Package: gspcr (via r-universe)

September 1, 2024

Title Generalized Supervised Principal Component Regression

Version 0.9.5

Description Generalization of supervised principal component regression (SPCR; Bair et al., 2006, <doi:10.1198/016214505000000628>) to support continuous, binary, and discrete variables as outcomes and predictors (inspired by the 'superpc' R package <https://cran.r-project.org/package=superpc>).

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Suggests knitr, lmtest, patchwork, rmarkdown, superpc, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 2.10)

LazyData true

Imports dplyr, FactoMineR, ggplot2, MASS, MLmetrics, nnet, PCAmixdata, reshape2, rlang

VignetteBuilder knitr

Repository https://edoardocostantini.r-universe.dev

RemoteUrl https://github.com/edoardocostantini/gspcr

RemoteRef HEAD

RemoteSha d20e96135b858e8a7982c628cb03e0a10692e95c

Contents

`CFA_data` *CFA example data*

Description

Contains a data set used to develop and test the main features of the `gspcr` package. The data contains 50 predictors generated based on true number of principal components.

Format

`CFA_data` is a list containing two objects:

- `X`: A data.frame with 5000 rows (observations) and 30 columns (possible predictors.) This data was generated based on a CFA model describing 10 independent latent variables measured by 3 items each, and a factor loading matrix describing simple structure.
- `y`: A numeric vector of length 1000. This variable was genearted as a linear combination of 5 latent variables used to generate `X`.

Details

A supervised PCA approach should identify that only 5 components are useful for the prediction of `y` and that only the first 15 variables should be used to compute them.

Examples

```
# Check out the first 6 rows of the predictors
head(CFA_data$X)

# Check out first 6 elements of the dependent variable
head(CFA_data$y)
```

---

compute_sc *Compute the GLM systematic component.*

Description

Compute the systematic component of a GLM of interest.

Usage

```
compute_sc(mod, predictors)
```

Arguments

`mod`	a fit object returned by one of `stats::lm`, `stats::glm`, `nnet::multinom`, or `MASS::polr`.
`predictors`	matrix or data.frame of predictor values to compute the systematic component based on.

Details

This function takes different model objects and knows how to treat the coefficient vector (or matrix) to obtain the systematic component.

Value

a matrix of $n \times k$, where k is equal to 1 for all but multi-categorical models. This matrix contains the systematic component values for the provided predictors.

Author(s)

Edoardo Costantini, 2023

cp_AIC *Compute Akaike's information criterion*

Description

Computes Akaike's information criterion for comparing competing models.

Usage

```
cp_AIC(ll, k)
```

Arguments

`ll`	numeric vector of length 1 (or an object of class 'logLik') storing the log-likelihood of the model of interest
`k`	numeric vector of length 1 storing the number of parameters estimated by the model

Value

numeric vector of length 1 storing the computed AIC.

Author(s)

Edoardo Costantini, 2023

Examples

```
# Fit some model
lm_out <- lm(mpg ~ cyl + disp, data = mtcars)

# Compute AIC with your function
AIC_M <- cp_AIC(
    ll = logLik(lm_out),
    k = length(coef(lm_out)) + 1 # intercept + reg coefs + error variance
)
```

cp_BIC *Compute bayesian information criterion*

Description

Computes bayesian information criterion for comparing competing models.

Usage

```
cp_BIC(ll, n, k)
```

Arguments

ll numeric vector of length 1 (or an object of class 'logLik') storing the log-likelihood of the model of interest

n numeric vector of length 1 storing the sample size of data used to compute the log-likelihood

k numeric vector of length 1 storing the number of estimated parameters by the model

Value

numeric vector of length 1 storing the computed BIC.

Author(s)

Edoardo Costantini, 2023

Examples

```
# Fit some model
lm_out <- lm(mpg ~ cyl + disp, data = mtcars)

# Compute BIC with your function
BIC_M <- cp_BIC(
    ll = logLik(lm_out),
    n = nobs(lm_out),
    k = length(coef(lm_out)) + 1 # intercept + reg coefs + error variance
)
```

cp_F *Compute F statistic*

Description

Computes the F statistic comparing two nested models.

