Package ‘prettyglm’

September 6, 2023

Type Package

Title Pretty Summaries of Generalized Linear Model Coefficients

Version 1.0.1

Maintainer Jared Fowler <jared.fowler8@gmail.com>

Description One of the main advantages of using Generalised Linear Models is their interpretability. The goal of ‘prettyglm’ is to provide a set of functions which easily create beautiful coefficient summaries which can readily be shared and explained. ‘prettyglm’ helps users create coefficient summaries which include categorical base levels, variable importance and type III p.values. ‘prettyglm’ also creates beautiful relativity plots for categorical, continuous and splined coefficients.

License GPL-3

URL https://jared-fowler.github.io/prettyglm/

Depends R (>= 4.1.0)

Imports broom, car, dplyr, forcats, kableExtra, knitr, methods, plotly, RColorBrewer, stringr, tibble, tidycat, tidyr, tidyselect, vip

Suggests rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

Author Jared Fowler [cre, aut]

Repository CRAN

Date/Publication 2023-09-06 08:40:02 UTC
Description

Provides a rank plot of the actual and predicted.

Usage

```r
actual_expected_bucketed(
  target_variable,
  model_object,
  data_set = NULL,
  number_of_buckets = 25,
  ylab = "Target",
  width = 800,
  height = 500,
  first_colour = "black",
  second_colour = "#cc4678",
  facetby = NULL,
  prediction_type = "response",
  predict_function = NULL,
  return_data = F
)
```

Arguments

- `target_variable`
  - String of target variable name.
- `model_object`
  - GLM model object.
data_set  Data to score the model on. This can be training or test data, as long as the
data is in a form where the model object can make predictions. Currently develop-
ning ability to provide custom prediction functions, currently implementation
defaults to 'stats::predict'

number_of_buckets

number of buckets for percentile

ylab  Y-axis label.

width  plotly plot width in pixels.

height  plotly plot height in pixels.

first_colour  First colour to plot, usually the colour of actual.

second_colour  Second colour to plot, usually the colour of predicted.

facetby  variable user wants to facet by.

prediction_type

Prediction type to be pasted to predict.glm if predict_function is NULL. Defaults
to "response".

predict_function  prediction function to use. Still in development.

return_data  Logical to return cleaned data set instead of plot.

Value

plot Plotly plot by default. ggplot if plotlyplot = F. Tibble if return_data = T.

Examples

library(dplyr)
library(prettyglm)

data('titanic')

columns_to_factor <- c('Pclass',
    'Sex',
    'Cabin',
    'Embarked',
    'Cabin_type',
    'Survived')

meanage <- base::mean(titanic$Age, na.rm=TRUE)

titanic <- titanic %>%
    dplyr::mutate_at(columns_to_factor, list(~factor(.))) %>%
    dplyr::mutate(Age = base::ifelse(is.na(Age)==TRUE,meanage,Age)) %>%
    dplyr::mutate(Age_0_25 = prettyglm::splineit(Age,0,25),
    Age_25_50 = prettyglm::splineit(Age,25,50),
    Age_50_120 = prettyglm::splineit(Age,50,120)) %>%
    dplyr::mutate(Fare_0_250 = prettyglm::splineit(Fare,0,250),
    Fare_250_600 = prettyglm::splineit(Fare,250,600))
survival_model <- stats::glm(Survived ~
  Sex:Age +
  Fare +
  Embarked +
  SibSp +
  Parch +
  Cabintype,
  data = titanic,
  family = binomial(link = 'logit'))

prettyglm::actual_expected_bucketed(target_variable = 'Survived',
  model_object = survival_model,
  data_set = titanic)

bank_data  

**Bank marketing campaigns data set analysis**

**Description**

It is a dataset that describing Portugal bank marketing campaigns results. Conducted campaigns were based mostly on direct phone calls, offering bank client to place a term deposit. If after all marking efforts client had agreed to place deposit - target variable marked 'yes', otherwise 'no'

**Usage**

data(bank)

**Format**

An object of class "data.frame"

- **job** Type of job
- **marital** marital status
- **education** education
- **default** has credit in default?
- **housing** has housing loan?
- **loan** has personal loan?
- **age** age
- **y** has the client subscribed a term deposit? (binary: "yes","no")

**Details**

Sourse of the data https://archive.ics.uci.edu/ml/datasets/bank+marketing
clean_coefficients

References

This dataset is public available for research. The details are described in S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

Examples

data(bank)
head(bank_data)

Description

Processing to split out base levels and add variable importance to each term. Inspired by ‘tidy-cat::tidy_categorical()’, modified for use in prettyglm.

