Package 'dotwhisker'

June 28, 2017

Type Package

Title Dot-and-Whisker Plots of Regression Results

Version 0.3.0

Date 2017-06-28

Maintainer Yue Hu <yue-hu-1@uiowa.edu>

Description Quick and easy dot-and-whisker plots of regression results.

Encoding UTF-8

BugReports https://github.com/fsolt/dotwhisker/issues

Depends R (>= 3.2.0), ggplot2, gridExtra, gtable

Imports grid, stats, broom, plyr, dplyr, stringr, ggstance

Suggests mfx, ordinal, tibble, knitr, rmarkdown

License MIT + file LICENSE

LazyData TRUE

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

Author Frederick Solt [aut],
Yue Hu [aut, cre],
Oliver Keyes [ctb],
Ben Bolker [ctb],
Stefan Müller [ctb]

Repository CRAN

Date/Publication 2017-06-28 03:50:23 UTC

R topics documented:

`add_brackets` *Add Labelled Brackets to Group Predictors in a Dot-and-Whisker Plot*

Description

`add_brackets` draws brackets along the y-axis beyond the plotting area of a dot-and-whisker plot generated by `dwplot`, useful for labelling groups of predictors

Usage

```
add_brackets(p, brackets, face = "italic")
```

Arguments

`p`	A dot-and-whisker plot generated by `dwplot`.
`brackets`	A list of brackets; each element of the list should be a character vector consisting of (1) a label for the bracket, (2) the name of the topmost variable to be enclosed by the bracket, and (3) the name of the bottommost variable to be enclosed by the bracket.
`face`	A typeface for the bracket labels; options are "plain", "bold", "italic", "oblique", and "bold.italic".

Value

The function returns a `gtable` object, which are viewed with `grid.arrange`.

To save, use `ggsave`.

Examples

```
library(broom)
library(dplyr)
data(mtcars)
m1 <- lm(mpg ~ wt + cyl + disp, data = mtcars)
m1_df <- broom::tidy(m1) %>% # create data.frame of regression results
    relabel_predictors(c("(Intercept)" = "Intercept",
                    wt = "Weight",
                    cyl = "Cylinder",
                    disp = "Displacement"))#'
p <- dwplot(m1_df, include_intercept = TRUE) +
   theme_bw() + xlab("Coefficient") + ylab("") +
   geom_vline(xintercept = 0, colour = "grey50", linetype = 2) +
   theme(legend.position="none")
```

```
two_brackets <- list(c("Engine", "Cylinder", "Displacement"),
c("Not Engine", "Intercept", "Weight"))
g <- p %>% add_brackets(two_brackets)
grid.arrange(g)  # to display

# to save (not run)
#ggsave(file = "plot.pdf", g)
```

---

by_2sd *Rescale regression results by multiplying by 2 standard deviations*

Description

by_2sd rescales regression results to facilitate making dot-and-whisker plots using dwplot.

Usage

```
by_2sd(df, dataset)
```

Arguments

df	A data.frame including the variables term (names of independent variables), estimate (corresponding coefficient estimates), std.error (corresponding standard errors), and optionally model (when multiple models are desired on a single plot) such as generated those by tidy.
dataset	The data analyzed in the models whose results are recorded in df

Details

by_2sd multiplies the results from regression models saved as tidy data frames for predictors that are not binary by twice the standard deviation of these variables in the dataset analyzed. Standardizing in this way yields coefficients that are directly comparable to those for untransformed binary predictors (Gelman 2008) and so facilitates plotting using dwplot. Note that the current version of by_2sd does not subtract the mean (in contrast to Gelman's (2008) formula). However, all estimates and standard errors of the independent variables are the same as if the mean was subtracted. The only difference to Gelman (2008) is that for all variables in the model the intercept is shifted by the coefficient times the mean of the variable.

An alternative available in some circumstances is to pass a model object to standardize before passing the results to tidy and then on to dwplot. The advantage of by_2sd is that it takes as its input is a tidy data.frame and so is not restricted to only those model objects that standardize accepts.