Usage

```
cp_F(y, y_hat_restricted, y_hat_full, n = length(y), p_restricted = 0, p_full)
```

Arguments

y numeric vector storing the observed values on the dependent variable

y_hat_restricted
numeric vector storing the predicted values on y based on the restricted model

y_hat_full numeric vector storing the predicted values on y based on the full model

n numeric vector of length 1 storing the sample size used to train the models `p_restricted` numeric vector of length 1 storing the number of predictors involved in training the restricted model

`p_full` numeric vector of length 1 storing the number of predictors involved in training the full model

Details

Note that:

- The full model is always the model with more estimated parameters, the model with more predictor variables.
- The restricted model is the model with fewer estimated parameters.
- The restricted model must be nested within the full model.

Value

numeric vector of length 1 storing the F-statistic

Author(s)

Edoardo Costantini, 2023

Examples

```
# Null vs full model
lm_n <- lm(mpg ~ 1, data = mtcars) # Fit a null model
lm_f <- lm(mpg ~ cyl + disp, data = mtcars) # Fit a full model
f_M <- cp_F(
    y = mtcars$mpg,
    y_hat_restricted = predict(lm_n),
    y_hat_full = predict(lm_f),
    p_full = 2
)

# Simpler vs more complex model
lm_f_2 <- lm(mpg ~ cyl + disp + hp + drat + qsec, data = mtcars) # a more complex full model
f_change_M <- cp_F(
    y = mtcars$mpg,
    y_hat_restricted = predict(lm_f),
    y_hat_full = predict(lm_f_2),
    p_restricted = 2,
    p_full = 5
)
```

cp_gR2 *Compute generalized R-squared*

Description

Computes the Cox and Snell generalized R-squared.

Usage

```
cp_gR2(ll_n, ll_f, n)
```

Arguments

ll_n	numeric vector of length 1 (or an object of class 'logLik') storing the log-likelihood of the null (restricted) model
ll_f	numeric vector of length 1 (or an object of class 'logLik') storing the log-likelihood of the full model
n	numeric vector of length 1 storing the sample size of the data used to estimate the models

Details

The Cox and Snell generalized R-squared is equal to the R-squared when applied to multiple linear regression. The highest value for this measure is 1 - exp(ll_n)^(2/n), which is usually < 1. The null (restricted) model must be nested within the full model.

Value

numeric vector of length 1 storing the computed Cox and Snell generalized R-squared.

Author(s)

Edoardo Costantini, 2023

References

Allison, P. D. (2014, March). Measures of fit for logistic regression. In Proceedings of the SAS global forum 2014 conference (pp. 1-13). Cary, NC: SAS Institute Inc.

Examples

```
# Fit a null model
lm_n <- lm(mpg ~ 1, data = mtcars)

# Fit a full model
lm_f <- lm(mpg ~ cyl + disp, data = mtcars)

# Compute generalized R2
```

```
gr2 <- cp_gR2(
    ll_n = as.numeric(logLik(lm_n)),
    ll_f = as.numeric(logLik(lm_f)),
    n = nobs(lm_f)
)
```

cp_LRT *Compute likelihood ratio test*

Description

Computes the likelihood ratio expressed as a difference between the log-likelihoods of observed data under two nested competing models.

Usage

```
cp_LRT(ll_restricted, ll_full)
```

Arguments

`ll_restricted`	numeric vector of length 1 (or an object of class 'logLik') storing the log-likelihood of the observed data under the restricted model
`ll_full`	numeric vector of length 1 (or an object of class 'logLik') storing the log-likelihood of the observed data under the full model

Details

Note that:

- The full model is always the model with more estimated parameters, the model with more predictor variables.
- The restricted model is the model with fewer estimated parameters.
- The restricted model must be nested within the full model.

Value

numeric vector of length 1 storing the likelihood ratio test statistic

Author(s)

Edoardo Costantini, 2023

Examples

```
# Fit a nested model
nested <- glm(mpg ~ cyl + disp, data = mtcars)

# Fit a complex model
complex <- glm(mpg ~ cyl + disp + hp + am, data = mtcars)

# Compute log-likelihood statistic with your function
LRT_M <- cp_LRT(
    ll_restricted = logLik(nested),
    ll_full = logLik(complex)
)
```

`cp_thrs_LLS` *Compute threshold values based on Log-likelihood values*

Description

Produces a vector of threshold values that define active predictors.

