Package 'distreg.vis'

May 27, 2023

Type Package

Title Framework for the Visualization of Distributional Regression Models

Version 1.7.4

Maintainer Stanislaus Stadlmann <stanislaus@stadlmann.cm>

Depends R (>= 3.5.0)

Imports stats, utils, methods, shiny (>= 1.0.3), bamlss (>= 0.1-2), gamlss (>= 5.0-6), gamlss.dist (>= 5.1-0), ggplot2 (>= 2.2.1), rhandsontable (>= 0.3.4), magrittr (>= 1.5), formatR (>= 1.5), betareg (>= 3.1-2)

Suggests testthat, gridExtra, glogis

Description Functions for visualizing distributional regression models fitted using the 'gamlss', 'bamlss' or 'betareg' R package. The core of the package consists of a 'shiny' application, where the model results can be interactively explored and visualized.

License GPL-3

LazyData TRUE

URL https://github.com/Stan125/distreg.vis

BugReports https://github.com/Stan125/distreg.vis/issues

RoxygenNote 7.1.2

NeedsCompilation no

Author Stanislaus Stadlmann [cre, aut] (<https://orcid.org/0000-0001-6542-6342>)

Repository CRAN

Date/Publication 2023-05-27 00:10:02 UTC

R topics documented:

`distreg.vis` *distreg.vis: Interactively visualizing distributional regression models*

Description

The package `distreg.vis` is a framework for the visualization of distributional regression models estimated with the R packages `bamlss`, `gamlss` and `betareg`. Current supported model classes can be found under distreg_checker.

Details

The main functions are:

- `vis()`: Starts the Graphical User Interface.
- `moments()`: Obtain predicted moments of the target distribution based on user-specified values of the explanatory variables.
- `plot_dist()`: Create a graph displaying the predicted probability density function or cumulative density function based on the same user-specified values.
- `plot_moments()`: View the marginal influence of a selected effect on the predicted moments of the target distribution.

To get a feel for the main capabilities of `distreg.vis`, you can run the examples or the demo called `vis-demo.R` which fits a couple of distributional regression models and then calls the Graphical User Interface.

For the main functions, certain target distributions from both `bamlss` and `gamlss` are supported. Check the `distreg.vis::dists` dataset to find out which distributions are supported for `plot_dist()` (column `implemented`) and which are also supported for `plot_moments()` (column `moment_funs`).

To make the process of interpreting fitted distributional regression models as easy as possible, `distreg.vis` features a rich Graphical User Interface (GUI) built on the `shiny` framework. Using this GUI, the user can (a) obtain an overview of the selected model fit, (b) easily select explanatory values for which to display the predicted distributions, (c) obtain marginal influences of selected covariates and (d) change aesthetical components of each displayed graph. After a successful analysis, the user can quickly obtain the R code needed to reproduce all displayed plots, without having to start the application again.

Maintainer:

- Stanislaus Stadlmann, <stadlmann@uni-goettingen.de>

`distreg_checker` *Check if model class is supported*

Description

This function is a quick way to find out whether a specific model class is supported.

Usage

```
distreg_checker(x)
```

Arguments

`x`	Model object or model object in quoted form, e.g. "mymodel"

Details

This function is one of the cornerstones of distreg.vis. It decides which models are supported. All core functions of this package call `distreg_checker` multiple times. So, if a model class is support here, it is supported in the whole package.

At the moment, the following model classes are supported:

- gamlss
- bamlss
- betareg from `betareg`
- betatree from `betareg`

`dists` *Information about supported and not yet supported distribution families*

Description

A dataset containing all of bamlss' exported and gamlss.dist families. This is the backbone of the package; whether you can use a distributional family or not depends on this dataset. Since 1.7.0 family `betareg` from the betareg package is also supported.

Usage

```
dists
```

Format

An object of class `data.frame` with 125 rows and 8 columns.

