Package ‘aglm’

June 9, 2021

Type Package

Title Accurate Generalized Linear Model

Version 0.4.0

Description Provides functions to fit Accurate Generalized Linear Model (AGLM) models, visualize them, and predict for new data. AGLM is defined as a regularized GLM which applies a sort of feature transformations using a discretization of numerical features and specific coding methodologies of dummy variables. For more information on AGLM, see Suguru Fujita, Toyoto Tanaka, Kenji Kondo and Hirokazu Iwasawa (2020) <https://www.institutdesactuaires.com/global/gene/link.php?doc_id=16273&fg=1>.


License GPL-2

Encoding UTF-8

Language en-US

RoxygenNote 7.1.1

Roxygen list(markdown = TRUE)

Depends R (>= 4.0.0),
Imports glmnet (>= 4.0.2),
assertthat,
methods,
mathjaxr

Suggests testthat,
knitr,
rmarkdown,
MASS,
faraway

RdMacros mathjaxr

R topics documented:

aglm-package ................................................................. 2
AccurateGLM-class .................................................... 4
aglm ................................................................. 4
AGLM_Input-class .................................................... 8
Provides functions to fit Accurate Generalized Linear Model (AGLM) models, visualize them, and predict for new data. AGLM is defined as a regularized GLM which applies a sort of feature transformations using a discretization of numerical features and specific coding methodologies of dummy variables. For more information on AGLM, see Suguru Fujita, Toyoto Tanaka, Kenji Kondo and Hirokazu Iwasawa (2020).

The collection of functions provided by the aglm package has almost the same structure as the famous glmnet package, so users familiar with the glmnet package will be able to handle it easily. In fact, this structure is reasonable in implementation, because what the aglm package does is applying appropriate transformations to the given data and passing it to the glmnet package as a backend.

The aglm package provides three different fitting functions, depending on how users want to handle hyper-parameters of AGLM models.

Because AGLM is based on regularized GLM, the regularization term of the loss function can be expressed as follows:

\[ R(\{\beta_{jk}\}; \lambda, \alpha) = \lambda \left\{ (1 - \alpha) \sum_{j=1}^{p} \sum_{k=1}^{m_j} |\beta_{jk}|^2 + \alpha \sum_{j=1}^{p} \sum_{k=1}^{m_j} |\beta_{jk}| \right\}, \]

where \( \beta_{jk} \) is the k-th coefficient of auxiliary variables for the j-th column in data, \( \alpha \) is a weight which controls how L1 and L2 regularization terms are mixed, and \( \lambda \) determines the strength of the regularization.
Searching hyper-parameters $\alpha$ and $\lambda$ is often useful to get better results, but usually time-consuming. That’s why the aglm package provides three fitting functions with different strategies for specifying hyper-parameters as follows:

- **aglm**: A basic fitting function with given $\alpha$ and $\lambda$ (s).
- **cv.aglm**: A fitting function with given $\alpha$ and cross-validation for $\lambda$.
- **cva.aglm**: A fitting function with cross-validation for both $\alpha$ and $\lambda$.

Generally speaking, setting an appropriate $\lambda$ is often important to get meaningful results, and using `cv.aglm()` with default $\alpha = 1$ (LASSO) is usually enough. Since `cva.aglm()` is much time-consuming than `cv.aglm()`, it is better to use it only if particularly better results are needed.

The following S4 classes are defined to store results of the fitting functions.

- **AccurateGLM-class**: A class for results of `aglm()` and `cv.aglm()`
- **CVA_AccurateGLM-class**: A class for results of `cva.aglm()`

**Using the fitted model**

Users can use models obtained from fitting functions in various ways, by passing them to following functions:

- `predict`: Make predictions for new data
- `plot`: Plot contribution of each variable and residuals
- `print`: Display textual information of the model
- `coef`: Get coefficients
- `deviance`: Get deviance
- `residuals`: Get residuals of various types

We emphasize that `plot()` is particularly useful to understand the fitted model, because it presents a visual representation of how variables in the original data are used by the model.

**Other functions**

The following functions are basically for internal use, but exported as utility functions for convenience.