Usage

```
cp_thrs_LLS(dv, ivs, fam)
```

Arguments

`dv` numeric vector or factor of dependent variable values

`ivs` $n \times p$ data.frame of independent variables (factors allowed)

`fam` character vector of length 1 storing the description of the error distribution and link function to be used in the model (see `cv_gspcr()` for the list of possible options)

Value

numeric vector of log-likelihood value from all of the univariate GLM models regressing `dv` on each column of `ivs`.

Author(s)

Edoardo Costantini, 2023

Examples

```
# Example inputs
dv <- mtcars[, 1]
ivs <- mtcars[, -1]
fam <- "gaussian"

# Use function
cp_thrs_LLS(dv, ivs, fam)
```

cp_thrs_NOR *Compute normalized association measure*

Description

A function to compute the normalized bivariate association measures between a dv and a collection of ivs.

Usage

```
cp_thrs_NOR(dv, ivs, s0_perc = NULL, scale_dv = TRUE, scale_ivs = TRUE)
```

Arguments

dv numeric vector of dependent variable values

ivs $n \times p$ matrix of numeric independent variables

s0_perc numeric vector of length 1 storing the factor for the denominator of association statistic (i.e., the percentile of standard deviation values added to the denominator, a value between 0 and 1.) The default is 0.5 (the median)

scale_dv logical value defining whether dv should be scaled

scale_ivs logical value defining whether ivs should be scaled

Details

This function is based on the function cor.func in the package superpc.

Value

numeric vector of bivariate association measures between dv and ivs. numeric vector of log-likelihood value from all of the univariate GLM models regressing dv on each column of ivs.

Author(s)

Edoardo Costantini, 2023

References

Bair E, Hastie T, Paul D, Tibshirani R (2006). “Prediction by supervised principal components.” J. Am. Stat. Assoc., 101(473), 119-137.

Examples

```
# Example inputs
dv <- mtcars[, 1]
ivs <- mtcars[, -1]
s0_perc <- 0

# Use the function
cp_thrs_NOR(dv, ivs, s0_perc)
```

cp_thrs_PR2 *Compute threshold values based on the pseudo R2*

Description

Produces a vector of threshold values that define active predictors.

Usage

```
cp_thrs_PR2(dv, ivs, fam)
```

Arguments

`dv`	numeric vector or factor of dependent variable values
`ivs`	$n \times p$ data.frame of independent variables (factors allowed)
`fam`	character vector of length 1 storing the description of the error distribution and link function to be used in the model (see `cv_gspcr()` for the list of possible options)

Value

A vector of bivariate association measures between `dv` and `ivs`.

Author(s)

Edoardo Costantini, 2023

Examples

```
# Example inputs
dv <- mtcars[, 1]
ivs <- mtcars[, -1]

# Use the function
cp_thrs_PR2(dv, ivs, fam = "gaussian")
```

---

cp_validation_fit *Compute fit measure(s) on the validation data set*

Description

Given a training and validation data set, it computes a target fit measure on the validation data set.

Usage

```
cp_validation_fit(y_train, y_valid, X_train, X_valid, fam, fit_measure)
```

Arguments

`y_train`	numeric vector or factor of dependent variable values from the training set
`y_valid`	numeric vector or factor of dependent variable values from the validation set
`X_train`	$n \times p$ data.frame of independent variables (factors allowed) from the training set. Can also be set to NULL to obtain the log-likelihood of the new data under the null model.
`X_valid`	$n \times p$ data.frame of independent variables (factors allowed) from the validation set. If `X_train` is set to NULL to obtain the log-likelihood of the new data under the null model, `X_valid` is ignored.
`fam`	character vector of length 1 storing the description of the error distribution and link function to be used in the model (see `cv_gspcr()` for the list of possible options)
`fit_measure`	character vector indicating which fit measure should be computed (see `cv_gspcr()` for the list of possible options)

Details

The validation data set can be specified to be the same as the training data set if desired.

Value

numeric vector of length 1 storing the requested fit measure

Author(s)

Edoardo Costantini, 2023

Examples

```
# Example inputs
y_train = mtcars[1:20, 1]
y_valid = mtcars[-c(1:20), 1]
X_train = mtcars[1:20, -1]
X_valid = mtcars[-c(1:20), -1]
fam = "gaussian"
fit_measure = "BIC"

# Use the function
cp_validation_fit(y_train, y_valid, X_train, X_valid, fam, fit_measure)
```

cv_average *Average fit measures computed in the K-fold cross-validation procedure*

Description

Function to average results from an array of K-fold CV fit measures.

