Package ‘easy.glmnet’

September 11, 2024

Type Package

Title Functions to Simplify the Use of 'glmnet' for Machine Learning

Version 1.0

Date 2024-09-06

Description Provides several functions to simplify using the 'glmnet' package: converting data frames into matrices ready for 'glmnet'; b) imputing missing variables multiple times; c) fitting and applying prediction models straightforwardly; d) assigning observations to folds in a balanced way; e) cross-validate the models; f) selecting the most representative model across imputations and folds; and g) getting the relevance of the model regressors; as described in several publications: Solanes et al. (2022) <doi:10.1038/s41537-022-00309-w>, Palau et al. (2023) <doi:10.1016/j.rpsm.2023.01.001>, Sobregrau et al. (2024) <doi:10.1016/j.jpsychores.2024.111656>.

License GPL-3

Imports doParallel, foreach, glmnet, parallel, survival

Suggests pROC

NeedsCompilation no

Author Joaquim Radua [aut, cre] (<`https://orcid.org/0000-0003-1240-5438`>)

Maintainer Joaquim Radua <`quimradua@gmail.com`>

Repository CRAN

Date/Publication 2024-09-11 12:50:02 UTC

Contents

`assign.folds` *Assign observations to folds in a balanced way*

Description

Function to assign observations to folds, ensuring a similar distribution across folds (and sites).

Usage

```
assign.folds(y, family = c("binomial", "cox", "gaussian"), site = NULL, nfolds = 10)
```

Arguments

`y`	response to be predicted. A binary vector for `"binomial"`, an object of class `"Surv"` for `"cox"`, or a numeric vector for `"gaussian"`.
`family`	distribution of y: `"binomial"`, `"cox"`, or `"gaussian"`.
`site`	vector with the sites' names, or NULL for studies conducted in a single site.
`nfolds`	number of folds.

Details

If `family` is `"binomial"`, the function randomly assigns the folds separately for the two outcomes. If `family` is `"gaussian"`, the function randomly assigns the folds separately for ranges of the outcome. If `family` is `"gaussian"`, the function randomly assigns the folds separately for ranges of time and censorship. If `site` is not null, the function randomly assigns the folds separately for each site.

Value

A numeric vector with the fold assigned to each observation

Author(s)

Joaquim Radua and Aleix Solanes

References

Solanes, A., Mezquida, G., Janssen, J., Amoretti, S., Lobo, A., Gonzalez-Pinto, A., Arango, C., Vieta, E., Castro-Fornieles, J., Berge, D., Albacete, A., Gine, E., Parellada, M., Bernardo, M.; PEPs group (collaborators); Pomarol-Clotet, E., Radua, J. (2022) Combining MRI and clinical data to detect high relapse risk after the first episode of psychosis. *Schizophrenia*, **8**, 100, doi:10.1038/s41537-022-00309-w.

See Also

`cv` for conducting a cross-validation.

Examples

```
# Create random y (numeric)
y = rnorm(200, sample(c(1, 10), 200, replace = TRUE))

# Assign folds
fold = assign.folds(y, "gaussian", nfolds = 4)

# Check that the distribution of y is similar across folds
oldpar = par(mfrow = c(2, 2))
for (i in 1:4) {
  hist(y[which(fold == i)], main = paste("Fold", i), xlab = "y")
}
par(oldpar)
```

`cv` *Conduct cross-validation*

Description

Function to easily cross-validate (including fold assignation, merging fold outputs, etc).

Usage

```
cv(x, y, family = c("binomial", "cox", "gaussian"), fit_fun, predict_fun, site = NULL,
covar = NULL, nfolds = 10, pred.format = NA, verbose = TRUE, ...)
```