- Functions for creating feature vectors
  - `getUDummyMatForOneVec`
  - `getODummyMatForOneVec`
  - `getLVarMatForOneVec`
- Functions for binning
  - `createEqualWidthBins`
  - `createEqualFreqBins`
  - `executeBinning`

**Author(s)**

- Kenji Kondo,
- Kazuhisa Takahashi and Hikari Banno (worked on L-Variable related features)
AccurateGLM-class

References


---

**AccurateGLM-class**

*Class for results of `aglm()` and `cv.aglm()`*

---

**Description**

Class for results of `aglm()` and `cv.aglm()`

**Slots**

- `backend_models` The fitted backend `glmnet` model is stored.
- `vars_info` A list, each of whose element is information of one variable.
- `lambda` Same as in the result of `cv.glmnet`.
- `cvm` Same as in the result of `cv.glmnet`.
- `cvsd` Same as in the result of `cv.glmnet`.
- `cvup` Same as in the result of `cv.glmnet`.
- `cvlo` Same as in the result of `cv.glmnet`.
- `nzero` Same as in the result of `cv.glmnet`.
- `name` Same as in the result of `cv.glmnet`.
- `lambda.min` Same as in the result of `cv.glmnet`.
- `lambda.1se` Same as in the result of `cv.glmnet`.
- `fit.preval` Same as in the result of `cv.glmnet`.
- `foldid` Same as in the result of `cv.glmnet`.
- `call` An object of class `call`, corresponding to the function call when this `AccurateGLM` object is created.

**Author(s)**

Kenji Kondo
**aglm**

*Fit an AGLM model with no cross-validation*

**Description**

A basic fitting function with given \( \alpha \) and \( \lambda \) (s). See aglm-package for more details on \( \alpha \) and \( \lambda \).

**Usage**

```r
aglm(
  x,
  y,
  qualitative_vars_UD_only = NULL,
  qualitative_vars_both = NULL,
  qualitative_vars_OD_only = NULL,
  quantitative_vars = NULL,
  use_LVar = FALSE,
  extrapolation = "default",
  add_linear_columns = TRUE,
  add_OD_columns_of_qualitatives = TRUE,
  add_interaction_columns = FALSE,
  OD_type_of_quantitatives = "C",
  nbin.max = NULL,
  bins_list = NULL,
  bins_names = NULL,
  family = c("gaussian", "binomial", "poisson"),
  ...
)
```

**Arguments**

- **x**
  A design matrix. Usually a data.frame object is expected, but a matrix object is fine if all columns are of a same class. Each column may have one of the following classes, and aglm will automatically determine how to handle it:
  - numeric: interpreted as a quantitative variable. aglm performs discretization by binning, and creates dummy variables suitable for ordered values (named O-dummies/L-variables).
  - factor (unordered) or logical: interpreted as a qualitative variable without order. aglm creates dummy variables suitable for unordered values (named U-dummies).
  - ordered: interpreted as a qualitative variable with order. aglm creates both O-dummies and U-dummies.

  These dummy variables are added to \( x \) and form a larger matrix, which is used internally as an actual design matrix. See our paper for more details on O-dummies, U-dummies, and L-variables.

  If you need to change the default behavior, use the following options: qualitative_vars_UD_only, qualitative_vars_both, qualitative_vars_OD_only, and quantitative_vars.

- **y**
  A response variable.
qualitative_vars_UD_only
Used to change the default behavior of aglm for given variables. Variables specified by this parameter are considered as qualitative variables and only U-dummies are created as auxiliary columns. This parameter may have one of the following classes:
- integer: specifying variables by index.
- character: specifying variables by name.

qualitative_vars_both
Same as qualitative_vars_UD_only, except that both O-dummies and U-dummies are created for specified variables.

qualitative_vars_OD_only
Same as qualitative_vars_UD_only, except that both only O-dummies are created for specified variables.

quantitative_vars
Same as qualitative_vars_UD_only, except that specified variables are considered as quantitative variables.

use_LVar
Set to use L-variables. By default, aglm uses O-dummies as the representation of a quantitative variable. If use_LVar=TRUE, L-variables are used instead.

extrapolation
Used to control values of linear combination for quantitative variables, outside where the data exists. By default, values of a linear combination outside the data is extended based on the slope of the edges of the region where the data exists. You can set extrapolation="flat" to get constant values outside the data instead.

add_linear_columns
By default, for quantitative variables, aglm expands them by adding dummies and the original columns, i.e. the linear effects, are remained in the resulting model. You can set add_linear_columns=FALSE to drop linear effects.

add_OD_columns_of_qualitatives
Set to FALSE if you do not want to use O-dummies for qualitative variables with order (usually, columns with ordered class).

add_interaction_columns
If this parameter is set to TRUE, aglm creates an additional auxiliary variable \( x_i \times x_j \) for each pair \((x_i, x_j)\) of variables.