Usage

```
cv_average(cv_array, fit_measure)
```

Arguments

cv_array	$Q \times nthrs \times K$ array containing fit measures computed for different combinations of the number of components, threshold values, and number of CV-folds.
fit_measure	character vector of length 1 indicating the type of fit measure to be used in the to cross-validation procedure

Details

The input of this function is an array of $Q \times nthrs \times K$, where Q is the number of principal components, nthrs is the number of thresholds, and K is the number of folds.

Value

list of three $Q \times nthrs$ matrices:

- scor: contains the average CV scores across the K folds
- scor_upr: contains the average CV scores across the K folds + 1 standard deviation
- scor_lwr: contains the average CV scores across the K folds - 1 standard deviation

Author(s)

Edoardo Costantini, 2023

Examples

```
# Example inputs
cv_array = array(abs(rnorm(10 * 3 * 2)), dim = c(10, 3, 2))
fit_measure = "F"

# Use the function
cv_average(cv_array, fit_measure)
```

cv_choose *Cross-validation choice*

Description

Extracting the CV choices of SPCR parameters.

Usage

```
cv_choose(scor, scor_lwr, scor_upr, K, fit_measure)
```

Arguments

scor	$npcs \times nthrs$ matrix of K-fold CV scores
scor_lwr	$npcs \times nthrs$ matrix of score lower bounds
scor_upr	$npcs \times nthrs$ matrix of score upper bounds
K	numeric vector of length 1 storing the number of folds for the K-fold cross-validation procedure
fit_measure	character vector of length 1 indicating the type of fit measure to be used in the to cross-validation procedure

Details

Given a matrix of $npcs \times nthrs$, returns the best choice based on the type of fit measure (best overall and 1se rule versions.) This function returns as solutions:

- `default`: the best choice based on the given fit measure (e.g. highest likelihood ratio test statistic, lowest BIC)
- `oneSE`: the solution that defined the most parsimonious model within 1 standard error from the best one. When both the number of components and the threshold parameter are cross-validated, the 1-standard error rule finds the candidate alternative solutions using the lowest number of PCs and having the best fit-measure. This decision is guided by the desire to counterbalance the tendency of GSPCR of selecting the highest number of components available when using cross-validation.

Value

A list of two numeric vectors:

- `default`: numeric vector of length 2 that reports the coordinates in `scor` defining the default solution.
- `oneSE`: numeric vector of length 2 that reports the coordinates for `scor` defining the solution based on the one standard error rule

Author(s)

Edoardo Costantini, 2023

Examples

```
# Score matrices
scor <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 3, ncol = 2)
scor_lwr <- matrix(c(1, 2, 3, 4, 5, 6) - 1.5, nrow = 3, ncol = 2)
scor_upr <- matrix(c(1, 2, 3, 4, 5, 6) + 1.5, nrow = 3, ncol = 2)

# Number of folds
K <- 10

# Type of fit_measure
fit_measure <- "F"

# Use the function
cv_choose(scor, scor_lwr, scor_upr, K, fit_measure)
```

`cv_gspcr` *Cross-validation of Generalized Principal Component Regression*

Description

Use K-fold cross-validation to decide on the number of principal components and the threshold value for GSPCR.

Usage

```
cv_gspcr(
  dv,
  ivs,
  fam = c("gaussian", "binomial", "poisson", "baseline", "cumulative")[1],
  thrs = c("LLS", "PR2", "normalized")[1],
  nthrs = 10L,
  npcs_range = 1L:3L,
  K = 5,
  fit_measure = c("F", "LRT", "AIC", "BIC", "PR2", "MSE")[1],
  max_features = ncol(ivs),
  min_features = 1,
  oneSE = TRUE,
  save_call = TRUE
)
```

Arguments

dv numeric vector or factor of dependent variable values

ivs $n \times p$ data.frame of independent variables (factors allowed)

fam character vector of length 1 storing the description of the error distribution and link function to be used in the model

thrs character vector of length 1 storing the type of threshold to be used (see below for available options)

nthrs numeric vector of length 1 storing the number of threshold values to be used

npcs_range numeric vector defining the numbers of principal components to be used

K numeric vector of length 1 storing the number of folds for the K-fold cross-validation procedure

fit_measure character vector of length 1 indicating the type of fit measure to be used in the cross-validation procedure

max_features numeric vector of length 1 indicating the maximum number of features that can be selected

min_features numeric vector of length 1 indicating the minimum number of features that should be selected

oneSE logical value indicating whether the results with the 1se rule should be saved

save_call logical value indicating whether the call should be saved and returned in the results

Details

The variables in ivs do not need to be standardized beforehand as the function handles scaling appropriately based on the measurement levels of the data.

