1111/j.1467-9868.2007.00607.x

Kim J-H. Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. Computational Statistics and Data Analysis. 2009:53(11):3735-3745. http://doi.org/10.1016/j.csda.2009.04.009

Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez J-C, Muller M. pROC: An open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics. 2011;12(77):1-8. http://doi.org/10.1186/1471-2105-12-77

See Also

glmpath, roc, auc

Examples

```
## Not run:
data(unicorns)
help(unicorns)
uniobj1 <- lasso_screenr(testresult ~ Q1 + Q2 + Q3 + Q4 + Q5 + Q6 + Q7,
                          data = unicorns, Nfolds = 10)
summary(uniobj1)

## End(Not run)
```

---

logreg_screenr *Fitting Screening Tools Using Ordinary Logistic Regression*

Description

logreg_screenr is a convenience function which integrates ordinary logistic regression *k*-fold cross-validation and estimation of the receiver-operating characteristic.

Usage

```
logreg_screenr(
  formula,
  data = NULL,
  link = c("logit", "cloglog", "probit"),
  Nfolds = 10,
  partial_auc = c(0.8, 1),
  partial_auc_focus = "sensitivity",
  partial_auc_correct = TRUE,
  boot_n = 4000,
  conf_level = 0.95,
  seed = Sys.time(),
  ...
)
```

Arguments

formula	an object of class stats::formula defining the testing outcome and predictor covariates, which is passed to stats::glm().
data	a dataframe containing the variables defined in formula. The testing outcome must be binary (0,1) indicating negative and positive test results, respectively, or logical (TRUE/FALSE). The covariates are typically binary (0 = no, 1 = yes) responses to questions which may be predictive of the test result, but any numeric or factor covariates can be used.
link	the character-valued name of the link function for logistic regression. Choices are "logit", "cloglog" or "probit". Default: "logit".

Nfolds number of folds used for *k*-fold cross validation (minimum = 2, maximum = 100). Default: 10.

partial_auc either a logical FALSE or a numeric vector of the form c(left, right) where left and right are numbers in the interval [0, 1] specifying the endpoints for computation of the partial area under the ROC curve (pAUC). The total AUC is computed if partial_auc = FALSE. Default: c(0.8, 1.0).

partial_auc_focus
one of "sensitivity" or specificity, specifying for which the pAUC should be computed. partial_auc_focus is ignored if partial_auc = FALSE. Default: "sensitivity".

partial_auc_correct
logical value indicating whether the pAUC should be transformed the interval from 0.5 to 1.0. partial_auc_correct is ignored if partial_auc = FALSE. Default: TRUE).

boot_n Number of bootstrap replications for computation of confidence intervals for the (partial)AUC. Default: 4000.

conf_level a number between 0 and 1 specifying the confidence level for confidence intervals for the (partial)AUC. Default: 0.95.

seed random-number generator seed for cross-validation data splitting.

... additional arguments passsed to or from other stats::glm or pROC::roc.

Details

The results provide information from which to choose a probability threshold above which individual out-of-sample probabilies indicate the need to perform a diagnostic test. Out-of-sample performance is estimated using *k*-fold cross validation.

The receiver operating characteristics are computed using the pROC package. See References and package documentation for additional details.

For a gentle but python-centric introduction to *k*-fold cross-validation, see https://machinelearningmastery.com/k-fold-cross-validation/.

Value

logreg_screenr returns an object of class logreg_screenr containing the elements:

Call The function call.

formula The formula object.

Prevalence Prevalence (proportion) of the test condition in the training sample.

ModelFit An object of class glm (See glm) containing the results of the model fit.

ISroc An object of class roc containing the "in-sample" (overly-optimistic) receiver operating characteristics, and additional functions for use with this object are available in the pROC package.

CVpreds An object of class cv.predictions containing the data and cross-validated predicted condition y.

CVroc An object of class roc containing the *k*-fold cross-validated "out-of-sample" receiver operating characteristics, and additional functions for use with this object are available in the pROC package.

CVcoef the estimated coefficients from cross-validation

X_ho the matrix of held-out predictors for each cross-validation fold

Note

logreg_screenr is intended mainly for comparison with lasso_screenr. Careful manual model selection is required with logreg_screenr. lasso_screenr is easier and should generally produce better results.

References

Kim J-H. Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. Computational Statistics and Data Analysis. 2009:53(11):3735-3745. http://doi.org/10.1016/j.csda.2009.04.009

Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez J-C, Muller M. pROC: An open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics. 2011;12(77):1-8. http://doi.org/10.1186/1471-2105-12-77

See Also

glm

Examples

```
## Not run:
data(unicorns)
help(unicorns)
uniobj2 <- logreg_screenr(testresult ~ Q1 + Q2 + Q3 + Q4 + Q5 + Q6 + Q7,
                           data = unicorns, link = "logit", Nfolds = 10)
summary(uniobj2)

## End(Not run)
```

---

nnt_ *Compute the Ratio of Total Tests Performed Per Postive Result*

Description

nnt_ computes the anticipated average number of tests performed in order to observe a positive test result.

Usage

```
nnt_(dframe)
```

Arguments

dframe	a dataframe containing columns sensitivities, specificities and prev.

Value

nnt_ returns adataframe containing sensitivity, specificity, the anticipated average number of tests required to observe a single positive test result ntpp, and the prevalence among those screened out of testing pre_untested.

---

ntpp — *An S3 Method to Compute the Ratio of Total Tests to Positive Results*

Description

ntpp computes the ratio of the total number of tests performed per positive test result.

Usage

```
ntpp(object, ...)
```

Arguments

object	an object from which to compute the number of tests per test positive test results.
...	additional arguments.