Usage

clean_coefficients(
  d = NULL,
  m = NULL,
  vimethod = "model",
  spline_seperator = NULL,
  ...)

Arguments

d          Data frame tibble output from tidy.lm; with one row for each term in the regression, including column ‘term’
m          Model object glm
vimethod   Variable importance method. Still in development
spline_seperator Sting of the spline separator. For example AGE_0_25 would be "_"
...        Any additional parameters to be past to vi

Value

Expanded tibble from the version passed to ‘d’ including additional columns:

variable The name of the variable that the regression term belongs to.
level The level of the categorical variable that the regression term belongs to. Will be an the term name for numeric variables.
Author(s)

Jared Fowler, Guy J. Abel

See Also
tidy.lm

cut3

cut3

description

Hmisc::cut2 bones repackaged to remove errors with importing Hmisc

Usage
cut3(
  x,
  cuts,
  m = 150,
  g,
  digits,
  minmax = TRUE,
  oneval = TRUE,
  onlycuts = FALSE,
  formatfun = format,
  ...
)

Arguments

x numeric vector to classify into intervals.
cuts cut points.
m desired minimum number of observations in a group. The algorithm does not guarantee that all groups will have at least m observations.
g number of quantile groups
digits number of significant digits to use in constructing levels.
minmax if cuts is specified but min(x)<min(cuts) or max(x)>max(cuts), augments cuts to include min and max x
oneval if an interval contains only one unique value, the interval will be labeled with the formatted version of that value instead of the interval endpoints, unless oneval=FALSE
onlycuts set to TRUE to only return the vector of computed cuts. This consists of the interior values plus outer ranges.
formatfun format function
... additional arguments passed to formatfun
**one_way_ave**

**Value**

vector of cut

---

**Description**

Creates a pretty html plot of one way actual vs expected by specified predictor.

**Usage**

```r
one_way_ave(
  feature_to_plot,
  model_object,
  target_variable,
  data_set,
  plot_type = "predictions",
  plot_factor_as_numeric = FALSE,
  ordering = NULL,
  width = 800,
  height = 500,
  number_of_buckets = 30,
  first_colour = "black",
  second_colour = "#cc4678",
  facetby = NULL,
  prediction_type = "response",
  predict_function = NULL,
  upper_percentile_to_cut = 0.01,
  lower_percentile_to_cut = 0
)
```

**Arguments**

- `feature_to_plot`: A string of the variable to plot.
- `model_object`: Model object to create coefficient table for. Must be of type: `glm`, `lm`.
- `target_variable`: String of target variable name in dataset.
- `data_set`: Data set to calculate the actual vs expected for. If no input default is to try and extract training data from model object.
- `plot_type`: one of "Residual", "predictions" or "actuals" defaults to "predictions".
- `plot_factor_as_numeric`: Set to TRUE to return `data.frame` instead of creating `kable`.
ordering  Option to change the ordering of categories on the x axis, only for discrete categories. Default to the ordering of the factor. Other options are: 'alphabetical', 'Number of records', 'Average Value'

width  Width of plot
height  Height of plot
number_of_buckets  Number of buckets for continuous variable plots
first_colour  First colour to plot, usually the colour of actual.
second_colour  Second colour to plot, usually the colour of predicted.
facetby  Variable to facet the actual vs expect plots by.
prediction_type  Prediction type to be pasted to predict.glm if predict_function is NULL. Defaults to 'response'.
predict_function  A custom prediction function can be provided here. It must return a data.frame with an "Actual_Values" column, and a "Predicted_Values" column.
upper_percentile_to_cut  For continuous variables this is what percentile to exclude from the upper end of the distribution. Defaults to 0.01, so the maximum percentile of the variable in the plot will be 0.99. Cutting off some of the distribution can help the views if outlier's are present in the data.
lower_percentile_to_cut  For continuous variables this is what percentile to exclude from the lower end of the distribution. Defaults to 0.01, so the minimum percentile of the variable in the plot will be 0.01. Cutting off some of the distribution can help the views if outlier's are present in the data.

Value

plotly plot of one way actual vs expected.