The fam argument is used to define which model will be used when regressing the dependent variable on the principal components:

- gaussian: fits a linear regression model (continuous dv)
- binomial: fits a logistic regression model (binary dv)
- poisson: fits a poisson regression model (count dv)
- baseline: fits a baseline-category logit model (nominal dv, using nnet::multinom())
- cumulative: fits a proportional odds logistic regression (ordinal dv, using MASS::polr())

The thrs argument defines the bivariate association-threshold measures used to determine the active set of predictors for a SPCR analysis. The following association measures are supported (measurement levels allowed reported between brackets):

- LLS: simple GLM regression likelihoods (any dv with any iv)
- PR2: Cox and Snell generalized R-squared is computed for the GLMs between dv and every column in ivs. Then, the square root of these values is used to obtain the threshold values. For more information about the computation of the Cox and Snell R2 see the help file for cp_gR2(). When using this measure for simple linear regressions (with continuous dv and ivs) is equivalent to the regular R-squared. Therefore, it can be thought of as equivalent to the bivariate correlations between dv and ivs. (any dv with any iv)
- normalized: normalized correlation based on superpc::superpc.cv() (continuous dv with continuous ivs)

The fit_measure argument defines which fit measure should be used within the cross-validation procedure. The supported measures are:

- F: F-statistic computed with cp_F() (continuous dv)
- LRT: likelihood-ratio test statistic computed with cp_LRT() (any dv)
- AIC: Akaike's information criterion computed with cp_AIC() (any dv)
- BIC: bayesian information criterion computed with cp_BIC() (any dv)
- PR2: Cox and Snell generalized R-squared computed with cp_gR2() (any dv)
- MSE: Mean squared error compute with MLmetrics::MSE() (continuous dv)

Details regarding the 1 standard error rule implemented here can be found in the documentation for the function cv_choose().

Value

Object of class gspcr, which is a list containing:

- solution: a list containing the number of PCs that was selected (Q), the threshold value used, and the resulting active set for both the standard and oneSE solutions
- sol_table: data.frame reporting the threshold number, value, and the number of PCs identified by the procedure
- thr: vector of threshold values of the requested type used for the K-fold cross-validation procedure
- thr_cv: numeric vector of length 1 indicating the threshold number that was selected by the K-fold cross-validation procedure using the default decision rule
- thr_cv_1se: numeric vector of length 1 indicating the threshold number that was selected by the K-fold cross-validation procedure using the 1-standard-error rule
- Q_cv: numeric vector of length 1 indicating the number of PCs that was selected by the K-fold cross-validation procedure using the default decision rule
- Q_cv_1se: numeric vector of length 1 indicating the number of PCs that was selected by the K-fold cross-validation procedure using the 1-standard-error rule
- scor: $npcs \times nthrs$ matrix of fit-measure scores averaged across the K folds
- scor_lwr: $npcs \times nthrs$ matrix of fit-measure score lower bounds averaged across the K folds
- scor_upr: $npcs \times nthrs$ matrix of fit-measure score upper bounds averaged across the K folds

- `pred_map`: matrix of $p \times nthrs$ logical values indicating which predictors were active for every threshold value used
- `gspcr_call`: the function call

Author(s)

Edoardo Costantini, 2023

References

Bair, E., Hastie, T., Paul, D., & Tibshirani, R. (2006). Prediction by supervised principal components. Journal of the American Statistical Association, 101(473), 119-137.