Arguments

`x`	input matrix for glmnet of dimension nobs x nvars; each row is an observation vector. It can be easily obtained with `data.frame2glmnet.matrix`.
`y`	response to be predicted. A binary vector for "binomial", a "Surv" object for "cox", or a numeric vector for "gaussian".
`family`	distribution of y: "binomial", "cox", or "gaussian".
`fit_fun`	function to create the prediction model using the training subsets. It can have between two and four arguments(the first two are compulsory): `x_training` (training X data.frame), `y_training` (training Y outcomes), `site_training` (training site names), and `covar_training` (training covariates). It must return the overall prediction model, which may be a list of the different submodels used in different steps and/or derived from different imputations.
`predict_fun`	function to apply the prediction model to the test sets. It can have between two and four arguments (the first two are compulsory): `model` (the overall prediction model), `x_test` (test X data.frame), `site_test` (test site names), and `covar_test` (test covariates). It must return the predictions.
`site`	vector with the sites' names, or NULL for studies conducted in a single site.
`covar`	other covariates that can be passed to fit_fun and predict_fun.
`...`	other arguments that can be passed to fit_fun and predict_fun.

`nfolds`	number of folds, only used if `folds` is NULL.
`pred.format`	format of the predictions returned by each fold. E.g., if the prediction is an array, use NA.
`verbose`	(optional) logical, whether to print some messages during execution.

Details

This function iteratively divides the dataset into a training dataset, with which fits the model using the function `fit_fun`, and a test dataset, to which applies the model using the function `predict_fun`. It saves the models fit with the training datasets and the predictions obtained in the test datasets. The fols are assigned automatically using `assign.folds`, accounting for the `site` is this is not null.

Value

A list with the predictions and the models used.

Author(s)

Joaquim Radua

See Also

`glmnet_predict` for obtaining predictions.

Examples

```
# Create random x (predictors) and y (binary)
x = matrix(rnorm(25000), ncol = 50)
y = 1 * (plogis(apply(x[,1:5], 1, sum) + rnorm(500, 0, 0.1)) > 0.5)

# Predict y via cross-validation
fit_fun = function (x_training, y_training) {
  list(
    lasso = glmnet_fit(x_training, y_training, family = "binomial")
  )
}
predict_fun = function (m, x_test) {
  glmnet_predict(m$lasso, x_test)
}
# Only 2 folds to ensure the example runs quickly
res = cv(x, y, family = "binomial", fit_fun = fit_fun, predict_fun = predict_fun, nfolds = 2)

# Show accuracy
se = mean(res$predictions$y.pred[res$predictions$y == 1] > 0.5)
sp = mean(res$predictions$y.pred[res$predictions$y == 0] < 0.5)
bac = (se + sp) / 2
cat("Sensitivity:", round(se, 2), "\n")
cat("Specificity:", round(sp, 2), "\n")
cat("Balanced accuracy:", round(bac, 2), "\n")
```

`data.frame2glmnet.matrix`

Convert a data.frame into a matrix ready for glmnet

Description

Function to convert categorical variables into dummy variables ready for `glmnet_fit` and `glmnet_predict`. Additionally, it also removes constant columns.

Usage

```
data.frame2glmnet.matrix_fit(x)
data.frame2glmnet.matrix(m, x)
```

Arguments

`m`	model to conduct the conversion, obtained with `data.frame2glmnet.matrix_fit`.
`x`	data.frame to be converted.

Details

Note that the returned matrix might differ from the design matrix of a linear model because for categoric variables with more than two levels, it creates as many dummy variables as levels (which is ok for lasso).

Value

A matrix ready for `glmnet_fit` and `glmnet_predict`.

Author(s)

Joaquim Radua and Aleix Solanes

See Also

`glmnet_predict` for obtaining predictions, `cv` for conducting a cross-validation.