Details

The anticipated number of tests required to detect a single positive *nntp* is given by

$$nntp = (SeP + (1 - Sp)(1 - P))/SeP$$

where *Se* is sensitivity, *P* is prevalence and *Sp* is specificity. The anticipated prevalence among those screened out is given by

$$Puntested = ((1 - Se)P)/((1 - Se)P + Sp(1 - P))$$

Value

ntpp returns a dataframe containing the following columns:

sensitivity The sensitivity (proportion) of the screener.

specificity The specificity (proportion) of the screener.

ntpp the number of tests required to discover a single positive test result.

prev_untested The prevalence proportion of the test condition among those who are screened out of testing.

See Also

ntpp.lasso_screenr ntpp.logreg_screenr ntpp.simple_screenr ntpp.default `ntpp.data.frame` *Compute the Ratio of Total Tests to Positive Results from a Data Frame*

Description

`ntpp.data.frame` computes the ratio of the total number of tests performed per positive test result from data frames.

Usage

```
## S3 method for class 'data.frame'
ntpp(object, ...)
```

Arguments

`object`	a dataframe containing columns named `sensitivity`, `specificity` and `prev`.
`...`	optional arguments to `ntpp` methods.

Details

The anticipated number of tests required to detect a single positive *nntp* is given by

$$nntp = (SeP + (1 - Sp)(1 - P))/SeP$$

where *Se* is sensitivity, *P* is prevalence and *Sp* is specificity. The anticipated prevalence among those screened out is given by

$$Puntested = ((1 - Se)P)/((1 - Se)P + Sp(1 - P))$$

Value

`ntpp.easy_tool` returns a data frame containing the following columns:

`sensitivity` the sensitivity (proportion)

`specificity` the specificity (proportion)

`prev` prevalence proportion of the test condition

`ntpp` anticipated total tests required per positive result

`prev_untested` anticipated prevalence proportion among the untested ntpp.default *Compute the Ratio of Total Tests to Positive Results from a Data Frame*

Description

ntpp.data.frame computes the ratio of the total number of tests performed per positive test result from data frames.

Usage

```
## Default S3 method:
ntpp(object = NULL, ..., se = NULL, sp = NULL, prev = NULL)
```

Arguments

object	unused, specify se, sp and prev
...	optional arguments to ntpp methods.
se	a numeric vector of sensitivities in (0,1)
sp	a numeric vector of sensitivities in (0,1)
prev	a numeric vector of prevalences of the testing condition, in (0,1)

Details

The anticipated number of tests required to detect a single positive *nntp* is given by

$$nntp = (SeP + (1 - Sp)(1 - P))/SeP$$

where *Se* is sensitivity, *P* is prevalence and *Sp* is specificity. The anticipated prevalence among those screened out is given by

$$Puntested = ((1 - Se)P)/((1 - Se)P + Sp(1 - P))$$

Value

ntpp.default returns a data frame containing the following columns:

sensitivity the sensitivity (proportion)

specificity the specificity (proportion)

prev prevalence proportion of the test condition

ntpp anticipated total tests required per positive result

prev_untested anticipated prevalence proportion among the untested

ntpp.easy_tool — *Compute the Ratio of Total Tests to Positive Results from* easy_tool *Objects*

Description

`ntpp.easy_tool` computes the ratio of the total number of tests performed per positive test result from `easy_tool`-class objects.

Usage

```
## S3 method for class 'easy_tool'
ntpp(object, ..., prev = NULL)
```

Arguments

`object`	an `easy_tool`-class object produced by `easy_tool`.
`...`	optional arguments to `ntpp` methods.
`prev`	an optional prevalence proportion for the test outcome; if missing the prevalence is obtained from `object`.

Details

The anticipated number of tests required to detect a single positive *nntp* is given by

$$nntp = (SeP + (1 - Sp)(1 - P))/SeP$$

where *Se* is sensitivity, *P* is prevalence and *Sp* is specificity. The anticipated prevalence among those screened out is given by

$$Puntested = ((1 - Se)P)/((1 - Se)P + Sp(1 - P))$$

Value

`ntpp.easy_tool` returns a dataframe containing the following columns:

`sensitivity` The sensitivity (proportion) of the screener.

`specificity` The specificity (proportion) of the screener.

`ntpp` the number of tests required to discover a single positive test result.

`prev_untested` The prevalence proportion of the test condition among those who are screened out of testing.

Examples

```
attach(uniobj1)
tool <- easy_tool(uniobj1, max = 3, crossval = TRUE)
ntpp(tool)
```

ntpp.lasso_screenr *Compute the Ratio of Total Tests to Positive Results from* lasso_screenr *Objects*

Description

ntpp.lasso_screenr computes the ratio of the total number of tests performed per positive test result from lasso_screenr-class objects.

Usage

```
## S3 method for class 'lasso_screenr'
ntpp(
  object,
  ...,
  model = c("minAIC", "minBIC"),
  type = c("cvResults", "isResults"),
  prev = NULL
)
```

Arguments

object	a lasso_screenr-class object produced by lasso_screenr.
...	optional arguments to ntpp methods.
model	(character) select the model which produced the minimum AIC ("minAIC", the default) or minimum BIC ("minBIC").
type	(character) one of "cvResults" (the default) or "isResults" to specify *k*-fold cross-validated or in-sample receiver-operating characteristics, respectively.
prev	an optional prevalence proportion for the test outcome; if missing the prevalence is obtained from object.

Details

The anticipated number of tests required to detect a single positive *nntp* is given by

$$nntp = (SeP + (1 - Sp)(1 - P))/SeP$$

where *Se* is sensitivity, *P* is prevalence and *Sp* is specificity. The anticipated prevalence among those screened out is given by

$$Puntested = ((1 - Se)P)/((1 - Se)P + Sp(1 - P))$$

Value

`ntpp.lasso_screenr` returns a data frame containing the following columns:

`sensitivity` The sensitivity (proportion) of the screener.

`specificity` The specificity (proportion) of the screener.

`ntpp` the number of tests required to discover a single positive test result.

`prev_untested` The prevalence proportion of the test condition among those who are screened out of testing.

