Package 'regress3d'

May 9, 2026

Title Create 3D Regression Surfaces

Version 1.0.0

Description
Plot regression surfaces and marginal effects in three dimensions. The plots are 'plotly' objects and can be customized using functions and arguments from the 'plotly' package.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports broom, dplyr, magrittr, plotly, rlang, stats, tibble

Depends R (>= 3.5)

LazyData true

Suggests knitr, rmarkdown, scales, stargazer, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/ellaFosterMolina/regress3d, https://ellafostermolina.github.io/regress3d/

BugReports https://github.com/ellaFosterMolina/regress3d/issues

Config/Needs/website rmarkdown

NeedsCompilation no

Author Ella Foster-Molina [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-7455-7221>)

Maintainer Ella Foster-Molina <ella.fostermolina@gmail.com>

Repository CRAN

Date/Publication 2025-09-02 05:40:07 UTC

Contents

`add_3d_surface` *Add 3D regression surface to a plot_ly object*

Description

Add a 3 dimensional regression surface layer to a plot_ly object.

Usage

```
add_3d_surface(
  p,
  model,
  data = NULL,
  ci = TRUE,
  surfacecolor = "blue",
  surfacecolor_ci = "grey",
  opacity = 0.5,
  ...
)
```

Arguments

`p`	A plotly object.
`model`	An lm or glm with exactly two x variables
`data`	An optional dataframe to be used to estimate the regression surface. By default, this will be the data used by the inherited plotly object.
`ci`	An optional logical. Defaults to TRUE, showing the confidence intervals of the predicted effects.
`surfacecolor`	A color recognized by plotly. Used within the colorscale parameter in add_trace. Defaults to 'blue'.
`surfacecolor_ci`	A color recognized by plotly. Used within the colorscale parameter in add_trace. Defaults to 'grey'.
`opacity`	Sets the opacity of the surface. Defaults to 0.5.
`...`	Arguments (i.e., attributes) passed along to the trace `type`. See `schema()` for a list of acceptable attributes for a given trace `type` (by going to `traces -> type -> attributes`). Note that attributes provided at this level may override other arguments (e.g. `plot_ly(x = 1:10, y = 1:10, color = I("red"), marker = list(color = "blue"))`).

Details

Note that the data used to estimate the regression surface in model must be the same as the data called in plot_ly or specified by the argument `data`.

Additional plotly layers such as `add_markers()` can be added to the plotly plot, but be aware that many plotly layers inherit the data from the prior layer. As such, a function such as `add_markers()` may not work as intended if called after `add_3d_surface()`.

The surface can be built from either an lm or glm. For glms, testing has been primarily focused on binomial and Gamma families.

Value

A plotly object with the regression surface added to the plot.

Examples

```
library(plotly)
mymodel <- lm(length ~ isFemale_num + isMale_num, data = hair_data)
p1 <- plot_ly(data = hair_data,
              x = ~isFemale_num,
              y = ~isMale_num,
              z = ~length )
add_3d_surface(p1, model = mymodel, data = hair_data)
```

---

add_direction *A flexible function to add a line of predicted effects to the plotly surface with optional confidence intervals.*

Description

Primarily used by functions such as `add_3d_surface()` or `add_marginals()`. If user defines `direction_data` appropriately, any line can be shown.

Usage

```
add_direction(
  p,
  model,
  direction_data,
  direction_name = "User defined line",
  linecolor = "black",
  ci = TRUE
)
```

Arguments

p	A plotly object
model	A glm with exactly two x variables
direction_data	A data frame with a column of x1 values, a column of x2 values, predicted y values and optional predicted confidence interval for each pair of x values. The variable names must be c("rownum", actual x1 variable name, actual x2 variable name, actual y variable name, "lowerCI", "upperCI").
direction_name	The hover text for the plotted line(s). Defaults to "User defined line".
linecolor	The color for the plotted line. Defaults to "black".
ci	An optional logical. Defaults to TRUE, showing the confidence intervals of the predicted effects.