Examples

library(dplyr)
library(prettyglm)
data('titanic')
columns_to_factor <- c('Pclass', 'Sex', 'Cabin', 'Embarked', 'Cabin_type', 'Survived')
meanage <- base::mean(titanic$Age, na.rm=TRUE)
titanic <- titanic %>%
dplyr::mutate_at(columns_to_factor, list(~factor(.))) %>%
dplyr::mutate(Age = base::ifelse(is.na(Age)==TRUE, meanage, Age)) %>%
dplyr::mutate(Age_0_25 = prettyglm::splineit(Age, 0, 25),
Age_25_50 = prettyglm::splineit(Age,25,50),
Age_50_120 = prettyglm::splineit(Age,50,120)) %>%
dplyr::mutate(Fare_0_250 = prettyglm::splineit(Fare,0,250),
Fare_250_600 = prettyglm::splineit(Fare,250,600))
survival_model <- stats::glm(Survived ~
   Sex:Age +
   Fare +
   Embarked +
   SibSp +
   Parch +
   Cabintype,
data = titanic,
family = binomial(link = 'logit'))

# Continuous Variable Example
one_way_ave(feature_to_plot = 'Age',
model_object = survival_model,
target_variable = 'Survived',
data_set = titanic,
number_of_buckets = 20,
upper_percentile_to_cut = 0.1,
lower_percentile_to_cut = 0.1)

# Discrete Variable Example
one_way_ave(feature_to_plot = 'Pclass',
model_object = survival_model,
target_variable = 'Survived',
data_set = titanic)

# Custom Predict Function and facet
a_custom_predict_function <- function(target, model_object, dataset){
dataset <- base::as.data.frame(dataset)
Actual_Values <- dplyr::pull(dplyr::select(dataset, tidyselect::all_of(c(target))))
if(class(Actual_Values) == 'factor'){
   Actual_Values <- base::as.numeric(as.character(Actual_Values))
}
Predicted_Values <- base::as.numeric(stats::predict(model_object, dataset, type='response'))

to_return <- base::data.frame(Actual_Values = Actual_Values,
   Predicted_Values = Predicted_Values)

to_return <- to_return %>%
dplyr::mutate(Predicted_Values = base::ifelse(Predicted_Values > 0.3,0.3,Predicted_Values))
return(to_return)

one_way_ave(feature_to_plot = 'Age',
model_object = survival_model,
target_variable = 'Survived',
data_set = titanic,
number_of_buckets = 20,
upper_percentile_to_cut = 0.1,
predict_outcome(lower_percentile_to_cut = 0.1, predict_function = a_custom_predict_function, facetby = 'Pclass')

predict_outcome  predict_outcome

Description
Processing to predict response for various actual vs expected plots

Usage
predict_outcome(
  target,
  model_object,
  dataset,
  prediction_type = NULL,
  weights = NULL
)

Arguments
target String of target variable name.
model_object Model object. prettyglm currently supports
dataset This is used to plot the number in each class as a barchart if plotly is TRUE.
prediction_type type of prediction to be passed to the model object. For ...GLM defaults to ....
weights weightings to be provided to predictions if required.

Value
dataframe Returns a dataframe of Actual and Predicted Values

Author(s)
Jared Fowler

See Also
tidy.lm
pretty_coefficients

Description

Creates a pretty kable of model coefficients including coefficient base levels, type III P-values, and variable importance.

Usage

```r
pretty_coefficients(
  model_object,
  relativity_transform = NULL,
  relativity_label = "relativity",
  type_iii = NULL,
  conf.int = FALSE,
  vmethod = "model",
  spline_seperator = NULL,
  significance_level = 0.05,
  return_data = FALSE,
  ...
)
```

Arguments

- **model_object**: Model object to create coefficient table for. Must be of type: `glm`, `lm`.
- **relativity_transform**: String of the function to be applied to the model estimate to calculate the relativity, for example: `exp(estimate)-1`. Default is for relativity to be excluded from output.
- **relativity_label**: String of label to give to relativity column if you want to change the title to your use case.
- **type_iii**: Type III statistical test to perform. Default is none. Options are "Wald" or "LR". Warning 'LR' can be computationally expensive. Test performed via `Anova`.
- **conf.int**: Set to TRUE to include confidence intervals in summary table. Warning, can be computationally expensive.
- **vmethod**: Variable importance method to pass to method of `vi`. Defaults to "model". Currently supports "permute" and "firm", pass any additional arguments to `vi` in ...
- **spline_seperator**: Separator to look for to identity a spline. If this input is not null, it is assumed any features with this separator are spline columns. For example an age spline from 0 to 25 you could use: AGE_0_25 and "_".
- **significance_level**: Significance level to P-values by in kable. Defaults to 0.05.
return_data  Set to TRUE to return \texttt{data.frame} instead of creating \texttt{kable}.