Examples

```
# Example input values
dv <- mtcars[, 1]
ivs <- mtcars[, -1]
thrs <- "PR2"
nthrs <- 5
fam <- "gaussian"
npcs_range <- 1:3
K <- 3
fit_measure <- "F"
max_features <- ncol(ivs)
min_features <- 1
oneSE <- TRUE
save_call <- TRUE

# Example usage
out_cont <- cv_gspcr(
  dv = GSPCRexdata$y$cont,
  ivs = GSPCRexdata$X$cont,
  fam = "gaussian",
  nthrs = 5,
  npcs_range = 1:3,
  K = 3,
  fit_measure = "F",
  thrs = "normalized",
  min_features = 1,
  max_features = ncol(GSPCRexdata$X$cont),
  oneSE = TRUE,
  save_call = TRUE
)
```

`est_gspcr` *Estimate Generalized Principal Component Regression*

Description

Estimate SPCA on the data given chosen parameter values

Usage

```
est_gspcr(object = NULL, dv, ivs, fam, active_set, ndim)
```

Arguments

`object` `gspcrcv` object resulting from the call of `cv_gspcr()`. If this is specified, then every other argument can be left blank.

`dv` numeric vector or factor of dependent variable values

`ivs` $n \times p$ data.frame of independent variables (factors allowed)

`fam` character vector of length 1 storing the description of the error distribution and link function to be used in the model

`active_set` names of the columns of ivs to be used as predictors

`ndim` numeric vector defining the number of principal components to be used (2 or more)

Details

After deciding on the number of components and the active set, this estimates the GSPCR model. This function can be used by specifying the object argument or by filling in custom values for every argument. If both the object and any other argument are specified, then the argument values will be prioritized.

Value

Description of function output

Author(s)

Edoardo Costantini, 2023

References

Bair, E., Hastie, T., Paul, D., & Tibshirani, R. (2006). Prediction by supervised principal components. Journal of the American Statistical Association, 101(473), 119-137.

est_univ_mods *Estimate simple GLM models*

Description

Given a dependent variable, a set of possible predictors, and a GLM family, this function estimates a null GLM and all of the simple GLMs.

Usage

```
est_univ_mods(dv, ivs, fam)
```

Arguments

dv	numeric vector or factor of dependent variable values
ivs	$n \times p$ data.frame of independent variables (factors allowed)
fam	character vector of length 1 storing the description of the error distribution and link function to be used in the model (see `cv_gspcr()` for the list of possible options)

Details

We use the expression "simple GLM models" to describe GLM models with a single dependent variable and a single predictor.

Value

List containing:

- `ll0`: log-likelihoods for the null model
- `lls`: log-likelihoods for all the simple models
- `coefs`: if `dv` and `ivs` are continuous, standardized simple regression coefficients

Author(s)

Edoardo Costantini, 2023

Examples

```
# Run the function on the example data set
dv_con_ivs_con <- est_univ_mods(
    dv = GSPCRexdata$y$cont,
    ivs = GSPCRexdata$X$cont,
    fam = "gaussian"
)
```

GSPCRexdata *GSPCR example data*

Description

Contains a data set used to develop and test the main features of the gspcr package. The data contains a dependent variable and 50 predictors generated based on true number of principal components.

Format

GSPCRexdata is a list containing two data.frame objects:

- X: A list of data.frames with 1000 rows (observations) and 50 columns (possible predictors). The list contains matrices storing data coded with different measurement levels:
  - cont with 50 continuous variables
  - bin with 50 binary variables (factors)
  - ord with 50 ordinal variables (ordered factors)
  - cat with 50 categorical variables (unordered factors)
  - mix with 20 continuous variables, 10 binary variables (factors), 10 ordinal variables (ordered factors), 10 categorical variables (unordered factors).
- y: A data.frame with 1000 rows and 5 columns. The first column cont is a continuous variable produced using a linear model with the first two PCs underlying X as a data-generating model. The other columns are transformed versions of cont to match common discrete target distribution in the social sciences. These are the variables stored:
  - cont continuous dependent variable (numeric vector)
  - bin binary dependent variable (factor)
  - ord ordinal dependent variable (ordered factor)
  - cat nominal dependent variable (unordered factor)
  - pois count dependent variable (numeric vector)

Examples

```
# Check out the first 6 rows of the continuous predictors
head(GSPCRexdata$X$cont)

# Check out first 6 rows of the dv data.frame
head(GSPCRexdata$y)
```

`LL_baseline` *Baseline category logistic regression log-likelihood*

Description

Computes the baseline category logistic regression log-likelihood given a nominal categorical variable and the corresponding GLM linear predictor values.

Usage

```
LL_baseline(y, x, mod)
```

Arguments

`y`	factor or disjunctive table representation recording a nominal variable with 3 or more categories.
`x`	data.frame (or matrix) containing predictor values.
`mod`	`multinom` object containing the estimated baseline-category logit model.