Examples

```
attach(uniobj1)
ntpp(uniobj1)
```

`ntpp.logreg_screenr` *Compute the Ratio of Total Tests to Positive Results from* `logreg_screenr` *Objects*

Description

`ntpp.logreg_screenr` computes the ratio of the total number of tests performed per positive test result from `logreg_screenr`-class objects.

Usage

```
## S3 method for class 'logreg_screenr'
ntpp(object, ..., type = c("cvResults", "isResults"), prev = NULL)
```

Arguments

`object`	a `logreg_screenr`-class object produced by `logreg_screenr`.
`...`	optional arguments to `ntpp` methods.
`type`	(character) one of `"cvResults"` (the default) or `"isResults"` to specify *k*-fold cross-validated or in-sample receiver-operating characteristics, respectively.
`prev`	an optional prevalence proportion for the test outcome; if missing the prevalence is obtained from `object`.

Details

The anticipated number of tests required to detect a single positive *nntp* is given by

$$nntp = (SeP + (1 - Sp)(1 - P))/SeP$$

where *Se* is sensitivity, *P* is prevalence and *Sp* is specificity. The anticipated prevalence among those screened out is given by

$$Puntested = ((1 - Se)P)/((1 - Se)P + Sp(1 - P))$$

Value

`ntpp.logreg_screenr` returns a data frame containing the following columns:

`sensitivity` The sensitivity (proportion) of the screener.

`specificity` The specificity (proportion) of the screener.

`ntpp` the number of tests required to discover a single positive test result.

`prev_untested` The prevalence proportion of the test condition among those who are screened out of testing.

Examples

```
attach(uniobj2)
ntpp(uniobj2)
```

`ntpp.simple_screenr` *Compute the Ratio of Total Tests to Positive Results from* `simple_screenr` *Objects*

Description

`ntpp.simple_screenr` computes the ratio of the total number of tests performed per positive test result from `simple_screenr`-class objects.

Usage

```
## S3 method for class 'simple_screenr'
ntpp(object, ..., prev = NULL)
```

Arguments

`object`	a `simple_screenr`-class object produced by `simple_screenr`.
`...`	optional arguments to `ntpp` methods.
`prev`	an optional prevalence proportion for the test outcome; if missing the prevalence is obtained from `object`.

Details

The anticipated number of tests required to detect a single positive *nntp* is given by

$$nntp = (SeP + (1 - Sp)(1 - P))/SeP$$

where *Se* is sensitivity, *P* is prevalence and *Sp* is specificity. The anticipated prevalence among those screened out is given by

$$Puntested = ((1 - Se)P)/((1 - Se)P + Sp(1 - P))$$

Value

ntpp.simple_screenr returns data frame containing the following columns:

sensitivity The sensitivity (proportion) of the screener.

specificity The specificity (proportion) of the screener.

ntpp the number of tests required to discover a single positive test result.

prev_untested The prevalence proportion of the test condition among those who are screened out of testing.

---

plot.easy_tool *Plot ROC Curves from* easy_tool*-Class Objects*

Description

plot.easy_tool plots the *k*-fold cross-validated receiver-operating characteristics, including confidence intervals on the combinations of the local maxima of sensitivity and specificity.

Usage

```
## S3 method for class 'easy_tool'
plot(
  x,
  ...,
  plot_ci = TRUE,
  conf_level = 0.95,
  bootreps = 4000,
  print_auc = TRUE,
  partial_auc = c(0.8, 1),
  partial_auc_focus = c("sensitivity", "specificity"),
  partial_auc_correct = TRUE,
  type = "S"
)
```

Arguments

x	an object of class easy_tool.
...	any additional arguments passed to pROC::plot.roc or pROC::lines.roc.
plot_ci	(logical) plot confidence intervals if TRUE.
conf_level	confidence level
bootreps	the number of bootstrap replications for estimation of confidence intervals. Default: 4000.
print_auc	logical indicator for printing the area under the ROC curve (AUC) on the plot. Default: TRUE.

partial_auc
: One of FALSE or a length two numeric vector of the form c(a, b) where a and b are the endpoints of the interval over which to compute the partial AUC (pAUC). Ignored if print_auc = FALSE. Default: c(0.8, 1).

partial_auc_focus
: one of "sensitivity" or "specificity", indicating the measure for which the partial AUC is to be computed. Default: "specificity".

partial_auc_correct
: logical indictor for transformation of the pAUC to fall within the range from 0.5 (random guess) to 1.0 (perfect classification). Default: TRUE.

type
: type of plot. See plot. Default: "S".

Details

plot.easy_tool is an enhanced convenience wrapper for pROC::plot.roc.

Value

This function produces a plot as a side effect and (optionally) returns a dataframe containing sensitivities, specificities and their lower and upper confidence limits for threshold values of Pr(response = 1).

References

Fawcett T. An introduction to ROC analysis. Pattern Recognition Letters. 2006. 27(8):861-874. https://doi.org/10.1016/j.patrec.2005.10.010

Linden A. Measuring diagnostic and predictive accuracy in disease management: an introduction to receiver operating characteristic (ROC) analysis. Journal of Evaluation in Clinical Practice. 2006; 12(2):132-139. https://onlinelibrary.wiley.com/doi/epdf/10.1111/j.1365-2753.2005.00598.x

Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez J-C, Muller M. pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics 2011; 12:77. https://www.biomedcentral.com/1471-2105/12/77

Examples

```
attach(uniobj1)
tool <- easy_tool(uniobj1, max = 3, crossval = TRUE)
plot(tool)
```

---

plot.lasso_screenr *Plot ROC Curves from lasso_screenr-Class Objects*

Description

plot.lasso_screenr plots the *k*-fold cross-validated receiver-operating characteristic for out-of-sample screening performance, including confidence intervals on the combinations of the local maxima of sensitivity and specificity.