Examples

```
# Create random x (predictors) and y (binary)
x = cbind(
  as.data.frame(matrix(rnorm(10000), ncol = 20)),
  matrix(sample(letters, 2500, TRUE), ncol = 5)
)
y = 1 * (plogis(apply(x[,1:5], 1, sum) + rnorm(500, 0, 0.1)) > 0.5)

# Predict y via cross-validation, including conversion to matrix
fit_fun = function (x_training, y_training) {
```

```
  m = list(
    matrix = data.frame2glmnet.matrix_fit(x_training)
  )
  x_mat = data.frame2glmnet.matrix(m$matrix, x_training)
  m$lasso = glmnet_fit(x_mat, y_training, family = "binomial")
  m
}
predict_fun = function (m, x_test) {
  x_mat = data.frame2glmnet.matrix(m$matrix, x_test)
  glmnet_predict(m$lasso, x_mat)
}
# Only 2 folds to ensure the example runs quickly
res = cv(x, y, family = "binomial", fit_fun = fit_fun, predict_fun = predict_fun, nfolds = 2)

# Show accuracy
se = mean(res$predictions$y.pred[res$predictions$y == 1] > 0.5)
sp = mean(res$predictions$y.pred[res$predictions$y == 0] < 0.5)
bac = (se + sp) / 2
cat("Sensitivity:", round(se, 2), "\n")
cat("Specificity:", round(sp, 2), "\n")
cat("Balanced accuracy:", round(bac, 2), "\n")
```

`glmnet_fit` *Obtain and use a glmnet prediction model*

Description

Function to easily fit and apply glmnet models (including best lambda estimation, etc).

Usage

```
glmnet_fit(x, y, family = c("binomial", "cox", "gaussian"),
            nfolds = 10, standardize = TRUE, min.beta = 1e-12)
glmnet_predict(m, x)
```

Arguments

`x`	input matrix of dimension nobs x nvars; each row is an observation vector. It can be easily obtained with `data.frame2glmnet.matrix`.
`y`	response to be predicted. A binary vector for "binomial", a "Surv" object for "cox", or a numeric vector for "gaussian".
`family`	distribution of y: "binomial", "cox", or "gaussian".
`m`	lasso model to conduct the prediction, obtained with `glmnet_fit`.
`nfolds`	number of folds.
`standardize`	logical flag for x variable standardization. The coefficients are always returned on the original scale.
`min.beta`	minimum value of betas.

Details

The function `glmnet_fit` mainly calls the function `glmnet` to fit a generalized linear model with lasso regularization, though with some extra code to make the call easier: it allow `x` to have a single column, it conducts an internal cross-validation using the function `cv.glmnet` to select the regularization parameter `lambda` automatically, and it removes the negligible coefficients.

Value

An object of class `"glmnet_fit"`, which is briefly a list with the intercept (`"a0"`) and regressors (`"beta"`) of the model; it also includes the indices of the regressors (`"i"`) and the `"family"` of the response.

Author(s)

Joaquim Radua and Aleix Solanes

References

Solanes, A., Mezquida, G., Janssen, J., Amoretti, S., Lobo, A., Gonzalez-Pinto, A., Arango, C., Vieta, E., Castro-Fornieles, J., Berge, D., Albacete, A., Gine, E., Parellada, M., Bernardo, M.; PEPs group (collaborators); Pomarol-Clotet, E., Radua, J. (2022) Combining MRI and clinical data to detect high relapse risk after the first episode of psychosis. *Schizophrenia*, **8**, 100, doi:10.1038/s41537-022-00309-w.

Palau, P., Solanes, A., Madre, M., Saez-Francas, N., Sarro, S., Moro, N., Verdolini, N., Sanchez, M., Alonso-Lana, S., Amann, B.L., Romaguera, A., Martin-Subero, M., Fortea, L., Fuentes-Claramonte, P., Garcia-Leon, M.A., Munuera, J., Canales-Rodriguez, E.J., Fernandez-Corcuera, P., Brambilla, P., Vieta, E., Pomarol-Clotet, E., Radua, J. (2023) Improved estimation of the risk of manic relapse by combining clinical and brain scan data. *Spanish Journal of Psychiatry and Mental Health*, **16**, 235–243, doi:10.1016/j.rpsm.2023.01.001.

See Also

`cv` for conducting a cross-validation.