Details

If `x` and `y` are equal to the data on which `mod` has been trained, this function returns the same result as the default `logLink` function. If `x` and `y` are new, the function returns the log-likelihood of the new data under the trained model.

A disjunctive table is a matrix representation of a multi-categorical variable. The dimensionality of the matrix is i times j, with i = number of observations, and j = number of categories. `y_{ij}` is equal to 1 if observation i responded with category j, and it is equal to 0 otherwise. The log-likelihood equation is based on Agresti (2002, p. 192).

Value

A list containing:

- `ll` atomic vector of length 1 containing the log-likelihood value.
- `sc` numeric matrix containing the systematic component for the input `x` and `mod`.

Author(s)

Edoardo Costantini, 2023

References

Agresti, A. (2012). Categorical data analysis (Vol. 792). John Wiley & Sons.

LL_binomial *Binomial log-likelihood*

Description

Computes the binomial log-likelihood given a response vector and corresponding GLM linear predictor values.

Usage

```
LL_binomial(y, x, mod)
```

Arguments

y	numeric vector (or factor) recording a binary dependent variable.
x	data.frame (or matrix) containing predictor values.
mod	glm object containing and estimated logistic regression model.

Details

If x and y are equal to the data on which mod has been trained, this function returns the same result as the default logLink function. If x and y are new, the function returns the log-likelihood of the new data under the trained model. The log-likelihood equation is based on Agresti (2002, p. 192).

Value

A list containing:

- ll an atomic vector of length 1 containing the log-likelihood value.
- sc an atomic vector containing the systematic component for the input x and mod.

Author(s)

Edoardo Costantini, 2022

References

Agresti, A. (2012). Categorical data analysis (Vol. 792). John Wiley & Sons.

LL_cumulative *Proportional odds model log-likelihood*

Description

Computes the log-likelihood given an ordered category response vector and corresponding GLM linear predictor values.

Usage

```
LL_cumulative(y, x, mod)
```

Arguments

y	ordered factor or disjunctive table representation recording an ordinal variable with 3 or more categories.
x	data.frame (or matrix) containing predictor values.
mod	polr object containing the estimated proportional odds model.

Details

If x and y are equal to the data on which mod has been trained, this function returns the same result as the default logLink function. If x and y are new, the function returns the log-likelihood of the new data under the trained model. The log-likelihood equation is based on Agresti (2002, p. 192).

Value

A list containing:

- ll an atomic vector of length 1 containing the log-likelihood value.
- sc, a numeric matrix containing the systematic component for the input x and mod.

Author(s)

Edoardo Costantini, 2022

References

Agresti, A. (2012). Categorical data analysis (Vol. 792). John Wiley & Sons.

LL_gaussian *Gaussian log-likelihood*

Description

Computes the gaussian (normal) log-likelihood of a vector of observed values given a trained linear regression model.

Usage

```
LL_gaussian(y, x, mod)
```

Arguments

y	numeric vector recording a continuous dependent variable.
x	data.frame (or matrix) containing predictor values.
mod	glm or lm object containing the estimated linear regression model.

Details

If x and y are equal to the data on which mod has been trained, this function returns the same result as the default logLink function. If x and y are new, the function returns the log-likelihood of the new data under the trained model.

Value

A list containing:

- ll an atomic vector of length 1 containing the log-likelihood value.
- sc an atomic vector containing the systematic component for the input x and mod.

Author(s)

Edoardo Costantini, 2022

LL_newdata *Log-Likelihood for new data*

Description

Given training and validation datasets, this function returns the log-likelihood of unobserved data under the model trained on the training data.

Usage

```
LL_newdata(y_train, y_valid, X_train = NULL, X_valid = NULL, fam)
```

Arguments

y_train Vector of DV values in the training dataset.

y_valid Vector of DV values in the validation dataset.

X_train Matrix of IV values in the training dataset. Can also be set to 1 to obtain the log-likelihood of the new data under the null model.

X_valid Matrix of IV values in the validation dataset. If X_train is set to 1 to obtain the log-likelihood of the new data under the null model, X_valid is ignored.

fam character vector of length 1 storing the description of the error distribution and link function to be used in the model (see cv_gspcr() for the list of possible options)

Details

This function trains a GLM regressing y_train on X_train using as link function what is specified in fam. Then, it computes the predictions for the validation data based on the trained model on the scale of the linear predictors (e.g., logit). The likelihood of the validation under the model is returned.