Usage

```
## S3 method for class 'lasso_screenr'
plot(
  x,
  ...,
  plot_ci = TRUE,
  model = c("minAIC", "minBIC"),
  conf_level = 0.95,
  bootreps = 4000,
  print_auc = TRUE,
  partial_auc = c(0.8, 1),
  partial_auc_focus = c("sensitivity", "specificity"),
  partial_auc_correct = TRUE,
  type = "S"
)
```

Arguments

`x`	an object of class `lasso_screenr`.
`...`	any additional arguments passed to `pROC::plot.roc` or `pROC::lines.roc`.
`plot_ci`	(logical) plot confidence intervals if `TRUE`. Default: `TRUE`.
`model`	(character) select either the model which produced the minimum AIC (`"minAIC"`) or minimum BIC (`"minBIC"`). Default: `minAIC`,
`conf_level`	confidence level. Default: 0.95.
`bootreps`	the number of bootstrap replications for estimation of confidence intervals. Default: 4000.
`print_auc`	logical indicator for printing the area under the ROC curve (AUC) on the plot. Default: `TRUE`.
`partial_auc`	One of `FALSE` or a length two numeric vector of the form `c(a, b)` where `a` and `b` are the endpoints of the interval over which to compute the partial AUC (pAUC). Ignored if `print_auc = FALSE`. Default: `c(0.8, 1)`.
`partial_auc_focus`	one of `"sensitivity"` or `"specificity"`, indicating the measure for which the partial AUC is to be computed. Default: `"specificity"`.
`partial_auc_correct`	logical indictor for transformation of the pAUC to fall within the range from 0.5 (random guess) to 1.0 (perfect classification). Default: `TRUE`.
`type`	type of plot. See `plot`. Default: `"S"`.

Details

Plot cross-validated (out-of-sample) ROC curve with pointwise confidence intevals along with the overly optimistic in-sample ROC curve. `plot.lasso_screenr` is an enhanced convenience wrapper for `pROC::plot.roc`.

Value

This function produces a plot as a side effect.

References

Fawcett T. An introduction to ROC analysis. Pattern Recognition Letters. 2006. 27(8):861-874. https://doi.org/10.1016/j.patrec.2005.10.010

Linden A. Measuring diagnostic and predictive accuracy in disease management: an introduction to receiver operating characteristic (ROC) analysis. Journal of Evaluation in Clinical Practice. 2006; 12(2):132-139. https://onlinelibrary.wiley.com/doi/epdf/10.1111/j.1365-2753.2005.00598.x

Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez J-C, Muller M. pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics 2011; 12:77. https://www.biomedcentral.com/1471-2105/12/77

Examples

```
## Not run:
attach(uniobj1)
plot(uniobj1, model = "minAIC")

## End(Not run)
```

---

plot.logreg_screenr   *Plot ROC Curves from* logreg_screenr*-Class Objects*

Description

plot.logreg_screenr plots the *k*-fold cross-validated receiver-operating characteristic for out-of-sample screening performanc, including confidence intervals on the combinations of the local maxima of sensitivity and specificity.

Usage

```
## S3 method for class 'logreg_screenr'
plot(
  x,
  ...,
  plot_ci = TRUE,
  conf_level = 0.95,
  bootreps = 4000,
  print_auc = TRUE,
  partial_auc = c(0.8, 1),
  partial_auc_focus = c("sensitivity", "specificity"),
  partial_auc_correct = TRUE,
  type = "S"
)
```

Arguments

x — an object of class logreg_screenr.

... — additional arguments passed to plot.roc and friends.

plot_ci — logical indicator for plotting point-wise confidence intervals at the locally maximum subset of coordinates for on sensitivity and specificity. Default: TRUE). See also ci.thresholds.

conf_level — confidence level in the interval (0,1). Default: 0.95.

bootreps — number of bootstrap replications for estimation of confidence intervals. Default: 4000.

print_auc — logical indicator for printing the area under the ROC curve (AUC) on the plot. Default: TRUE.

partial_auc — One of FALSE or a length two numeric vector of the form c(a, b) where a and b are the endpoints of the interval over which to compute the out-of-sample partial AUC (pAUC). Ignored if print_auc = FALSE. Default: c(0.8, 1).

partial_auc_focus — one of "sensitivity" or "specificity", indicating the measure for which the out-of-sample partial AUC is to be computed. Default: "specificity".

partial_auc_correct — logical indictor for transformation of the pAUC to fall within the range from 0.5 (random guess) to 1.0 (perfect classification). Default: TRUE.

type — type of plot. See plot. Default: "S".

Details

Plot cross-validated (out-of-sample) ROC curve with pointwise confidence intevals along with the overly optimistic in-sample ROC curve. plot.lasso_screenr is an enhanced convenience wrapper for pROC::plot.roc.

Value

This function produces a plot as a side effect.

References

Fawcett T. An introduction to ROC analysis. Pattern Recognition Letters. 2006. 27(8):861-874. https://doi.org/10.1016/j.patrec.2005.10.010

Linden A. Measuring diagnostic and predictive accuracy in disease management: an introduction to receiver operating characteristic (ROC) analysis. Journal of Evaluation in Clinical Practice. 2006; 12(2):132-139. https://onlinelibrary.wiley.com/doi/epdf/10.1111/j.1365-2753.2005.00598.x

Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez J-C, Muller M. pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics 2011; 12:77. https://www.biomedcentral.com/1471-2105/12/77

Examples

```
## Not run:
attach(uniobj2)
plot(uniobj2)

## End(Not run)
```

plot.simple_screenr *Plot ROC Curves from* simple_screenr-*Class Objects*

Description

plot.simple_screenr plots the *k*-fold cross-validated receiver-operating characteristic, including confidence intervals on the combinations of the local maxima of sensitivity and specificity.

Plot ROC curve with pointwise 95 intevals on sensitivity and specificity and (optionally) returns a dataframe containing numerical values.