Details

This `data.frame` object contains one row for each distribution, and columns with the following content:

- `dist_name`: Name of the distribution.
- `class`: Either "bamlss" or "gamlss" detailing from which package the target distribution comes from.
- `implemented`: Is this distribution generally usable for `plot_dist()`, and was this usage already tested?
- `moment_funs`: Are functions implemented with which to calculate the moments of the distribution, given the parameters? This column is especially relevant for `plot_moments()`, in which the predicted moments are displayed.
- `type_limits`: Details the range the values from the distribution can have. Can be "both_limits", "one_limit", "no_limit" and "cat_limit" (for categorical distributions).
- `l_limit, u_limit`: Integers detailing where the limits of the distributions lie.
- `type`: Character string for the type of distribution. Can be "Discrete", "Continuous", "Mixed" and "Categorical".

Examples

```
## Find out which GAMLSS or BAMLSS families are supported

dists_char <- dists[dists$moment_funs, c("dist_name", "class")]

# GAMLSS families
dists_char[dists_char$class == "gamlss", "dist_name"]

# BAMLSS families
dists_char[dists_char$class == "bamlss", "dist_name"]
```

---

model_data *Model data getter*

Description

Get the data with which the distributional regression model of interest was estimated (see distreg_checker for a list of supported object classes). By default, only explanatory variables are returned.

Usage

```
model_data(model, dep = FALSE, varname = NULL, incl_dep = FALSE)
```

Arguments

model	A gamlss or bamlss object.
dep	If TRUE, then only the dependent variable is returned.
varname	Variable name in character form that should be returned. If this is specified, only the desired variable is returned.
incl_dep	Should the dependent variable be included?

Value

A data.frame object if dep or varname is not specified, otherwise a vector.

Examples

```
library("betareg")

# Get some data
beta_dat <- model_fam_data(fam_name = "betareg")

# Estimate model
betamod <- betareg(betareg ~ ., data = beta_dat)

# Get data
model_data(betamod)
```

---

model_fam_data *Create a dataset to fit models with all possible families in distreg packages*

Description

Create a dataset to fit models with all possible families in distreg packages

Usage

```
model_fam_data(nrow = 500, seed = 1408, fam_name = "NO")
```

Arguments

nrow	Number of observations of the exported dataset.
seed	The seed which should be used, for reproducibility.
fam_name	The name of the distribution family to which the first dimension of the uniform distribution should be transformed to.

Details

This function creates a 3-dimensional uniform distribution (with support from 0 to 1) which has a cross-correlation of 0.5. Then the first dimension is transformed into a specified distribution (argument `fam_name`) via Inverse Transform Sampling https://en.wikipedia.org/wiki/Inverse_transform_sampling. The other two dimensions are transformed into a normal distribution (norm2) and a binomial distribution (binomial1, for testing categorical explanatory covariates). This procedure ensures that there is a dependency structure of the transformed first distribution and the other two.

Value

A data.frame with columns for differently distributed data.

Examples

```
# Beta distributed random values
model_fam_data(nrow = 500, fam_name = "BE")
```

---

moments *Compute distributional moments from the parameters*

Description

This function takes (predicted) parameters of a response distribution and calculates the corresponding distributional moments from it. Furthermore, you can specify own functions that calculate measures depending on distributional parameters.

Usage

```
moments(par, fam_name, what = "mean", ex_fun = NULL)
```

Arguments

`par`	Parameters of the modeled distribution in a data.frame form. Can be Output of `preds`, for example.
`fam_name`	Name of the used family in character form. Can be one of `distreg.vis::dists$dist_name`. All gamlss.dist and exported bamlss families are supported. To obtain the family from a model in character form, use `fam_obtainer`.
`what`	One of `"mean"`, `"upperlimit"`, `"lowerlimit"`. If it is `mean` (which is also the default), then the mean of the parameter samples is calculated. 2.5 for `lowerlimit` and `upperlimit`, respectively.
`ex_fun`	An external function `function(par) {...}` which calculates a measure, whose dependency from a certain variable is of special interest.

Details

With the exception of betareg, the distributional families behind the estimation of the distributional regression models are represented by own objects, e.g. GA or lognormal_bamlss. We worked together with both the authors of gamlss and bamlss such that the functions to compute the moments from the parameters of the underlying distribution is already implemented in the family functon itself. As an example, try out `gamlss.dist::BE()$mean`, which shows one example. The function `moments()` utilizes this fact and ensures that the outcome is always in the right format: Two columns named 'Expected_Value' and 'Variance' detailing the first two moments. One exception appears when an external function is specified, at which point there are three columns.