Value

A tidy data.frame

References

Gelman, Andrew. 2008. "Scaling Regression Inputs by Dividing by Two Standard Deviations." Statistics in Medicine, 27:2865-2873.

See Also

standardize

Examples

```
library(broom)
library(dplyr)

data(mtcars)
m1 <- lm(mpg ~ wt + cyl + disp, data = mtcars)
m1_df <- tidy(m1) %>% by_2sd(mtcars) # create data.frame of rescaled regression results
```

---

dwplot *Dot-and-Whisker Plots of Regression Results*

Description

dwplot is a function for quickly and easily generating dot-and-whisker plots of regression models saved in tidy data frames.

Usage

```
dwplot(x, alpha = 0.05, dodge_size = 0.4, order_vars = NULL,
  show_intercept = FALSE, model_name = "model", dot_args = list(size =
  0.3), ...)
```

Arguments

x	Either a tidy data.frame (see 'Details'), a model object to be tidied with tidy, or a list of such model objects.
alpha	A number setting the criterion of the confidence intervals. The default value is .05, corresponding to 95-percent confidence intervals.
dodge_size	A number (typically between 0 and 0.3) indicating how much vertical separation should be between different models' coefficients when multiple models are graphed in a single plot. Lower values tend to look better when the number of independent variables is small, while a higher value may be helpful when many models appear on the same plot.
order_vars	A vector of variable names that specifies the order in which the variables are to appear along the y-axis of the plot.
show_intercept	A logical constant indicating whether the coefficient of the intercept term should be plotted.

model_name	The name of a variable that distinguishes separate models within a tidy data.frame.
dot_args	A list of arguments specifying the appearance of the dots representing mean estimates. For supported arguments, see geom_pointrangeh.
...	Extra arguments to pass to tidy.

Details

dwplot visualizes regression results saved in tidy data.frames by, e.g., tidy as dot-and-whisker plots generated by ggplot.

Tidy data.frames to be plotted should include the variables term (names of predictors), estimate (corresponding estimates of coefficients or other quantities of interest), std.error (corresponding standard errors), and optionally model (when multiple models are desired on a single plot; a different name for this last variable may be specified using the model_name argument). In place of std.error one may substitute conf.low (the lower bounds of the confidence intervals of each estimate) and conf.high (the corresponding upper bounds).

For convenience, dwplot also accepts as input those model objects that can be tidied by tidy, or a list of such model objects.

Because the function takes a data.frame as input, it is easily employed for a wide range of models, including those not supported by tidy. And because the output is a ggplot object, it can easily be further customized with any additional arguments and layers supported by ggplot2. Together, these two features make dwplot extremely flexible.

Value

The function returns a ggplot object.

References

Kastellec, Jonathan P. and Leoni, Eduardo L. 2007. "Using Graphs Instead of Tables in Political Science." Perspectives on Politics, 5(4):755-771.

Examples

```
library(broom)
library(dplyr)
# Plot regression coefficients from a single model object
data(mtcars)
m1 <- lm(mpg ~ wt + cyl + disp, data = mtcars)
dwplot(m1) +
    xlab("Coefficient") + ylab("") +
    geom_vline(xintercept = 0, colour = "grey50", linetype = 2) +
    theme(legend.position="none")
# Plot regression coefficients from multiple models on the fly
m2 <- update(m1, . ~ . - disp)
dwplot(list(full = m1, nodisp = m2))
# Change the appearance of dots and whiskers
dwplot(m1, dot_args = list(size = 3, pch = 21, fill = "white"))
# Plot regression coefficients from multiple models in a tidy data.frame
by_trans <- mtcars %>% group_by(am) %>%
```

```
    do(tidy(lm(mpg ~ wt + cyl + disp, data = .))) %>% rename(model=am)
dwplot(by_trans) +
    theme_bw() + xlab("Coefficient") + ylab("") +
    geom_vline(xintercept = 0, colour = "grey60", linetype = 2) +
    ggtitle("Predicting Gas Mileage, OLS Estimates") +
    theme(plot.title = element_text(face = "bold"),
          legend.justification=c(0, 0), legend.position=c(0, 0),
          legend.background = element_rect(colour="grey80"),
          legend.title.align = .5) +
    scale_colour_grey(start = .4, end = .8,
                      name = "Transmission",
                      breaks = c(0, 1),
                      labels = c("Automatic", "Manual"))
```