Examples

```
# Create random x (predictors) and y (binary)
x = matrix(rnorm(25000), ncol = 50)
y = 1 * (plogis(apply(x[,1:5], 1, sum) + rnorm(500, 0, 0.1)) > 0.5)

# Predict y via cross-validation
fit_fun = function (x_training, y_training) {
  list(
    lasso = glmnet_fit(x_training, y_training, family = "binomial")
  )
}
predict_fun = function (m, x_test) {
  glmnet_predict(m$lasso, x_test)
}
# Only 2 folds to ensure the example runs quickly
```

```
res = cv(x, y, family = "binomial", fit_fun = fit_fun, predict_fun = predict_fun, nfold = 2)

# Show accuracy
se = mean(res$predictions$y.pred[res$predictions$y == 1] > 0.5)
sp = mean(res$predictions$y.pred[res$predictions$y == 0] < 0.5)
bac = (se + sp) / 2
cat("Sensitivity:", round(se, 2), "\n")
cat("Specificity:", round(sp, 2), "\n")
cat("Balanced accuracy:", round(bac, 2), "\n")
```

`glmnet_get.items.relevance`

Get the relevance of the model items

Description

Function to calculate the relevance of the items of a model or of a list of models.

Usage

```
glmnet_get.items.relevance(x, childname = NULL)
```

Arguments

`x`	an object of class `"glmnet_fit"`, a list of objects of class `"glmnet_fit"`, or a list of objects that have a child of class `"glmnet_fit"`.
`childname`	name of the child of class `"glmnet_fit"` (if `x`) is a list of objects that have a child of class `"glmnet_fit"`).

Details

The relevance is calculated as `abs( standardized_coefficient ) / sum(abs( standardized_coefficients ))`, as in the function `lasso_vars`.

Value

A numeric vector representing the relevance of the items of the model.

Author(s)

Joaquim Radua, based on the previous work of others (see Details)

References

Palau, P., Solanes, A., Madre, M., Saez-Francas, N., Sarro, S., Moro, N., Verdolini, N., Sanchez, M., Alonso-Lana, S., Amann, B.L., Romaguera, A., Martin-Subero, M., Fortea, L., Fuentes-Claramonte, P., Garcia-Leon, M.A., Munuera, J., Canales-Rodriguez, E.J., Fernandez-Corcuera, P., Brambilla, P., Vieta, E., Pomarol-Clotet, E., Radua, J. (2023) Improved estimation of the risk of manic relapse by combining clinical and brain scan data. *Spanish Journal of Psychiatry and Mental Health*, **16**, 235–243, doi:10.1016/j.rpsm.2023.01.001.

See Also

`glmnet_predict` for obtaining predictions, `cv` for conducting a cross-validation.

Examples

```
# Create random x (predictors) and y (binary)
x = matrix(rnorm(25000), ncol = 50)
y = 1 * (plogis(apply(x[,1:5], 1, sum) + rnorm(500, 0, 0.1)) > 0.5)

# Predict y via cross-validation
fit_fun = function (x_training, y_training) {
  list(
    lasso = glmnet_fit(x_training, y_training, family = "binomial")
  )
}
predict_fun = function (m, x_test) {
  glmnet_predict(m$lasso, x_test)
}
# Only 2 folds to ensure the example runs quickly
res = cv(x, y, family = "binomial", fit_fun = fit_fun, predict_fun = predict_fun, nfolds = 2)

# Show the relevance of the predictors
relevance = glmnet_get.items.relevance(res$models, "lasso")
relevance = relevance[which(relevance >= 0.01)] # Select items with >=1% relevance
round(relevance, 2)
```

`glmnet_get.main.model` *Get the main glmnet model across imputations and folds*

Description

Function to choose the `glmnet` model most similar to the other models on the list according to the Dice coefficient.

Usage

```
glmnet_get.main.model(x, childname = NULL, verbose = TRUE)
```

Arguments

`x`	a list of objects of class `"glmnet_fit"` or a list of objects that have a child of class `"glmnet_fit"`.
`childname`	name of the child of class `"glmnet_fit"` (if `x`) is a list of objects that have a child of class `"glmnet_fit"`).
`verbose`	(optional) logical, whether to print some messages during execution.

Details

If there are several instances of the most similar model, it averages them.

Value

An object of class "glmnet_fit", representing the model most similar to the other models of the list according to the Dice coefficient.