Each row details one 'scenario' meaning one covariate combination for which to predict the moments. `moments()` is heavily used in plot_moments, where moments are calculated over the entire range of one variable.

If target distribution stems from a bamlss model, `moments()` can also utilize the samples from the preds function to transform them. This is important for correct estimates, as just taking the mean of the samples and then using those means to estimate the moments can lead to inaccurate results. `moments()` knows when samples of predicted parameters were specified in the `par` argument, and then transforms the samples to the moments, before taking averages. Only through this procedure we even get credible intervals for the expected moments (see "upperlimit" and "lowerlimit" as possible outcomes of argument `what`).

Examples

```
# Get some artificial data
gamma_data <- model_fam_data(fam_name = "gamma", nrow = 100)

# Estimate model
library("bamlss")
model <- bamlss(list(gamma ~ norm2 + binomial1,
                     sigma ~ norm2 + binomial1),
                     data = gamma_data,
                     family = gamma_bamlss())

# Get some predicted parameters in sample and without sample form
pred_params <- preds(model, vary_by = "binomial1")
pred_params_samples <- preds(model, vary_by = "binomial1", what = "samples")

# Now calculate moments - with samples more correct estimates come out
moments(pred_params, fam_name = "gamma", what = "mean")
moments(pred_params_samples, fam_name = "gamma", what = "mean")

# Now with specifying an external function
my_serious_fun <- function(par) {
  return(par[["mu"]] + 3*par[["sigma"]])
}
moments(pred_params_samples,
        what = "mean",
        fam_name = "gamma",
        ex_fun = "my_serious_fun")
```

plot_dist *Plot predicted distributional regression models*

Description

This function plots the parameters of a predicted distribution (e.g. obtained through preds) with ggplot2. You can use all supported distributional regression model classes (check details of distreg_checker) as well as all supported distributional families (available at dists).

Usage

```
plot_dist(
  model,
  pred_params = NULL,
  palette = "viridis",
  type = "pdf",
  rug = FALSE,
  vary_by = NULL,
  newdata = NULL
)
```

Arguments

model	A fitted distributional regression model object. Check distreg_checker to see which classes are supported.
pred_params	A data.frame with rows for every model prediction and columns for every predicted parameter of the distribution. Is easily obtained with the distreg.vis function preds.
palette	The colour palette used for colouring the plot. You can use any of the ones supplied in scale_fill_brewer though I suggest you use one of the qualitative ones: Accent, Dark2, etc. Since 0.5.0 "viridis" is included, to account for colour blindness.
type	Do you want the probability distribution function ("pdf") or the cumulative distribution function ("cdf")?
rug	If TRUE, creates a rug plot
vary_by	Variable name in character form over which to vary the mean/reference values of explanatory variables. It is passed to set_mean. See that documentation for further details.
newdata	A data.frame object being passed onto preds. You can do this if you don't want to specify the argument pred_params directly. If you specify newdata, then preds(model, newdata = newdata) is going to be executed to be used as pred_params.

Details

To get a feel for the predicted distributions and their differences, it is best to visualize them. In combination with the obtained parameters from preds, the function `plot_dist()` looks for the necessary distribution functions (probability density function or cumulative distribution function) from the respective packages and then displays them graphically.

After `plot_dist()` has received all necessary arguments, it executes validity checks to ensure the argument's correct specification. This includes controlling for the correct `model` class, checking whether the distributional family can be used safely and whether cdf or pdf functions for the modeled distribution are present and ready to be graphically displayed. If this is the case, the internal fam_fun_getter is used to create a list with two functions pointing to the correct pdf and cdf functions in either the gamlss or bamlss namespace. The functions for betareg are stored in distreg.vis.

Following a successful calculation of the plot limits, the graph itself can be created. Internally, distreg.vis divides between continuous, discrete and categorical distributions. Continuous distributions are displayed as filled line plots, while discrete and categorical distributions take bar graph shapes.

For plotting, distreg.vis relies on the ggplot2 package (Wickham 2016). After an empty graph is constructed, the previously obtained cdf or pdf functions are evaluated for each predicted parameter combination and all values inside the calculated plot limits.

Value

A ggplot2 object.