... Any additional parameters to be past to \texttt{vi}

\textbf{Value}

\texttt{kable} if return\_data = FALSE. \texttt{data.frame} if return\_data = TRUE.

\textbf{Examples}

```r
library(dplyr)
library(prettyglm)
data('titanic')
columns_to_factor <- c('Pclass',
                     'Sex',
                     'Cabin',
                     'Embarked',
                     'Cabintype',
                     'Survived')
meanage <- base::mean(titanic$Age, na.rm=TRUE)
titanic <- dplyr::mutate_at(columns_to_factor, list(~factor(.)))
titanic <- dplyr::mutate(Age = base::ifelse(is.na(Age)==TRUE,meanage,Age))
titanic <- dplyr::mutate(Age_0_25 = prettyglm::splineit(Age,0,25),
                        Age_25_50 = prettyglm::splineit(Age,25,50),
                        Age_50_120 = prettyglm::splineit(Age,50,120))
titanic <- dplyr::mutate(Fare_0_250 = prettyglm::splineit(Fare,0,250),
                         Fare_250_600 = prettyglm::splineit(Fare,250,600))

# A simple example
survival_model <- stats::glm(Survived ~
                           Pclass +
                           Sex +
                           Age +
                           Fare +
                           Embarked +
                           SibSp +
                           Parch +
                           Cabintype,
                           data = titanic,
                           family = binomial(link = 'logit'))
pretty_coefficients(survival_model)

# A more complicated example with a spline and different importance method
survival_model3 <- stats::glm(Survived ~
                           Pclass +
                           Age_0_25 +
                           Age_25_50 +
                           Age_50_120 +
                           Sex:Fare_0_250 +
                           Sex:Fare_250_600 +
```
pretty_relativities

Description

Creates a pretty html plot of model relativities including base Levels.

Usage

```r
pretty_relativities(
  feature_to_plot,
  model_object,
  plot_approx_ci = TRUE,
  relativity_transform = "exp(estimate)-1",
  relativity_label = "Relativity",
  ordering = NULL,
  plot_factor_as_numeric = FALSE,
  width = 800,
  height = 500,
  interactionplottype = NULL,
  facetorcolourby = NULL,
  upper_percentile_to_cut = 0.01,
  lower_percentile_to_cut = 0,
  spline_seperator = NULL
)
```
Arguments

feature_to_plot
A string of the variable to plot.

model_object
Model object to create coefficient table for. Must be of type: glm, lm

plot_approx_ci
Set to TRUE to include confidence intervals in summary table. Warning, can be computationally expensive.

relativity_transform
String of the function to be applied to the model estimate to calculate the relativity, for example: 'exp(estimate)'. Default is for relativity to be 'exp(estimate)-1'.

relativity_label
String of label to give to relativity column if you want to change the title to your use case, some users may prefer to refer to this as odds ratio.

ordering
Option to change the ordering of categories on the x axis, only for discrete categories. Default to the ordering of the fitted factor. Other options are: 'alphabetical', 'Number of records', 'Average Value'

plot_factor_as_numeric
Set to TRUE to return data.frame instead of creating kable.

width
Width of plot

height
Height of plot

interactionplottype
If plotting the relativity for an interaction variable you can "facet" or "colour" by one of the interaction variables. Defaults to null.

facetorcolourby
If interactionplottype is not Null, then this is the variable in the interaction you want to colour or facet by.

upper_percentile_to_cut
For continuous variables this is what percentile to exclude from the upper end of the distribution. Defaults to 0.01, so the maximum percentile of the variable in the plot will be 0.99. Cutting off some of the distribution can help the views if outlier’s are present in the data.

lower_percentile_to_cut
For continuous variables this is what percentile to exclude from the lower end of the distribution. Defaults to 0.01, so the minimum percentile of the variable in the plot will be 0.01. Cutting off some of the distribution can help the views if outlier’s are present in the data.

spline_separator
String of the spline separator. For example AGE_0_25 would be "_".