Author(s)

Joaquim Radua

References

Sobregrau, P., Bailles, E., Radua, J., Carreno, M., Donaire, A., Setoain, X., Bargallo, N., Rumia, J., Sanchez-Vives, M.V., Pintor, L. (2024) Design and validation of a diagnostic suspicion checklist to differentiate epileptic from psychogenic nonepileptic seizures (PNES-DSC). *Journal of Psychosomatic Research*, **180**, 111656, doi:10.1016/j.jpsychores.2024.111656.

See Also

`glmnet_predict` for obtaining predictions. `cv` for conducting a cross-validation.

Examples

```
# Create random x (predictors) and y (binary)
x = matrix(rnorm(25000), ncol = 50)
y = 1 * (plogis(apply(x[,1:5], 1, sum) + rnorm(500, 0, 0.1)) > 0.5)

# Predict y via cross-validation
fit_fun = function (x_training, y_training) {
  list(
    lasso = glmnet_fit(x_training, y_training, family = "binomial")
  )
}
predict_fun = function (m, x_test) {
  glmnet_predict(m$lasso, x_test)
}
# Only 2 folds to ensure the example runs quickly
res = cv(x, y, family = "binomial", fit_fun = fit_fun, predict_fun = predict_fun, nfolds = 2)

# Show the main model
lasso = glmnet_get.main.model(res$models, "lasso")
cat(
  "Model: ~plogis(", round(lasso$a0, 2), "+",
  paste0(round(lasso$beta, 2), "*", names(lasso$beta), collapse = " + "),
  ")\n"
)
```

`impute.glmnet.matrix_fit` *Impute missing variables in a glmnet matrix multiple times*

Description

Function to impute, multiple times, the missing variables in a `glmnet.matrix`. `impute.glmnet.matrix_fit` finds the "lasso" models to conduct the imputations, and `impute.glmnet.matrix` does the imputations (in the same or a different dataset).

Usage

```
impute.glmnet.matrix_fit(x, ncores = 1, verbose = TRUE)
impute.glmnet.matrix(m, x, nimp = 20, verbose = TRUE)
```

Arguments

`m`	model to conduct the imputations, obtained with `impute.glmnet.matrix_fit`.
`x`	input matrix for glmnet of dimension nobs x nvars; each row is an observation vector. It can be easily obtained with `data.frame2glmnet.matrix`.
`ncores`	number of number of worker nodes (for parallelization).
`nimp`	number of imputations
`verbose`	(optional) logical, whether to print some messages during execution.

Details

The user can then obtain a prediction from each dataset and combine the predictions using Rubin's rules (which usually means just averaging them). Note also that this function may take a lot of time.

Value

A list of complete matrixes ready for `glmnet_fit` and `glmnet_predict`.

Author(s)

Joaquim Radua and Aleix Solanes

References

Solanes, A., Mezquida, G., Janssen, J., Amoretti, S., Lobo, A., Gonzalez-Pinto, A., Arango, C., Vieta, E., Castro-Fornieles, J., Berge, D., Albacete, A., Gine, E., Parellada, M., Bernardo, M.; PEPs group (collaborators); Pomarol-Clotet, E., Radua, J. (2022) Combining MRI and clinical data to detect high relapse risk after the first episode of psychosis. *Schizophrenia*, **8**, 100, doi:10.1038/s41537-022-00309-w.

Palau, P., Solanes, A., Madre, M., Saez-Francas, N., Sarro, S., Moro, N., Verdolini, N., Sanchez, M., Alonso-Lana, S., Amann, B.L., Romaguera, A., Martin-Subero, M., Fortea, L., Fuentes-Claramonte, P., Garcia-Leon, M.A., Munuera, J., Canales-Rodriguez, E.J., Fernandez-Corcuera, P., Brambilla, P., Vieta, E., Pomarol-Clotet, E., Radua, J. (2023) Improved estimation of the risk of manic relapse by combining clinical and brain scan data. *Spanish Journal of Psychiatry and Mental Health*, **16**, 235–243, doi:10.1016/j.rpsm.2023.01.001.

See Also

`glmnet_predict` for obtaining predictions. `cv` for conducting a cross-validation.