References

Wickham H (2016). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. ISBN 978-3-319-24277-4. https://ggplot2.tidyverse.org.

Examples

```
# Generating data
data_fam <- model_fam_data(fam_name = "BE")

# Fit model
library("gamlss")
beta_model <- gamlss(BE ~ norm2 + binomial1,
  data = data_fam, family = BE())

# Obtains all explanatory variables and set them to the mean, varying by binomial1
# (do this if you do not want to specify ndata of preds by yourself)
ndata <- set_mean(model_data(beta_model), vary_by = "binomial1")

# Obtain predicted parameters
param_preds <- preds(beta_model, newdata = ndata)

# Create pdf, cdf plots
plot_dist(beta_model, param_preds, rug = TRUE)
plot_dist(beta_model, param_preds, type = "cdf")
plot_dist(beta_model, param_preds, palette = 'default')
```

```
# You can also let plot_dist do the step of predicting parameters of the mean explanatory variables:
plot_dist(beta_model, pred_params = NULL, vary_by = 'binomial1')
```

`plot_moments` *Plot function: Display the influence of a covariate*

Description

This function takes a dataframe of predictions with one row per prediction and one column for every explanatory variable. Then, those predictions are held constant while one specific variable is varied over it's whole range (min-max). Then, the constant variables with the varied interest variables are predicted and plotted against the expected value and the variance of the underlying distribution.

Usage

```
plot_moments(
  model,
  int_var,
  pred_data = NULL,
  rug = FALSE,
  samples = FALSE,
  uncertainty = FALSE,
  ex_fun = NULL,
  palette = "viridis",
  vary_by = NULL
)
```

Arguments

`model`	A fitted model on which the plots are based.
`int_var`	The variable for which influences of the moments shall be graphically displayed. Has to be in character form.
`pred_data`	Combinations of covariate data, sometimes also known as "newdata", including the variable of interest, which will be ignored in later processing.
`rug`	Should the resulting plot be a rug plot?
`samples`	If the provided model is a bamlss model, should the moment values be "correctly" calculated, using the transformed samples? See details for details.
`uncertainty`	If `TRUE`, displays uncertainty measures about the covariate influences. Can only be `TRUE` if samples is also `TRUE`.
`ex_fun`	An external function `function(par) {...}` which calculates a measure, whose dependency from a certain variable is of interest. Has to be specified in character form. See examples for an example.
`palette`	See `plot_dist`.
`vary_by`	Variable name in character form over which to vary the mean/reference values of explanatory variables. It is passed to set_mean. See that documentation for further details.

Details

The target of this function is to display the influence of a selected effect on the predicted moments of the modeled distribution. The motivation for computing influences on the moments of a distribution is its interpretability: In most cases, the parameters of a distribution do not equate the moments and as such are only indirectly location, scale or shape properties, making the computed effects hard to understand.

Navigating through the disarray of link functions, non-parametric effects and transformations to moments, `plot_moments()` supports a wide range of target distributions. See dists for details.

Whether a distribution is supported or not depends on whether the underlying `R` object possesses functions to calculate the moments of the distribution from the predicted parameters. To achieve this for as many distributional families as possible, we worked together with both the authors of gamlss (Rigby and Stasinopoulos 2005) and bamlss (Umlauf et al. 2018) and implemented the moment functions for almost all available distributions in the respective packages. The betareg family was implemented in distreg.vis as well.

References

Rigby RA, Stasinopoulos DM (2005). "Generalized Additive Models for Location, Scale and Shape." Journal of the Royal Statistical Society C, 54(3), 507-554.

Umlauf, N, Klein N, Zeileis A (2018). "BAMLSS: Bayesian Additive Models for Location, Scale and Shape (and Beyond)." Journal of Computational and Graphical Statistics, 27(3), 612-627.