OD_type_of_quantitatives
Used to control the shape of linear combinations obtained by O-dummies for quantitative variables (deprecated).

nbin.max
An integer representing the maximum number of bins when aglm perform binning for quantitative variables.

bins_list
Used to set custom bins for variables with O-dummies.

bins_names
Used to set custom bins for variables with O-dummies.

family
A family object or a string representing the type of the error distribution. Currently aglm supports gaussian, binomial, and poisson.

...
Other arguments are passed directly when calling glmnet().

Value
A model object fitted to the data. Functions such as predict and plot can be applied to the returned object. See AccurateGLM-class for more details.
Author(s)

• Kenji Kondo,
• Kazuhisa Takahashi and Hikari Banno (worked on L-Variable related features)

References


Examples

############################ Gaussian case ############################

```r
library(MASS) # For Boston
library(aglm)

## Read data
xy <- Boston # xy is a data.frame to be processed.
colnames(xy)[ncol(xy)] <- "y" # Let medv be the objective variable, y.

## Split data into train and test
n <- nrow(xy) # Sample size.
set.seed(2018) # For reproducibility.
ten <- sample(n, round(n/4)) # ID numbers for test data.
train <- xy[-ten,] # train is the data.frame for training.
x <- train[-ncol(xy)]
y <- train$y
newx <- test[-ncol(xy)]
y_true <- test$y

## Fit the model
model <- aglm(x, y) # alpha=1 (the default value)

## Predict for various alpha and lambda
lambda <- 0.1
y_pred <- predict(model, newx=newx, s=lambda)
rmse <- sqrt(mean((y_true - y_pred)^2))
cat(sprintf("RMSE for lambda=%.2f: %.5f \n\n", lambda, rmse))

lambda <- 1.0
y_pred <- predict(model, newx=newx, s=lambda)
rmse <- sqrt(mean((y_true - y_pred)^2))
cat(sprintf("RMSE for lambda=%.2f: %.5f \n\n", lambda, rmse))

alpha <- 0
model <- aglm(x, y, alpha=alpha)

lambda <- 0.1
y_pred <- predict(model, newx=newx, s=lambda)
rmse <- sqrt(mean((y_true - y_pred)^2))
cat(sprintf("RMSE for alpha=%.2f and lambda=%.2f: %.5f \n\n", alpha, lambda, rmse))
```

############################ Binomial case ############################
library(aglm)
library(faraway)

## Read data
xy <- nes96

## Split data into train and test
n <- nrow(xy) # Sample size.
set.seed(2018) # For reproducibility.
test.id <- sample(n, round(n/5)) # ID numbers for test data.
test <- xy[test.id,] # test is the data.frame for testing.
train <- xy[-test.id,] # train is the data.frame for training.
y <- train$vote

## Fit the model
model <- aglm(x, y, family="binomial")

## Make the confusion matrix
lambda <- 0.1
y_true <- test$vote
y_pred <- levels(y_true)[as.integer(predict(model, newx, s=lambda, type="class"))]
print(table(y_true, y_pred))

library(MASS) # For Boston
library(aglm)

## Randomly created train and test data
set.seed(2021)
sd <- 0.2
x <- 2 * runif(1000) + 1
f <- function(x){x^3 - 6 * x^2 + 13 * x}
y <- f(x) + rnorm(1000, sd = sd)
xy <- data.frame(x=x, y=y)
x_test <- seq(0.75, 3.25, length.out=101)
y_test <- f(x_test) + rnorm(101, sd=sd)
xy_test <- data.frame(x=x_test, y=y_test)

## Plot
nbin.max <- 10
models <- c(cv.aglm(x, y, use_LVar=FALSE, extrapolation="default", nbin.max=nbin.max),
            cv.aglm(x, y, use_LVar=FALSE, extrapolation="flat", nbin.max=nbin.max),
            cv.aglm(x, y, use_LVar=TRUE, extrapolation="default", nbin.max=nbin.max),
            cv.aglm(x, y, use_LVar=TRUE, extrapolation="flat", nbin.max=nbin.max))
titles <- c("O-Dummies with extrapolation="default"",
            "O-Dummies with extrapolation="flat"",
            "L-Variables with extrapolation="default"",
            "L-Variables with extrapolation="flat"")
par.old <- par(mfrow=c(2, 2))
for (i in 1:4) {
model <- models[[i]]
title <- titles[[i]]
pred <- predict(model, newx=x_test, s=model@lambda.min, type="response")

plot(x_test, y_test, pch=20, col="grey", main=title)
lines(x_test, f(x_test), lty="dashed", lwd=2) # the theoretical line
lines(x_test, pred, col="blue", lwd=3) # the smoothed line by the model
par(par.old)