Examples

```
# Quick example

# Create random x with missing values
x = matrix(rnorm(300), ncol = 3)
x = x + rnorm(1) * x[,sample(1:3)] + rnorm(1) * x[,sample(1:3)]
x[sample(1:300, 30)] = NA

# Impute missing values
m_impute = impute.glmnet.matrix_fit(x, ncores = 2)
x_imputed = impute.glmnet.matrix(m_impute, x)


# Complete example (it might take some time even if the example is simple...)

  # Create random x (predictors) and y (binary)
  x = matrix(rnorm(4000), ncol = 20)
  x = x + rnorm(1) * x[,sample(1:20)] + rnorm(1) * x[,sample(1:20)]
  y = 1 * (plogis(x[,1] - x[,2] + rnorm(200, 0, 0.1)) > 0.5)

  # Make some x missing values
  x[sample(1:4000, 400)] = NA

  # Predict y via cross-validation, including imputations
  fit_fun = function (x_training, y_training) {
    m = list(
    impute = impute.glmnet.matrix_fit(x_training, ncores = pmax(1, parallel::detectCores() - 2)),
      lasso = list()
    )
    x_imputed = impute.glmnet.matrix(m$impute, x_training)
    for (imp in 1:length(x_imputed)) {
      m$lasso[[imp]] = glmnet_fit(x_imputed[[imp]], y_training, family = ”binomial”)
    }
    m
  }
  predict_fun = function (m, x_test) {
    x_imputed = impute.glmnet.matrix(m$impute, x_test)
    y_pred = NULL
    for (imp in 1:length(x_imputed)) {
      y_pred = cbind(y_pred, glmnet_predict(m$lasso[[imp]], x_imputed[[imp]]))
    }
    apply(y_pred, 1, mean)
```

```
 }
 # Only 2 folds to ensure the example runs quickly
res = cv(x, y, family = "binomial", fit_fun = fit_fun, predict_fun = predict_fun, nfolds = 2)

 # Show accuracy
 se = mean(res$predictions$y.pred[res$predictions$y == 1] > 0.5)
 sp = mean(res$predictions$y.pred[res$predictions$y == 0] < 0.5)
 bac = (se + sp) / 2
 cat("Sensitivity:", round(se, 2), "\n")
 cat("Specificity:", round(sp, 2), "\n")
 cat("Balanced accuracy:", round(bac, 2), "\n")
```

`surv2binary` *Convert a "Surv" object into binary variables at different time points*

Description

Function to convert a `"Surv"` object (e.g., the predictions obtained from `glmnet_predict` using a `"cox"` model) into a list of binary variables (e.g., as obtained from `glmnet_predict` using a `"binomial"` model) at different time points.

Usage

```
surv2binary(x)
```

Arguments

`x`	a `"Surv"` object.

Details

This function is useful, for instance, to estimate the AUC at different timepoints from `"cox"` predictions.

Value

A list of times and binary variables.

Author(s)

Joaquim Radua

See Also

`glmnet_predict` for obtaining `"cox"` predictions. `cv` for conducting a cross-validation.

Examples

```
library(survival)
library(pROC)

# Create random x (predictors) and y (survival)
x = matrix(rnorm(5000), ncol = 10)
time = rexp(500)
y = Surv(time, plogis(x[,1] / pmax(1, time^2) + rnorm(500)) > 0.5)

# Predict y via cross-validation
fit_fun = function (x, y) {
  glmnet_fit(x, y, family = "cox")
}
predict_fun = function (m, x) {
  glmnet_predict(m, x)
}
res = cv(x, y, family = "cox", fit_fun = fit_fun, predict_fun = predict_fun)

# Convert y to binary
y.binary = surv2binary(y)

# Calculate and plot AUC for binary y at each timepoint
time_auc = NULL
for (i in 1:length(y.binary)) {
  status_i = y.binary[[i]]$status
  if (length(unique(na.omit(status_i))) == 2) {
    time_auc = rbind(time_auc, data.frame(
      time = y.binary[[i]]$time,
      auc = roc(status_i ~ res$predictions$y.pred, levels = 0:1, direction = "<")$auc
    ))
  }
}
plot(time_auc$time, time_auc$auc, type = "l", xlab = "Time", ylab = "AUC", ylim = 0:1)
abline(h = 0.5)
```

Index