Value

plotly plot of fitted relativities.
**Examples**

```r
library(dplyr)
library(prettyglm)
data('titanic')

columns_to_factor <- c('Pclass',
  'Sex',
  'Cabin',
  'Embarked',
  'Cabintype',
  'Survived')

meanage <- base::mean(titanic$Age, na.rm=TRUE)

titanic <- titanic %>%
dplyr::mutate_at(columns_to_factor, list(~factor(.))) %>%
dplyr::mutate(Age = base::ifelse(is.na(Age)==TRUE,meanage,Age)) %>%
dplyr::mutate(Age_0_25 = prettyglm::splineit(Age,0,25),
  Age_25_50 = prettyglm::splineit(Age,25,50),
  Age_50_120 = prettyglm::splineit(Age,50,120)) %>%
dplyr::mutate(Fare_0_250 = prettyglm::splineit(Fare,0,250),
  Fare_250_600 = prettyglm::splineit(Fare,250,600))

survival_model3 <- stats::glm(Survived ~
  Pclass:Embarked +
  Age_0_25 +
  Age_25_50 +
  Age_50_120 +
  Sex:Fare_0_250 +
  Sex:Fare_250_600 +
  SibSp +
  Parch +
  Cabintype,
  data = titanic,
  family = binomial(link = 'logit'))

# categorical factor
pretty_relativities(feature_to_plot = 'Cabintype',
  model_object = survival_model3)

# continuous factor
pretty_relativities(feature_to_plot = 'Parch',
  model_object = survival_model3)

# splined continuous factor
pretty_relativities(feature_to_plot = 'Age',
  model_object = survival_model3,
  spline_separator = '_',
  upper_percentile_to_cut = 0.01,
  lower_percentile_to_cut = 0.01)

# factor factor interaction
pretty_relativities(feature_to_plot = 'Pclass:Embarked',
  model_object = survival_model3)
```
model_object = survival_model3,
interactionplottype = 'colour',
facetorcolourby = 'Pclass')

# Continuous spline and categorical by colour
pretty_relativities(feature_to_plot = 'Sex:Fare',
model_object = survival_model3,
spline_seperator = '_')

# Continuous spline and categorical by facet
pretty_relativities(feature_to_plot = 'Sex:Fare',
model_object = survival_model3,
spline_seperator = '_',
interactionplottype = 'facet')

---

splineit

---

**Description**

Splines a continuous variable

**Usage**

splineit(var, min, max)

**Arguments**

- **var**: Continuous vector to spline.
- **min**: Min of spline.
- **max**: Max of spline.

**Value**

Splined Column

**Examples**

```r
library(dplyr)
library(prettyglm)
data('titanic')

columns_to_factor <- c('Pclass',
'Sex',
'Cabin',
'Embarked',
'Cabintype',
'Survived')

meanage <- base::mean(titanic$Age, na.rm=TRUE)
```
```r
titanic <- titanic %>%
dplyr::mutate_at(columns_to_factor, list(~factor(.))) %>%
dplyr::mutate(Age = base::ifelse(is.na(Age)==TRUE, mean_age, Age)) %>%
dplyr::mutate(Age_0_25 = prettyglm::splineit(Age, 0, 25),
              Age_25_50 = prettyglm::splineit(Age, 25, 50),
              Age_50_120 = prettyglm::splineit(Age, 50, 120)) %>%
dplyr::mutate(Fare_0_250 = prettyglm::splineit(Fare, 0, 250),
              Fare_250_600 = prettyglm::splineit(Fare, 250, 600))
```

### Description

The sinking of the Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren’t enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew. While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others. In this challenge, we ask you to build a predictive model that answers the question: “what sorts of people were more likely to survive?” using passenger data (ie name, age, gender, socio-economic class, etc).

### Usage

```r
data(titanic)
```

### Format

An object of class "data.frame"

- **survival**: Survival
- **pclass**: Ticket class
- **sex**: Sex
- **Age**: Age in years
- **sibsp**: number of siblings / spouses
- **parch**: number of parents / children
- **ticket**: Ticket number
- **fare**: Passenger fare
- **cabin**: Cabin Number
- **cabintype**: Type of cabin
- **embarked**: Port of Embarkation
References

This data set sourced from https://www.kaggle.com/c/titanic/data?select=train.csv

Examples

data(titanic)
head(titanic)
Index

* datasets
  bank_data, 4
  titanic, 17

actual_expected_bucketed, 2
Anova, 11

bank_data, 4

clean_coefficients, 5
cut3, 6
data.frame, 7, 8, 12, 14
glm, 5, 7, 11, 14
kable, 7, 12, 14
lm, 7, 11, 14
one_way_ave, 7

predict_outcome, 10
pretty_coefficients, 11
pretty_relativities, 11

splineit, 16

tibble, 5
tidy.lm, 5, 6, 10
titanic, 17

vi, 5, 11, 12