Value

A plotly object

Examples

```
library(plotly)
mymodel <- lm(r_shift ~ median_income16 + any_college, data = cali_counties)
xvars <- data.frame(x1 = seq(min(cali_counties$median_income16, na.rm=TRUE),
                             max(cali_counties$median_income16, na.rm=TRUE),
                              length.out=10),
                    x2 = seq(min(cali_counties$any_college, na.rm=TRUE),
                             max(cali_counties$any_college, na.rm=TRUE),
                             length.out=10))

predicted_xvars_data <- create_y_estimates(x_vals = xvars,
                                           model = mymodel,
                                           coefficient_names = c(y = "r_shift",
                                                                 x1= "median_income16",
                                                                 x2= "any_college") )
plot_ly( data = cali_counties,
         x = ~median_income16,
         y = ~any_college,
         z = ~r_shift) %>%
  add_markers(size = ~pop_estimate16, color = I('black')) %>%
  add_3d_surface(model = mymodel)%>%
  add_direction(model = mymodel, direction_data = predicted_xvars_data)
```

---

add_jitter *Jitter scattercloud points*

Description

Add a jitter to a scatter trace with the mode of markers.

Usage

```
add_jitter(
  p,
  x = NULL,
  y = NULL,
  z = NULL,
  data = NULL,
  x_jitter = NULL,
  y_jitter = NULL,
  z_jitter = NULL,
  ...
)
```

Arguments

p	a plotly object
x, y, z	an optional x, y, and/or z variable. Defaults to the data inherited from the plotly object p.
data	an optional data frame. Defaults to the data inherited from the plotly object p.
x_jitter, y_jitter, z_jitter	Amount of vertical, horizontal, and depth jitter. The jitter is added in both positive and negative directions, so the total spread is twice the value specified here. If omitted, defaults to 40% of the spread in the data, so the jitter values will occupy 80% of the implied bins.
...	Arguments (i.e., attributes) passed along to the trace type. See schema() for a list of acceptable attributes for a given trace type (by going to traces -> type -> attributes). Note that attributes provided at this level may override other arguments (e.g. plot_ly(x = 1:10, y = 1:10, color = I("red"), marker = list(color = "blue"))).

Details

This adds a small amount of random variation to the location of each point, and is a useful way of handling overplotting caused by discreteness. It is based on ggplot's ggplot2::geom_jitter().

The arguments x_jitter, y_jitter, z_jitter are not from plotly's syntax. If these arguments are misspelled, plot_ly will generate a warning message listing all valid arguments, but note that plotly uses the term attributes instead of arguments. Since regress3d is an add on to plotly, this list of valid attributes does not include the attributes/arguments created in this function.

Value

a plotly object

Examples

```
library(plotly)
plot_ly( data = hair_data,
             x = ~isFemale_num,
```

```
            y = ~isMale_num,
            z = ~length) %>%
  add_jitter( x_jitter = 0, z_jitter = 0, color = ~gender,
              colors = c("pink", "skyblue", "purple"))
```

---

add_marginals *Add 3d marginal effects to a plot_ly object*

Description

Add 3d marginal effects to a plot_ly plot

Usage

```
add_marginals(
  p,
  model,
  data = NULL,
  ci = TRUE,
  x1_constant_val = "mean",
  x2_constant_val = "mean",
  x1_color = "darkorange",
  x2_color = "crimson",
  x1_direction_name = "Predicted marginal effect of x1",
  x2_direction_name = "Predicted marginal effect of x2",
  omit_x1 = FALSE,
  omit_x2 = FALSE
)
```

Arguments

p	A plotly object
model	A lm or glm with exactly two x variables
data	An optional dataframe to be used to create the regression surface. By default, this will be the data used by the inherited plotly object.
ci	A logical. Defaults to TRUE, showing the confidence intervals of the predicted effects.
x1_constant_val, x2_constant_val	A string or numeric value indicating which constant value to set for x1 or x2 when the marginal effect of x2 is plotted. Defaults to the mean value. The string can take on "mean", "median", "min", or "max". Alternately, a numeric value may be specified.
x1_color	The color to be used for the line(s) depicting the marginal effect of x1. Defaults to "darkorange".
x2_color	The color to be used for the line(s) depicting the marginal effect of x2. Defaults to "crimson".