Examples

```
# Generating some data
dat <- model_fam_data(fam_name = "LOGNO")

# Estimating the model
library("gamlss")
model <- gamlss(LOGNO ~ ps(norm2) + binomial1,
                ~ ps(norm2) + binomial1,
                data = dat, family = "LOGNO")

# Get newdata by either specifying an own data.frame, or using set_mean()
# for obtaining mean vals of explanatory variables
ndata_user <- dat[1:5, c("norm2", "binomial1")]
ndata_auto <- set_mean(model_data(model))

# Influence graphs
plot_moments(model, int_var = "norm2", pred_data = ndata_user) # cont. var
plot_moments(model, int_var = "binomial1", pred_data = ndata_user) # discrete var
plot_moments(model, int_var = "norm2", pred_data = ndata_auto) # with new ndata

# If pred_data argument is omitted plot_moments uses mean explanatory
# variables for prediction (using set_mean)
plot_moments(model, int_var = "norm2")

# Rug Plot
plot_moments(model, int_var = "norm2", rug = TRUE)
```

```
# Different colour palette
plot_moments(model, int_var = "binomial1", palette = "Dark2")

# Using an external function
ineq <- function(par) {
  2 * pnorm((par[["sigma"]] / 2) * sqrt(2)) - 1
}
plot_moments(model, int_var = "norm2", pred_data = ndata_user, ex_fun = "ineq")
```

`preds` *Predict parameters of a distreg models' target distribution*

Description

This function takes a fitted model and a dataframe with explanatory variables and a column for the intercept to compute predicted parameters for the specified distribution. Without worrying about class-specific function arguments, `preds()` offers a consistent way of obtaining predictions based on specific covariate combinations.

Usage

```
preds(model, newdata = NULL, what = "mean", vary_by = NULL)
```

Arguments

`model`	A fitted distributional regression model object. Check supported classes at distreg_checker.
`newdata`	A data.frame with explanatory variables as columns, and rows with the combinations you want to do predictions for. Furthermore, whether or not to include the intercept has to be specified via a logical variable `intercept`. If omitted, the average of the explanatory variables is used (see set_mean).
`what`	One of `"mean"` or `"samples"`. The default for bamlss models is "samples", while the default for gamlss models is "mean". This argument changes how the mean of the parameter is calculated. See details for details.
`vary_by`	Variable name in character form over which to vary the mean/reference values of explanatory variables. It is passed to set_mean. See that documentation for further details.

Value

A data.frame with one column for every distributional parameter and a row for every covariate combination that should be predicted.

Examples

```
# Generating data
data_fam <- model_fam_data(fam_name = "BE")

# Fit model
library("gamlss")
beta_model <- gamlss(BE ~ norm2 + binomial1,
  data = data_fam, family = BE())

# Get 3 predictions
ndata <- data_fam[sample(1:nrow(data_fam), 3), c("binomial1", "norm2")]
preds(model = beta_model, newdata = ndata)

# If newdata argument is omitted preds uses the means of the explanatory variables
preds(model = beta_model, newdata = NULL) # this gives the same results as ...
preds(model = beta_model, newdata = set_mean(model_data(beta_model))) # ...this
```

---

search_funs *function Searcher*

Description

Function that looks for objects of class 'function' in the working directory.

Usage

```
search_funs()
```

---

set_mean *Obtain mean values and reference categories of variables in a data.frame*

Description

This function purely exists for the `set_mean` argument of `plot_moments`. It takes a data.frame and obtains the mean values (numeric variables) and reference categories (categorical covariates).

Usage

```
set_mean(input, vary_by = NULL)
```

Arguments

`input`	A `data.frame` object
`vary_by`	A character string with the name of a variable over which the output dataframe should vary.

Value

A `data.frame` object with one row

Examples

```
library("betareg")

# Get some data
beta_dat <- model_fam_data(fam_name = "betareg")

# Estimate model
betamod <- betareg(betareg ~ ., data = beta_dat)

# Obtain explanatory variables and set to mean
set_mean(model_data(betamod))
set_mean(model_data(betamod), vary_by = "binomial1")
```

---

vis *distreg.vis function*

Description

Function to call the distreg.vis Shiny App which represents the core of this package.

Usage

```
vis()
```

Examples

```
library("gamlss")
library("bamlss")
# A gamlss model
normal_gamlss <- gamlss(NO ~ binomial1 + ps(norm2),
                        sigma.formula = ~ binomial1 + ps(norm2),
                        data = model_fam_data(),
                        trace = FALSE)

# Start the App - only in interactive modes
if (interactive()) {
distreg.vis::vis()
}
```

Index