Value

A list of objects.

Author(s)

Edoardo Costantini, 2023

LL_poisson *Poisson regression log-likelihood*

Description

Computes the Poisson regression log-likelihood of a vector of observed values given the GLM systematic component.

Usage

```
LL_poisson(y, x, mod)
```

Arguments

y numeric vector recording a count dependent variable.

x data.frame (or matrix) containing predictor values.

mod glm object containing the estimated poisson regression model.

Details

If `x` and `y` are equal to the data on which `mod` has been trained, this function returns the same result as the default `logLink` function. If `x` and `y` are new, the function returns the log-likelihood of the new data under the trained model.

Value

A list containing:

- `ll` an atomic vector of length 1 containing the log-likelihood value.
- `sc` an atomic vector containing the systematic component for the input `x` and `mod`.

Author(s)

Edoardo Costantini, 2023

References

Agresti, A. (2012). Categorical data analysis (Vol. 792). John Wiley & Sons.

`pca_mix` *PCA of a mixture of numerical and categorical data*

Description

Wrapper for the `PCAmixdata::PCAmix()` function to be used in the main cross-validation procedure.

Usage

```
pca_mix(X_tr, X_va, npcs = 1)
```

Arguments

`X_tr`	data.frame of training data
`X_va`	data.frame of validation data
`npcs`	number of principal components to keep

Value

a list of training and validation PC scores

Author(s)

Edoardo Costantini, 2023

References

Chavent M, Kuentz V, Labenne A, Liquet B, Saracco J (2017). PCAmixdata: Multivariate Analysis of Mixed Data. R package version 3.1, https://CRAN.R-project.org/package=PCAmixdata.

Examples

```
# Example inputs
data(wine, package = "FactoMineR")
X <- wine[, c(1, 2, 16, 22)]
X$Label <- factor(X$Label)
X$Soil <- factor(X$Soil)
X_tr <- X[1:15, ]
X_va <- X[16:21, ]
npcs <- 2

# Example use
pca_mix(
    X_tr = X[1:15, ],
    X_va = X[16:21, ],
    npcs = 2
)
```

plot.gspcrcv *Plot the cross-validation solution path for the GSPCR algorithm*

Description

Produces a scatter plot showing the CV score obtained by cv_gspcr (Y-axis) with different threshold values (X-axis) for a different number of components (lines).

Usage

```
## S3 method for class 'gspcrcv'
plot(
  x,
  y = NULL,
  labels = TRUE,
  errorBars = FALSE,
  discretize = TRUE,
  y_reverse = FALSE,
  print = TRUE,
  ...
)
```

Arguments

`x` An object of class `gspcr`.

`y` The CV fit measure to report on the Y axis. Default is the fit measure specified in `cv_gspcr()`.

`labels` Logical value. `FALSE` hides the labels of the points indicating the number of components used. The default is `TRUE`.

`errorBars` Logical value. `TRUE` shows the error bars for each point. The default is `FALSE`.

`discretize` Logical value. `TRUE` treats the X-axis as a discrete measure that facilitates comparing solution paths between different fit measures. The default is `TRUE`.

`y_reverse` Logical value. `TRUE` reverses the `y` axis scale. The default is `FALSE`.

`print` Logical value. TRUE prints the plot when the function is called. The default is `TRUE`.

`...` Other arguments passed on to methods. Not currently used.

Details

The bounds defining the error bars are computed by `cv_gspcr()`. First, the K-fold cross-validation score of the statistic of interest (e.g., the F score, the MSE) is computed. Then, the standard deviation of the statistic across the K folds is computed. Finally, the bounds used for the error bars are computed by summing and subtracting this standard deviation to and from the K-fold cross-validation score of the statistic.

Reversing the `y`-axis with `y_reverse` can be helpful to compare results obtained by different fit measures.

Value

A scatter plot of `ggplot` class

Author(s)

Edoardo Costantini, 2023

`predict.gspcrout` *Predict GSPCR model dependent variable scores*

Description

Predicts dependent variable values based on (new) predictor variables values.

Usage

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

Arguments

`object`	An object of class `gspcr`.
`newdata`	optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
`...`	further arguments passed to or from other methods.

Value

Vector of prediction in "response" format for numerical data and probability of class membership for categorical data

Author(s)

Edoardo Costantini, 2023

Index