AGLM_Input-class  
S4 class for input

Description

S4 class for input

Slots

vars_info  A list, each of whose element is information of one variable.
data  The original data.

coef.AccurateGLM  Get coefficients

Description

Get coefficients

Usage

## S3 method for class 'AccurateGLM'
coef(object, index = NULL, name = NULL, s = NULL, exact = FALSE, ...)

Arguments

object  A model object obtained from aglm() or cv.aglm().
index  An integer value representing the index of variable whose coefficients are required.
name  A string representing the name of variable whose coefficients are required. Note that if both index and name are set, index is discarded.
s  Same as in coef.glmnet.
exact  Same as in coef.glmnet.
...  Other arguments are passed directly to coef.glmnet().
Value

If index or name is given, the function returns a list with the one or combination of the following fields, consisting of coefficients related to the specified variable.

• `coef.linear`: A coefficient of the linear term. (If any)
• `coef.OD`: Coefficients of O-dummies. (If any)
• `coef.UD`: Coefficients of U-dummies. (If any)
• `coef.LV`: Coefficients of L-variables. (If any)

If both index and name are not given, the function returns entire coefficients corresponding to the internal designed matrix.

Author(s)

Kenji Kondo

createEqualFreqBins

Create bins (equal frequency binning)

Description

Create bins (equal frequency binning)

Usage

createEqualFreqBins(x_vec, nbin.max)

Arguments

x_vec A numeric vector, whose quantiles are used as breaks.
nbin.max The maximum number of bins.

Value

A numeric vector representing breaks obtained by binning. Note that the number of bins is equal to `min(nbin.max, length(x_vec))`.

Author(s)

Kenji Kondo
Create bins (equal width binning)

Usage

createEqualWidthBins(left, right, nbin)

Arguments

left 
The leftmost value of the interval to be binned.
right 
The rightmost value of the interval to be binned.
nbin 
The number of bins.

Value

A numeric vector representing breaks obtained by binning.

Author(s)

Kenji Kondo

Fit an AGLM model with cross-validation for \( \lambda \)

Description

A fitting function with given \( \alpha \) and cross-validation for \( \lambda \). See aglm-package for more details on \( \alpha \) and \( \lambda \).

Usage

cv.aglm(
  x,
y,
  qualitative_vars_UD_only = NULL,
  qualitative_vars_both = NULL,
  qualitative_vars_OD_only = NULL,
  quantitative_vars = NULL,
  use_LVar = FALSE,
  extrapolation = "default",
  add_linear_columns = TRUE,
  add_OD_columns_of_qualitatives = TRUE,
  add_interaction_columns = FALSE,
  OD_type_of_quantitatives = "C",
  nbin.max = NULL,
)
`cv.aglm`

```r
cv.aglm = function(x = NULL, y = NULL, qualitative_vars_UD_only = NULL,
                    qualitative_vars_both = NULL, qualitative_vars_OD_only = NULL,
                    quantitative_vars = NULL, use_LVar = NULL, extrapolation = NULL,
                    add_linear_columns = NULL, add_OD_columns_of_qualitatives = NULL,
                    add_interaction_columns = NULL, OD_type_of_quantitatives = NULL,
                    nbin.max = NULL, bins_list = NULL, bins_names = NULL,
                    family = c("gaussian", "binomial", "poisson"),
                    keep = FALSE, ...)
```

Arguments

- **x**: A design matrix. See `aglm` for more details.
- **y**: A response variable.
- **qualitative_vars_UD_only**: Same as in `aglm`.
- **qualitative_vars_both**: Same as in `aglm`.
- **qualitative_vars_OD_only**: Same as in `aglm`.
- **quantitative_vars**: Same as in `aglm`.
- **use_LVar**: Same as in `aglm`.
- **extrapolation**: Same as in `aglm`.
- **add_linear_columns**: Same as in `aglm`.
- **add_OD_columns_of_qualitatives**: Same as in `aglm`.
- **add_interaction_columns**: Same as in `aglm`.
- **OD_type_of_quantitatives**: Same as in `aglm`.
- **nbin.max**: Same as in `aglm`.
- **bins_list**: Same as in `aglm`.
- **bins_names**: Same as in `aglm`.
- **family**: Same as in `aglm`.
- **keep**: Set to `TRUE` if you need the `fit.preval` field in the returned value, as in `cv.glmnet()`. ... Other arguments are passed directly when calling `cv.glmnet()`.