`x1_direction_name`
: The hover text for the plotted line(s). Defaults to "Predicted marginal effect of x1".

`x2_direction_name`
: The hover text for the plotted line(s). Defaults to "Predicted marginal effect of x2".

`omit_x1`, `omit_x2`
: An optional logical. Defaults to FALSE. If set to TRUE, the marginal effect for that variable will not be included.

Details

Additional plotly layers such as `add_markers()` can be added to the plotly plot, but be aware that many plotly layers inherit the data from the prior layer. As such, a function such as `add_markers()` may not work as intended if called after `add_marginals()`.

Value

A plotly object with the predicted marginal effects added to the plot.

Examples

```
library(plotly)
mymodel <- lm(r_shift ~ median_income16 + any_college,
              data = cali_counties, weight = pop_estimate16)
p <- plot_ly( data = cali_counties,
         x = ~median_income16,
         y = ~any_college,
         z = ~r_shift) %>%
  add_marginals(model = mymodel)
```

---

cali_counties *County level voting and demographics data for California counties.*

Description

Demographics sourced from census.gov in 2019. Voting records from https://github.com/tonmcg/US_County_Level_Election
16, sourced from The Guardian (2012) and Townhall.com (2016)

Usage

```
cali_counties
```

Format

`cali_counties` A data frame with 3142 rows and 65 columns:

FIPS Five digit Federal Information Processing Standards code that uniquely identifies counties and county equivalents in the United States

state State name

county_state County and state names for display

state_abbrv State abbreviation

county County name

pop_estimate16 Population in the county in 2016

any_college Percent of the county that attended college for some period of time, regardless of whether they got a degree.

college_2cat_num Coded as 1 if the county is categorized as high college attendance, zero if low. See college_2cat for categorization rule.

college_2cat High college attendance counties have over 51.24% college attendance (any_college). Low college attendance counties have under 51.24% college attendance. The mean value of any_college is 51.24.

prcnt_black Percent of the county that is Black.

prcnt_unemployed Percent of the population that is unemployed but looking for employment in 2016.

prcnt_unemployed_log Logged percent of the population that is unemployed but looking for employment in 2016.

median_income16 o Median household income in the county in 2016.

median_income16_1k Median household income in the county in 2016 in units of 1,000 dollars.

r_shift Percentage difference between the Republican presidential vote in that county in 2016 and 2012. For example, 46.7955% of Kent County in Delaware (FIPS 20001) voted for Romney in 2012. In 2016, 49.81482% of that county voted for Trump. Therefore, the county shifted towards the Republican presidential candidate by 3.01325%. Positive value mean leaning more Republican; negative values mean leaning less Republican.

prcnt_GOP16 Percent of the county that voted for the Republican presidential candidate, Donald Trump, in 2016.

plurality_Trump16 Binary: 0 if less than a plurality of the county that voted for the Republican presidential candidate, Donald Trump, in 2016. 1 otherwise

Source

R/cali_counties.R

county_data *County level voting and demographics data.*

Description

Demographics sourced from census.gov in 2019. Voting records from https://github.com/tonmcg/US_County_Level_Election
16, sourced from The Guardian (2012) and Townhall.com (2016)

Usage

```
county_data
```

Format

county_data A data frame with 3142 rows and 65 columns:

FIPS Five digit Federal Information Processing Standards code that uniquely identifies counties and county equivalents in the United States

state State name

county_state County and state names for display

state_abbrv State abbreviation

county County name

pop_estimate16 Population in the county in 2016

any_college Percent of the county that attended college for some period of time, regardless of whether they got a degree.

college_2cat_num Coded as 1 if the county is categorized as high college attendance, zero if low. See college_2cat for categorization rule.