Usage

```
## S3 method for class 'simple_screenr'
plot(
  x,
  ...,
  plot_ci = TRUE,
  conf_level = 0.95,
  bootreps = 4000,
  print_auc = TRUE,
  partial_auc = c(0.8, 1),
  partial_auc_focus = c("sensitivity", "specificity"),
  partial_auc_correct = TRUE,
  type = "S"
)
```

Arguments

x	an object of class simple_screenr.
...	additional arguments for \link{plot} or passed to \link{plot.roc} and friends.
plot_ci	logical indicator for plotting point-wise confidence intervals at the locally maximum subset of coordinates for on sensitivity and specificity. Default: TRUE. See also ci.thresholds.
conf_level	confidence level in the interval (0,1). Default is 0.95 producing 95% confidence intervals. Default: TRUE.
bootreps	numeric-valued number of bootstrap replication for estimation of 95% confidence intervals. Default: 4000.
print_auc	logical indicator for printing the area under the ROC curve (AUC) on the plot. Default: TRUE.

`partial_auc` One of `FALSE` or a length two numeric vector of the form `c(a, b)` where a and b are the endpoints of the interval over which to compute the out-of-sample partial AUC (pAUC). Ignored if `print_auc = FALSE`. Default: `c(0.8, 1)`.

`partial_auc_focus` one of `"sensitivity"` or `"specificity"`, indicating the measure for which the out-of-sample partial AUC is to be computed. Default: `"specificity"`.

`partial_auc_correct` logical indictor for transformation of the pAUC to fall within the range from 0.5 (random guess) to 1.0 (perfect classification). Default: `TRUE`.

`type` type of plot. See `plot`. Default: `"S"`.

Value

This function produces a plot as a side effect, and (optionally) returns a dataframe dataframe containing medians and bootstrap confidence limits of sensitivity and specificity.

References

Fawcett T. An introduction to ROC analysis. Pattern Recognition Letters. 2006. 27(8):861-874. https://doi.org/10.1016/j.patrec.2005.10.010

Linden A. Measuring diagnostic and predictive accuracy in disease management: an introduction to receiver operating characteristic (ROC) analysis. Journal of Evaluation in Clinical Practice. 2006; 12(2):132-139. https://onlinelibrary.wiley.com/doi/epdf/10.1111/j.1365-2753.2005.00598.x

Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez J-C, Muller M. pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics 2011; 12:77. https://www.biomedcentral.com/1471-2105/12/77

Examples

```
data(unicorns)
too_simple <- simple_screenr(testresult ~ Q1 + Q2 + Q3 + Q4 + Q5 + Q6 + Q7,
                             data = unicorns)
plot(too_simple)
```

`predict.easy_tool` *A Prediction Method for* `easy_tool`*-Class Objects*

Description

`predict.easy_tool` computes predicted simplified screening scores from new data.

Usage

```
## S3 method for class 'easy_tool'
predict(object = NULL, ..., newdata = NULL)
```

Arguments

object	an object of class easy_tool produced by `easy_tool`.
...	optional arguments to predict methods.
newdata	new dataframe from which predicted simplified screening scores are desired. The dataframe must contain values of the same response variables and covariates that were used to obtain object.

Value

predict.easy_tool returns (invisibly) a dataframe augmenting newdata with the predicted simplified test screening scores score.

Examples

```
new_corns <- data.frame(ID = c("Alice D.", "Bernie P."),
                        testresult = c(NA, NA), Q1 = c(0, 0), Q2 = c(0, 0),
                        Q3 = c(0, 0), Q4 = c(0, 0), Q5 = c(0, 1), Q6 = c(0, 1 ),
                        Q7 = c(0, 1))
attach(uniobj1)
et <- easy_tool(uniobj1, max = 3)
print(predict(et, newdata = new_corns))
```

---

predict.lasso_screenr *A Prediction Method for* lasso_screenr*-Class Objects*

Description

predict.lasso_screenr computes predicted probabilities of positive test results from new data.

Usage

```
## S3 method for class 'lasso_screenr'
predict(object = NULL, ..., newdata = NULL)
```

Arguments

object	an object of class lasso_screenr produced by `lasso_screenr`.
...	optional arguments to predict methods.
newdata	new dataframe from which predicted probabilities of positive test results are desired. The dataframe must contain values of the same response variables and covariates that were used to obtain obj.

Details

This method is a convenience wrapper for `glmpath::predict.glmpath`.

Value

predict.lasso_screenr returns (invisibly) a dataframe augmenting the complete cases in newdata with the predicted probabilities of positive test results phat_minAIC and phat_minBIC from the models that produced the minimum AIC and BIC, respectively.

Examples

```
attach(uniobj1)
## Get some new observations
new_corns <- data.frame(ID = c("Alice D.", "Bernie P."),
                        testresult = c(NA, NA),
                        Q1 = c(0, 0), Q2 = c(0, 0), Q3 = c(0, 1), Q4 = c(0, 0),
                        Q5 = c(0, 1), Q6 = c(0, 1), Q7 = c(0, 1))
## Predict the probabilities of testing positive for the new subjects
print(predict(uniobj1, newdata = new_corns))
```

---

predict.logreg_screenr

A Prediction Method for logreg_screenr*-Class Objects*

Description

predict.logreg_screenr computes predicted probabilities of positive test results from new data.

Usage

```
## S3 method for class 'logreg_screenr'
predict(object = NULL, ..., newdata = NULL)
```

Arguments

object	an object of class logreg_screenr produced by `logreg_screenr`.
...	optional arguments to predict methods.
newdata	new dataframe from which predicted probabilities of positive test results are desired. The dataframe must contain values of the same response variables and covariates that were used to obtain object.

Details

This method is a convenience wrapper for `stats::predict.glm`.

Value

predict.logreg_screenr returns (invisibly) a dataframe augmenting newdata with the predicted probabilities of positive test results phat.

Examples

```
attach(uniobj2)
## Get some new observations
new_corns <- data.frame(ID = c("Alice D.", "Bernie P."),
                        testresult = c(NA, NA), Q1 = c(0, 0), Q2 = c(0, 0),
                        Q3 = c(0, 0), Q4 = c(0, 0), Q5 = c(0, 1), Q6 = c(0, 1 ),
                        Q7 = c(0, 1))
## Predict the probabilities of testing positive for the new subjects
print(predict(uniobj2, newdata = new_corns))
```

---

print.easy_tool *A Print Method for* easy_tool*-Class Objects*

Description

print.easy_tool is a print method.