Value

A model object fitted to the data with cross-validation results. Functions such as `predict` and `plot` can be applied to the returned object, same as the result of `aglm()`. See `AccurateGLM-class` for more details.

Author(s)

- Kenji Kondo,
- Kazuhisa Takahashi and Hikari Banno (worked on L-Variable related features)
References


Examples

################################### Cross-validation for lambda ###################################

library(aglm)
library(faraway)

## Read data
xy <- nes96

## Split data into train and test
n <- nrow(xy) # Sample size.
set.seed(2018) # For reproducibility.
test.id <- sample(n, round(n/5)) # ID numbers for test data.
test <- xy[test.id,] # test is the data.frame for testing.
train <- xy[-test.id,] # train is the data.frame for training.
y <- train$vote

# NOTE: Codes below will take considerable time, so run it when you have time.

## Fit the model
model <- cv.aglm(x, y, family="binomial")

## Make the confusion matrix
lambda <- model$lambda.min
y_true <- test$vote
y_pred <- levels(y_true)[as.integer(predict(model, newx, s=lambda, type="class"))]

cat(sprintf("Confusion matrix for lambda=%.5f:\n", lambda))
print(table(y_true, y_pred))

cva.aglm

Fit an AGLM model with cross-validation for both $\alpha$ and $\lambda$

Description

A fitting function with cross-validation for both $\alpha$ and $\lambda$. See aglm-package for more details on $\alpha$ and $\lambda$. 
Usage

cva.aglm(
  x,
  y,
  alpha = seq(0, 1, len = 11)^3,
  nfolds = 10,
  foldid = NULL,
  parallel.alpha = FALSE,
  ...
)

Arguments

x               A design matrix. See aglm for more details.
y               A response variable.
alpha            A numeric vector representing $\alpha$ values to be examined in cross-validation.
nfolds           An integer value representing the number of folds.
foldid           An integer vector with the same length as observations. Each element should take a value from 1 to nfolds, identifying which fold it belongs.
parallel.alpha   (not used yet)
...               Other arguments are passed directly to cv.aglm().

Value

An object storing fitted models and information of cross-validation. See CVA_AccurateGLM-class for more details.

Author(s)

• Kenji Kondo,
• Kazuhisa Takahashi and Hikari Banno (worked on L-Variable related features)

References


Examples

#################### Cross-validation for alpha and lambda ####################

library(aglm)
library(faraway)

## Read data
xy <- nes96

## Split data into train and test
n <- nrow(xy) # Sample size
set.seed(2018) # For reproducibility.
test.id <- sample(n, round(n/5)) # ID numbers for test data.
test <- xy[test.id,] # test is the data.frame for testing.
train <- xy[-test.id,] # train is the data.frame for training.
y <- train$vote

# NOTE: Codes bellow will take considerable time, so run it when you have time.

## Fit the model
cva_result <- cva.aglm(x, y, family="binomial")

alpha <- cva_result@alpha.min
lambda <- cva_result@lambda.min

mod_idx <- cva_result@alpha.min.index
model <- cva_result@models_list[[mod_idx]]

## Make the confusion matrix
y_true <- test$vote
y_pred <- levels(y_true)[as.integer(predict(model, newx, s=lambda, type="class"))]

cat(sprintf("Confusion matrix for alpha=%.5f and lambda=%.5f:\n", alpha, lambda))
print(table(y_true, y_pred))

CVA_AccurateGLM-class

Class for results of cva.aglm()

Description
Class for results of cva.aglm()

Slots
models_list A list consists of cv.glmnet()’s results for all α values.
alpha Same as in cv.glm.
nfolds Same as in cv.aglm.
alpha.min.index The index of alpha.min in the vector alpha.
alpha.min The α value achieving the minimum loss among all the values of alpha.
lambda.min The λ value achieving the minimum loss when α is equal to alpha.min.
call An object of class call, corresponding to the function call when this CVA_AccurateGLM object is created.