college_2cat High college attendance counties have over 51.24% college attendance (any_college). Low college attendance counties have under 51.24% college attendance. The mean value of any_college is 51.24.

prcnt_black Percent of the county that is Black.

prcnt_unemployed Percent of the population that is unemployed but looking for employment in 2016.

prcnt_unemployed_log Logged percent of the population that is unemployed but looking for employment in 2016.

median_income16 o Median household income in the county in 2016.

median_income16_1k Median household income in the county in 2016 in units of 1,000 dollars.

r_shift Percentage difference between the Republican presidential vote in that county in 2016 and 2012. For example, 46.7955% of Kent County in Delaware (FIPS 20001) voted for Romney in 2012. In 2016, 49.81482% of that county voted for Trump. Therefore, the county shifted towards the Republican presidential candidate by 3.01325%. Positive value mean leaning more Republican; negative values mean leaning less Republican.

prcnt_GOP16 Percent of the county that voted for the Republican presidential candidate, Donald Trump, in 2016.

plurality_Trump16 Binary: 0 if less than a plurality of the county that voted for the Republican presidential candidate, Donald Trump, in 2016. 1 otherwise

Source

R/county_data.R

`create_surface_data` *Create data frame used to plot a surface of predicted y values*

Description

Create data frame used to plot a surface of predicted y values. There can be only exactly 2 columns of x values. The predicted y values can be estimated from an lm or glm model. Interaction terms are allowed, as are weights.

Usage

```
create_surface_data(data, model)
```

Arguments

`data`	A data frame being used to estimate the regression model
`model`	A glm with exactly two x variables

Value

A data frame with generated values for two x variables, as well as the predicted y values and predicted confidence intervals for each pair of x values. These can be used to plot the estimated regression surface and confidence interval surfaces.

Examples

```
mymodel <- lm(length ~ isFemale_num + isMale_num,
              data = hair_data)
surface_data <- create_surface_data(data = hair_data,
                                    model = mymodel)
```

`create_y_estimates` *Create predicted y values from a data frame of x values.*

Description

Create predicted y values from a data frame of x values. There can be only exactly 2 columns of x values. The predicted y values can be estimated from an lm or glm model. Interaction terms are allowed, as are weights.

Usage

```
create_y_estimates(x_vals, model, coefficient_names)
```

Arguments

`x_vals`	A data frame or tibble with exactly two columns. The first column has x1 values and the second column has x2 values. These will form a curve or line if plotted in the regression surface. The column names do not matter.
`model`	A glm with exactly two x variables
`coefficient_names`	A named character vector that attaches coefficient names to standardized names (x1, x2, y)

Value

A data frame with x values and their corresponding predicted y values, as well as 95% confidence intervals

Examples

```
mymodel <- lm(r_shift ~ median_income16 + any_college, data = cali_counties)
xvars <- data.frame(x1 = seq(min(cali_counties$median_income16, na.rm=TRUE),
                             max(cali_counties$median_income16, na.rm=TRUE),
                              length.out=10),
                    x2 = seq(min(cali_counties$any_college, na.rm=TRUE),
                             max(cali_counties$any_college, na.rm=TRUE),
                             length.out=10))

predicted_xvars_data <- create_y_estimates(x_vals = xvars,
                                           model = mymodel,
                                           coefficient_names = c(y = "r_shift",
                                                                 x1= "median_income16",
                                                                 x2= "any_college") )
```

`hair_data` *Simulated hair length data*

Description

A simulated dataset that shows plausible hair lengths for 3 gender categories.

Usage

```
hair_data
```

Format

`hair_data` A data frame with 1630 rows and 9 columns:

X Row number

gender Gender identification: male, female, or nonbinary

length Hair length in inches

isFemale_num numeric binary indicator for whether observation is female

isFemale Labels for isFemale_num

isMale_num numeric binary indicator for whether observation is male

isMale Labels for isMale_num

isNonbinary_num numeric binary indicator for whether observation is nonbinary

isNonbinary Labels for isNonbinary_num

Source

R/hair_data.R

Index