Usage

```
## S3 method for class 'easy_tool'
print(x, ...)
```

Arguments

x	an object of class easy_tool.
...	optional arguments to print methods.

See Also

get_what.easy_tool(from, what) for what = "ROCci".

Examples

```
attach(uniobj1)
print(uniobj1)
```

print.lasso_screenr *A Print Method for* lasso_screenr*-Class Objects*

Description

print.lasso_screenr is a print method for lasso_screenr-class objects.

Usage

```
## S3 method for class 'lasso_screenr'
print(x, ...)
```

Arguments

x	an object of class lasso_screenr
...	optional arguments to print methods.

See Also

get_what.lasso_screenr(from, what) for what = "ROCci".

Examples

```
attach(uniobj1)
print(uniobj1)
```

print.logreg_screenr *A Print Method for* logreg_screenr*-Class Objects*

Description

print.logreg_screenr is a print method for logreg_screenr-class objects.

Usage

```
## S3 method for class 'logreg_screenr'
print(x, ..., quote = FALSE)
```

Arguments

x	an object of class logreg_screenr.
...	optional arguments to print methods.
quote	logical indicator for whether or not strings should be printed.

Value

Nothing. Thresholds, specificities and sensitivities are printed as a side effect.

See Also

`get_what.logreg_screenr(from, what)` for `what = "ROCci"`.

Examples

```
attach(uniobj2)
print(uniobj2)
```

`print.simple_screenr` *A Print Method for* `simple_screenr`*-Class Objects*

Description

`print.simple_screenr` is print method for `simple_screenr` objects.

Usage

```
## S3 method for class 'simple_screenr'
print(x, ...)
```

Arguments

`x`	an object of class `simple_screenr`.
`...`	optional arguments to `print` methods.

Value

Nothing. Thresholds, specificities and sensitivities are printed as a side effect.

See Also

`get_what.simple_screenr(from, what)` for `what = "ROCci"`.

Examples

```
data(unicorns)
toosimple <- simple_screenr(testresult ~ Q1 + Q2 + Q3 + Q4 + Q5 + Q6,
                           data = unicorns)
print(toosimple)
```

rescale_to_int — *Rescale Positive Vectors or Matrices to Integers*

Description

`rescale_to_int` rescales the *non-zero* elements of real-valued numeric vectors or matrices to integers in the closed interval [1, `max`]. Any zero-valued elements are left unchanged.

Usage

```
rescale_to_int(x, max, colwise = TRUE)
```

Arguments

`x`	numeric matrix or vector of non-negative real numbers.
`max`	the value of largest element in the rescaled integer-valued vector.
`colwise`	(logical) rescale the matrix by column if `TRUE` (the default) or by row if `FALSE`.

Value

`rescale_to_int` returns a matrix of integers corresponding to `x` in which smallest *non-zero* element in each column/row is 1 and the largest element is `max`. Any elements having value zero are unchanged. If `x` is a vector then the result is an r x 1 matrix, where r is the number of elements in `x`. Otherwise the result is a r x c matrix where c is the number of columns in `x`.

See Also

`rescale`

Examples

```
x <- c(0.55, 1.21, 0.94, 0, 0.13)
rescale_to_int(x, max = 5)
```

roc_ci — *Compute Bootstrap Confidence Limits for Sensitivities and Specificities*

Description

`roc_ci` computes bootstrap confidence intervals from objects of class `roc`, as produced by the `pROC` package. `roc_ci` is simply a convenience wrapper for `pROC::ci.thresholds` re-formatted for `screenr`.

Usage

```
roc_ci(
  object,
  bootreps = 4000,
  conf_level = 0.95,
  progress = "none",
  thresholds = "local maximas",
  se_min = 0.8
)
```

Arguments

`object`	an object of class `roc`.
`bootreps`	number of bootstrap replicates. Default: 4000.
`conf_level`	confidence level for uncertainty intervals. Default: 0.95.
`progress`	character-valued type of progress display (see `help(pROC::ci.thresholds)`). Default `"none"`.
`thresholds`	type of thresholds (see `help(pROC::ci.thresholds)`).
`se_min`	minimum value of sensitivity returned. Default: 0.8.

Value

`roc_ci` returns a dataframe containing thresholds with their sensititives, specificities and uncertainy intervals.

See Also

`ci.thresholds`

`screenr` *screenr Package*

Description

The `screenr` package enables construction of binary test-screening tools. It is designed to enable those with only a basic familiarity with `R` to develop, validate and implement screening tools for diagnostic tests. `screenr` integrates the capabilities of the `glm`, `glmpath` and `pROC` packages for convenience and ease of use.

Consider the situation where a diagnostic test for some condition is relatively expensive, and the condition is rare. In that case, universal testing would not be efficient in terms of the yield of postive results per test performed. Now suppose that responses to a set of simple screening questions may be predictive of the condition. Package `screenr` enables estimation of thresholds for making decisions about when to test in order to screen in/out individuals based on Receiver Operating Characteristics (ROC) estimated from an initial sample. The choice of a particular screening threshold is left to the user, and should be based on careful consideration of application-specific tradeoffs between sensitivity and specificity. `screenr` also enables easy construction of screening tools.

A tutorial is available from `vignette("screenr_Tutorial", package = "screenr")`.

The pdf versions of the package manual and the tutorial are available at https://github.com/sgutreuter/screenr.

Details

The high-level functions in the `screenr` package are:

- `lasso_screenr` Selection of logistic models based on GLM path regularization
- `logreg_screenr` Test-screening based maximum-likelihood estimation of logistic models
- `easy_tool` Easy implementation of test-screening tools
- `simple_screenr` (To)simple un-optimized test-screening
- `rescale_to_int` Rescale a strictly positive vector of real numbers to integers
- `sens_spec_plus` Sensitivity, specificity and friends

`screenr` provides the usual `plot`, `print`, `summary`, `predict` methods for the objects produced by `lasso_screenr`, `logreg_screenr`, `simple_screenr` and `easy_tool`, and also `coef` methods for `lasso_screenr` and `logreg_screenr` objects. `screenr` also provides `get_what` methods to extract object components, and `ntpp` methods for computation of the average number of tests required to detect a single positive result and the residual positivity among those screened out of testing.