Author(s)
Kenji Kondo
### deviance.AccurateGLM

Get deviance

**Description**

Get deviance

**Usage**

```r
## S3 method for class 'AccurateGLM'
deviance(object, ...)
```

**Arguments**

- `object`: A model object obtained from `aglm()` or `cv.aglm()`.
- `...`: Other arguments are passed directly to `deviance.glmnet()`.

**Value**

The value of deviance extracted from the object `object`.

**Author(s)**

Kenji Kondo

### executeBinning

Binning the data to given bins.

**Description**

Binning the data to given bins.

**Usage**

```r
executeBinning(x_vec, breaks = NULL, nbin.max = 100, method = "freq")
```

**Arguments**

- `x_vec`: The data to be binned.
- `breaks`: A numeric vector representing breaks of bins (If `NULL`, automatically generated).
- `nbin.max`: The maximum number of bins (used only if `breaks=NULL`).
- `method`: "freq" for equal frequency binning or "width" for equal width binning (used only if `breaks=NULL`).

**Value**

A list with the following fields:

- `labels`: An integer vector with same length as `x_vec`, where `labels[i]=k` means the i-th element of `x_vec` is in the k-th bin.
- `breaks`: Breaks of bins used for binning.
getLVarMatForOneVec

Author(s)
Kenji Kondo

Description
Create L-variable matrix for one variable

Usage
getLVarMatForOneVec(x_vec, breaks = NULL, nbin.max = 100, only_info = FALSE)

Arguments
- x_vec: A numeric vector representing original variable.
- breaks: A numeric vector representing breaks of bins (If NULL, automatically generated).
- nbin.max: The maximum number of bins (used only if breaks=NULL).
- only_info: If TRUE, only information fields of returned values are filled and no dummy matrix is returned.

Value
A list with the following fields:
- breaks: Same as input
- dummy_mat: The created L-variable matrix (only if only_info=FALSE).

Author(s)
Kenji Kondo

getODummyMatForOneVec

Create a O-dummy matrix for one variable

Usage
getODummyMatForOneVec(
  x_vec, 
  breaks = NULL, 
  nbin.max = 100, 
  only_info = FALSE, 
  dummy_type = NULL
)

Description
Create a O-dummy matrix for one variable
Arguments

- **x_vec**: A numeric vector representing original variable.
- **breaks**: A numeric vector representing breaks of bins (if NULL, automatically generated).
- **nbin.max**: The maximum number of bins (used only if breaks=NULL).
- **only_info**: If TRUE, only information fields of returned values are filled and no dummy matrix is returned.
- **dummy_type**: Used to control the shape of linear combinations obtained by O-dummies for quantitative variables (deprecated).

Value

A list with the following fields:

- **breaks**: Same as input
- **dummy_mat**: The created O-dummy matrix (only if only_info=FALSE).

Author(s)

Kenji Kondo

getUDummyMatForOneVec  
Create a U-dummy matrix for one variable

Description

Create a U-dummy matrix for one variable

Usage

```r
getUDummyMatForOneVec(
  x_vec,
  levels = NULL,
  drop_last = TRUE,
  only_info = FALSE
)
```

Arguments

- **x_vec**: A vector representing original variable. The class of x_vec should be one of integer, character, or factor.
- **levels**: A character vector representing values of x_vec used to create U-dummies. If NULL, all the unique values of x_vec are used to create dummies.
- **drop_last**: If TRUE, the last column of the resulting matrix is dropped to avoid multicollinearity.
- **only_info**: If TRUE, only information fields of returned values are filled and no dummy matrix is returned.
Value

A list with the following fields:

- `levels`: Same as input.
- `drop_last`: Same as input.
- `dummy_mat`: The created U-dummy matrix (only if `only_info=FALSE`).