Note

The canonical source repository for `screenr` is https://github.com/sgutreuter/screenr

Author(s)

Steve Gutreuter: <sgutreuter@gmail.com>

`sens_spec_plus` *Compute Sensitivity, specificity and a few friends*

Description

`sens_spec_plus` computes sensitivity, specificity and a few friends from a gold standard and testing results. `sens_spec_plus` is a convenience wrapper for `epiR::epi.tests`.

Usage

```
sens_spec_plus(
  test = NULL,
  gold = NULL,
  data = NULL,
  method = c("exact", "wilson", "agresti", "clopper-pearson", "jeffreys"),
  conf_level = 0.95
)
```

Arguments

`test`	character-valued name of the variable containing testing results, coded as 0 for negative and 1 for positive.
`gold`	character-valued name of the variable containing gold standard, coded as 0 for negative and 1 for positive.
`data`	data frame containing `test` and `gold`.
`method`	type of confidence interval (`"exact"`, `"wilson"`, `"agresti"`, `"clopper-pearson"` or`"jeffreys"`). Default: `"exact"`.
`conf_level`	confidence level, a numeric value between 0 and 1. Default: 0.95.

Value

`sens_spec_plus` returns a list containing components `table` and `ests`:

`table` a 2 x 2 table which is the anti-transpose of the result produced by `base::table(gold, test)`.

`ests` a dataframe containing the apparent and true positive proportions, sensitivity, specificity, positive predictive value (PPV), negative predictive value (NPV), and the lower and upper confidence limits for each.

See Also

`epi.tests`

Examples

```
Gold <- rbinom(20, 1, 0.50)
Test <- Gold; Test[c(3, 5, 9, 12, 16)] <- 1 - Test[c(3, 5, 9, 12, 16)]
dat <- data.frame(Gold = Gold, Test = Test)
sens_spec_plus(test = "Test", gold = "Gold", data = dat)
```

`se_sp_max` *Return dataframe rows for which specificity is the maximum for each sensitivity*

Description

Given a dataframe containing multiple values of specificity for each value of sensitivity, return only the rows containing the largest specificity for each unique value of sensitivity.

Usage

```
se_sp_max(object)
```

Arguments

`object`	a dataframe containing at least columns named `sensitivities` and `specificities`

Value

se_sp_max returns a dataframe which is a subset of object containing only those rows for which specificity was the maximum for each unique value of sensitivity.

---

simple_screenr *An Overly Simple Approach to Test Screening*

Description

simple_screenr implements the method described in Bandason et al. (2016).

Usage

```
simple_screenr(
  formula,
  data,
  partial_auc = c(0.8, 1),
  partial_auc_focus = "sensitivity",
  partial_auc_correct = TRUE,
  conf_level = 0.95
)
```

Arguments

formula	an object of class formula defining the testing outcome and predictor covariates.
data	the "training" sample; a data frame containing the testing outcome and predictive covariates to be used for testing screening. The testing outcome must be binary (0,1) indicating negative and positive test results, respectively, or logical (TRUE/FALSE), and the screening scores are the row-wise sums of the values of those covariates. The covariates are typically binary (0 = no, 1 = yes) responses to questions, but the responses may also be ordinal numeric values.
partial_auc	either a logical FALSE or a numeric vector of the form c(left, right) where left and right are numbers in the interval [0, 1] specifying the endpoints for computation of the partial area under the ROC curve (pAUC). The total AUC is computed if partial_auc = FALSE. Default: c(0.8, 1.0)
partial_auc_focus	one of "sensitivity" or specificity, specifying for which the pAUC should be computed. partial.auc.focus is ignored if partial_auc = FALSE. Default: "sensitivity".
partial_auc_correct	logical value indicating whether the pAUC should be transformed the interval from 0.5 to 1.0. partial_auc_correct is ignored if partial_auc = FALSE. Default: TRUE).
conf_level	a number between 0 and 1 specifying the confidence level for confidence intervals for the (partial)AUC. Default: 0.95.

Details

simple_screenr computes the in-sample (*overly optimistic*) performances for development of a very simple test screening tool based on the sums of affirmative questionnaire responses. simpleScreener is not optimized and is intended only for comparision with lasso_screenr or logreg_screenr, either of which will almost certainly out-perform simple_screenr.

Value

simple_screenr returns (invisibly) an object of class simple_screenr containing the elements:

Call The function call.

Prevalence Prevalence of the test condition in the training sample.

ISroc An object of class roc containing the "in-sample" (overly-optimistic) receiver operating characteristics, and additional functions for use with this object are available in the pROC package.

Scores The training sample, including the scores.

References

Bandason T, McHugh G, Dauya E, Mungofa S, Munyati SM, Weiss HA, Mujuru H, Kranzer K, Ferrand RA. Validation of a screening tool to identify older children living with HIV in primary care facilities in high HIV prevalence settings. AIDS. 2016;30(5):779-785 http://dx.doi.org/10.1097/QAD.0000000000000959

Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez J-C, Muller M. pROC: An open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics. 2011;12(77):1-8. http://doi.org/10.1186/1471-2105-12-77

See Also

easy_tool for a better approach to simplification using the results from lasso_screenr or logreg_screenr.

lasso_screenr, logreg_screenr

Examples

```
data(unicorns)
toosimple <- simple_screenr(testresult ~ Q1 + Q2 + Q3 + Q4 + Q5 + Q6 + Q7,
                            data = unicorns)
summary(toosimple)
```

summary.easy_tool *A Summary Method for* easy_tool*-Class Objects*

Description

summary.easy_tool provides a summary method for easy-tool-class objects.

Usage

```
## S3 method for class 'easy_tool'
summary(object, ...)
```

Arguments

object	an easy_tool object.
...	optional arguments passed to summary methods.

Details

This is essentially a wrapper for glmpath::summary.glmpath provided for lasso_screenr objects.

Value

a dataframe containing the summary, including the Df, Deviance, AIC and BIC for each step along the GLM path for which the active set changed.

Examples

```
attach(uniobj1)
summary(uniobj1)
```

summary.lasso_screenr *A Summary Method for* lasso_screenr*-Class Objects*

Description

summary.lasso_screenr provides a summary method for lasso_screenr-class objects.

Usage

```
## S3 method for class 'lasso_screenr'
summary(object, ...)
```

Arguments

`object`	a lasso_screenr object
`...`	optional arguments passed to `summary` methods.

Details

This is essentially a wrapper for `glmpath::summary.glmpath` provided for `lasso_screenr` objects.

Value

a dataframe containing the summary, including the Df, Deviance, AIC and BIC for each step along the GLM path for which the active set changed.

Examples

```
attach(uniobj1)
summary(uniobj1)
```

`summary.logreg_screenr`

A Summary Method for `logreg_screenr`*-Class Objects*

Description

`summary.logreg_screenr` provides a summary method for `logreg_screenr`-class objects.

Usage

```
## S3 method for class 'logreg_screenr'
summary(object, ..., diagnostics = FALSE)
```

Arguments

`object`	an object of class `logreg_screenr` produced by function `logreg_screenr`.
`...`	optional arguments passed to `summary` methods.
`diagnostics`	a logical value; plot model diagnostics if `TRUE`.

Value

Nothing. Summaries are printed as a side effect.

Examples

```
attach(uniobj2)
summary(uniobj2)
```

summary.simple_screenr

A Summary Method for `simple_screenr`*-Class Objects*

Description

`summary.simple_screenr` provides a summary method for `simple_screenr`-class objects.

Usage

```
## S3 method for class 'simple_screenr'
summary(object, ...)
```

Arguments

`object`	an object of class `simple_screenr`.
`...`	optional arguments passed to `summary` methods.

Value

Nothing. Thresholds, specificities and sensitivities are printed as a side effect.

Examples

```
data(unicorns)
toosimple <- simple_screenr(testresult ~ Q1 + Q2 + Q3 + Q4 + Q5 + Q6 + Q7,
                            data = unicorns)
summary(toosimple)
```

unicorns

UIV Testing Training Data on Unicorns

Description

A preliminary study was conducted in which a random sample of 6,000 properly consented [unicorns](https://www.britannica.com/topic/unicorn) were recruited from 20 clinics. Each unicorn was asked seven questions about their behavior and health. Unicorns responded by stomping a hoof once to indicate "no", and twice to indicate "yes". A sample of venous blood was drawn from each, and was subsequently tested for the presence of antibodies to Unicorn Immunodeficiency Virus (UIV) using a standard assay algorithm.

Usage

```
data(unicorns)
```

Format

A data frame with eight columns:

`ID` Patient ID

`Q1` Response to screening question 1 (0 = "no", 1 = "yes")

`Q2` Response to screening question 2 (0 = "no", 1 = "yes")

`Q3` Response to screening question 3 (0 = "no", 1 = "yes")

`Q4` Response to screening question 4 (0 = "no", 1 = "yes")

`Q5` Response to screening question 5 (0 = "no", 1 = "yes")

`Q6` Response to screening question 6 (0 = "no", 1 = "yes")

`Q7` Response to screening question 7 (0 = "no", 1 = "yes")

`testresult` UIV status, where 0 and 1 denote negative and positive test results, repectively.

Note

In reality, the question responses and test results were generated using Bernoulli random-number generators.

Examples

```
## Not run:
head(unicorns)

## End(Not run)
```

---

uniobj1 *A* `lasso_screenr` *object*

Description

The result of `uniobj1 <- lasso_screenr(testresult ~ Q1 + Q2 + Q3 + Q4 + Q5 + Q6 + Q7, data = unicorns, Nfolds = 10, seed = 123)`

Usage

```
uniobj1
```

Format

An object of class `lasso_screenr`

Examples

```
## Not run:
summary(uniobj1)

## End(Not run)
```

uniobj2 — *A* logreg_screenr *object*

Description

The result of uniobj2 <- logreg_screenr(testresult ~ Q1 + Q2 + Q3 + Q4 + Q5 + Q6 + Q7, data = unicorns, link = "logit", Nfolds = 10, seed = 123)

Usage

```
uniobj2
```

Format

An object of class logreg_screenr

Examples

```
## Not run:
summary(uniobj2)

## End(Not run)
```

val_data — *UIV Test Validation Data on Unicorns*

Description

A follow-up study was conducted in which a random sample of 3,000 properly consented unicorns were recruited from 20 additional clinics. Each unicorn was asked six questions about their behavior and health. Unicorns responded by stomping a hoof once to indicate "no", and twice to indicate "yes". A sample of venous blood was drawn from each, and was subsequently tested for the presence of antibodies to Unicorn Immunodeficiency Virus (UIV) using a standard assay algorithm.

Usage

```
val_data
```

Format

A data frame with eight columns:

ID Patient ID

Q1 Response to screening question 1 (0 = "no", 1 = "yes")

Q2 Response to screening question 2 (0 = "no", 1 = "yes")

Q3 Response to screening question 3 (0 = "no", 1 = "yes")

`Q4` Response to screening question 4 (0 = "no", 1 = "yes")

`Q5` Response to screening question 5 (0 = "no", 1 = "yes")

`Q6` Response to screening question 6 (0 = "no", 1 = "yes")

`Q7` Response to screening question 7 (0 = "no", 1 = "yes")

`testresult` UIV status, where 0 and 1 denote negative and positive test results, repectively.

Examples

```
## Not run:
head(val_data)

## End(Not run)
```

Index