Author(s)

Kenji Kondo

---

**plot.AccurateGLM**

*Plot contribution of each variable and residuals*

Description

Plot contribution of each variable and residuals

Usage

```r
## S3 method for class 'AccurateGLM'
plot(
  x, vars = NULL, verbose = TRUE, s = NULL, resid = FALSE,
  smooth_resid = TRUE, smooth_resid_fun = NULL, ask = TRUE,
  layout = c(2, 2), only_plot = FALSE, main = "", add_rug = FALSE,
  ...
)
```

Arguments

- **x**: A model object obtained from `aglm()` or `cv.aglm()`.
- **vars**: Used to specify variables to be plotted (NULL means all the variables). This parameter may have one of the following classes:
  - `integer`: specifying variables by index.
  - `character`: specifying variables by name.
- **verbose**: Set to FALSE if textual outputs are not needed.
- **s**: A numeric value specifying $\lambda$ at which plotting is required. Note that plotting for multiple $\lambda$’s are not allowed and s always should be a single value. When the model is trained with only a single $\lambda$ value, just set it to NULL to plot for that value.
resid  Used to display residuals in plots. This parameter may have one of the following classes:
  • logical (single value): If TRUE, working residuals are plotted.
  • character (single value): type of residual to be plotted. See residuals.AccurateGLM for more details on types of residuals.
  • numerical (vector): residual values to be plotted.

smooth_resid  Used to display smoothing lines of residuals for quantitative variables. This parameter may have one of the following classes:
  • logical: If TRUE, smoothing lines are drawn.
  • character:
    – smooth_resid="both": Balls and smoothing lines are drawn.
    – smooth_resid="smooth_only": Only smoothing lines are drawn.

smooth_resid_fun  Set if users need custom smoothing functions.

ask  By default, plot() stops and waits inputs each time plotting for each variable is completed. Users can set ask=FALSE to avoid this. It is useful, for example, when using devices as bmp to create image files.

layout  Plotting multiple variables for each page is allowed. To achieve this, set it to a pair of integer, which indicating number of rows and columns, respectively.

only_plot  Set to TRUE if no automatic graphical configurations are needed.

main  Used to specify the title of plotting.

add_rug  Set to TRUE for rug plots.

...  Other arguments are currently not used and just discarded.

Value  
No return value, called for side effects.

Author(s)
  • Kenji Kondo,
  • Kazuhisa Takahashi and Hikari Banno (worked on L-Variable related features)

References

Examples

#################### using plot() and predict() ####################

library(MASS) # For Boston
library(aglm)

## Read data
xy <- Boston # xy is a data.frame to be processed.
colnames(xy)[ncol(xy)] <- "y" # Let medv be the objective variable, y.
## Split data into train and test
n <- nrow(xy) # Sample size.
set.seed(2018) # For reproducibility.
test.id <- sample(n, round(n/4)) # ID numbers for test data.
test <- xy[test.id,] # test is the data.frame for testing.
train <- xy[-test.id,] # train is the data.frame for training.
x <- train[-ncol(xy)]
y <- train$y
newx <- test[-ncol(xy)]
y_true <- test$y

## With the result of aglm()
model <- aglm(x, y)
lambda <- 0.1
plot(model, s=lambda, resid=TRUE, add_rug=TRUE, verbose=FALSE, layout=c(3, 3))
y_pred <- predict(model, newx=newx, s=lambda)
plot(y_true, y_pred)

## With the result of cv.aglm()
model <- cv.aglm(x, y)
lambda <- model@lambda.min
plot(model, s=lambda, resid=TRUE, add_rug=TRUE, verbose=FALSE, layout=c(3, 3))
y_pred <- predict(model, newx=newx, s=lambda)
plot(y_true, y_pred)

---

predict.AccurateGLM

Make predictions for new data

### Description
Make predictions for new data

### Usage

```
## S3 method for class 'AccurateGLM'
predict(
  object,
  newx = NULL,
  s = NULL,
  type = c("link", "response", "coefficients", "nonzero", "class"),
  exact = FALSE,
  newoffset,
  ...
)
```
predict.AccurateGLM

Arguments

- **object**: A model object obtained from `aglm()` or `cv.aglm()`.
- **newx**: A design matrix for new data. See the description of `x` in `aglm` for more details.
- **s**: Same as in `predict.glmnet`.
- **type**: Same as in `predict.glmnet`.
- **exact**: Same as in `predict.glmnet`.
- **newoffset**: Same as in `predict.glmnet`.
- **...**: Other arguments are passed directly when calling `predict.glmnet()`.

Value

The returned object depends on `type`. See `predict.glmnet` for more details.

Author(s)

- Kenji Kondo,
- Kazuhisa Takahashi and Hikari Banno (worked on L-Variable related features)

References


Examples

```
library(MASS) # For Boston
library(aglm)

## Split data into train and test
n <- nrow(xy) # Sample size.
set.seed(2018) # For reproducibility.
test.id <- sample(n, round(n/4)) # ID numbers for test data.
test <- xy[test.id,] # test is the data.frame for testing.
train <- xy[-test.id,] # train is the data.frame for training.
x <- train[-ncol(xy)]
y <- train$y
newx <- test[-ncol(xy)]
y_true <- test$y

## With the result of aglm()
model <- aglm(x, y)
lambda <- 0.1
plot(model, s=lambda, resid=TRUE, add_rug=TRUE,
     verbose=FALSE, layout=c(3, 3))
```
\begin{verbatim}
y_pred <- predict(model, newx=newx, s=lambda)
plot(y_true, y_pred)

## With the result of cv.aglm()
model <- cv.aglm(x, y)
lambda <- model@lambda.min

plot(model, s=lambda, resid=TRUE, add_rug=TRUE,
     verbose=FALSE, layout=c(3, 3))

y_pred <- predict(model, newx=newx, s=lambda)
plot(y_true, y_pred)
\end{verbatim}

\textbf{print.AccurateGLM} \hspace{1cm} Display textual information of the model

\textbf{Description}

Display textual information of the model

\textbf{Usage}

\begin{verbatim}
## S3 method for class 'AccurateGLM'
print(x, digits = max(3, getOption("digits") - 3), ...)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{x} \hspace{1cm} A model object obtained from \texttt{aglm()} or \texttt{cv.aglm()}.
  \item \texttt{digits} \hspace{1cm} Used to control significant digits in printout.
  \item ... \hspace{1cm} Other arguments are passed directly to \texttt{print.glmnet()}.
\end{itemize}

\textbf{Value}

No return value, called for side effects.

\textbf{Author(s)}

Kenji Kondo
residuals.AccurateGLM  Get residuals of various types

Description

Get residuals of various types

Usage

```r
## S3 method for class 'AccurateGLM'
residuals(
  object,
  x = NULL,
  y = NULL,
  offset = NULL,
  weights = NULL,
  type = c("working", "pearson", "deviance"),
  s = NULL,
  ...
)
```

Arguments

- **object** A model object obtained from `aglm()` or `cv.aglm()`.
- **x** A design matrix. If not given, x for fitting is used.
- **y** A response variable. If not given, y for fitting is used.
- **offset** An offset values. If not given, offset for fitting is used.
- **weights** Sample weights. If not given, weights for fitting is used.
- **type** A string representing type of deviance:
  - "working" get working residual
    \[ r_i^W = (y_i - \mu_i) \left( \frac{\partial \eta}{\partial \mu} \right)_{\mu=\mu_i}, \]
    where \( y_i \) is a response value, \( \mu \) is GLM mean, and \( \eta = g^{-1}(\mu) \) with the link function \( g \).
  - "pearson" get Pearson residuals
    \[ r_i^P = \frac{y_i - \mu_i}{\sqrt{V(\mu_i)}}, \]
    where \( V \) is the variance function.
  - "deviance" get deviance residuals
    \[ r_i^D = \text{sign}(y_i - \mu_i) \sqrt{d_i}, \]
    where \( d_i \) is the contribution to deviance.
- **s** A numeric value specifying \( \lambda \) at which residuals are calculated.
- **...** Other arguments are currently not used and just discarded.
**Value**

A numeric vector representing calculated residuals.

**Author(s)**

Kenji Kondo
Index

AccurateGLM-class, 3, 4, 6, 12
aglm, 3, 4, 11–13, 21
aglm-package, 2, 4, 11, 13
AGLM_Input-class, 8

ccoef, 3
ccoef.AccurateGLM, 9
ccoef.glmnet, 9
createEqualFreqBins, 3, 10
createEqualWidthBins, 3, 10
cv.aglm, 3, 11, 15
cv.glmnet, 4
cva.aglm, 3, 13
CVA_AccurateGLM-class, 3, 13, 14

device, 3
device.AccurateGLM, 15
executeBinning, 3, 16

getLVarMatForOneVec, 3, 16
getODummyMatForOneVec, 3, 17
getUDummyMatForOneVec, 3, 18

plot, 3
plot.AccurateGLM, 19
predict, 3
predict.AccurateGLM, 21
predict.glmnet, 21, 22
print, 3
print.AccurateGLM, 23

residuals, 3
residuals.AccurateGLM, 19, 23