`relabel_predictors` *Relabel the Predictors in a Tidy Data Frame of Regression Results*

Description

`relabel_predictors` is a convenience function for relabeling the predictors in a tidy data.frame to be passed to `dwplot`

Usage

```
relabel_predictors(x, replace = NULL)
```

Arguments

`x`	Either a plot generated by `dwplot` or a tidy data.frame to be passed to `dwplot`
`replace`	A named character vector, with new values as values, and old values as names

Value

The function returns a tidy data.frame.

Examples

```
library(broom)
library(dplyr)

data(mtcars)
m1 <- lm(mpg ~ wt + cyl + disp, data = mtcars)
m1_df <- broom::tidy(m1) %>%
         relabel_predictors(c("(Intercept)" = "Intercept",
                              wt = "Weight",
                              cyl = "Cylinder",
                              disp = "Displacement"))
dwplot(m1_df)
```

```
dwplot(m1, show_intercept = TRUE) %>%
    relabel_predictors(c("(Intercept)" = "Intercept",
                         wt = "Weight",
                         cyl = "Cylinder",
                         disp = "Displacement"))
```

---

relabel_y_axis *Relabel the Y-Axis of a Dot-Whisker Plot*

Description

relabel_y_axis DEPRECATED. A convenience function for relabeling the predictors on the y-axis of a dot-whisker plot created by dwplot. It is deprecated; use relabel_predictors instead.

Usage

```
relabel_y_axis(x)
```

Arguments

x	A vector of labels for predictors, listed from top to bottom

See Also

relabel_predictors to relabel predictors on the y-axis of a dot-whisker plot or in a tidy data.frame

---

secret_weapon *Generate a 'Secret Weapon' Plot of Regression Results from Multiple Models*

Description

secret_weapon is a function for plotting regression results of multiple models as a 'secret weapon' plot

Usage

```
secret_weapon(x, var = NULL, alpha = 0.05, dot_args = NULL,
  whisker_args = NULL)
```

Arguments

x	Either a tidy data.frame including results from multiple models (see 'Details') or a list of model objects that can be tidied with tidy
var	The predictor whose results are to be shown in the 'secret weapon' plot
alpha	A number setting the criterion of the confidence intervals. The default value is .05, corresponding to 95-percent confidence intervals.
dot_args	A list of arguments specifying the appearance of the dots representing mean estimates. For supported arguments, see geom_point.
whisker_args	A list of arguments specifying the appearance of the whiskers representing confidence intervals. For supported arguments, see geom_segment.

Details

Andrew Gelman has coined the term "the secret weapon" for dot-and-whisker plots that compare the estimated coefficients for a single predictor across many models or datasets. secret_weapon takes a tidy data.frame of regression results or a list of model objects and generates a dot-and-whisker plot of the results of a single variable across the multiple models.

Tidy data.frames to be plotted should include the variables term (names of predictors), estimate (corresponding estimates of coefficients or other quantities of interest), std.error (corresponding standard errors), and model (identifying the corresponding model). In place of std.error one may substitute lb (the lower bounds of the confidence intervals of each estimate) and ub (the corresponding upper bounds).

Alternately, secret_weapon accepts as input a list of model objects that can be tidied by tidy.

Value

The function returns a ggplot object.

Examples

```
library(broom)
library(dplyr)

# Estimate models across many samples, put results in a tidy data.frame
by_clarity <- diamonds %>% group_by(clarity) %>%
 do(broom::tidy(lm(price ~ carat + cut + color, data = .))) %>%
 ungroup %>% rename(model=clarity)

# Generate a 'secret weapon' plot of the results of diamond size
secret_weapon(by_clarity, "carat")
```

small_multiple *Generate a 'Small Multiple' Plot of Regression Results*

Description

small_multiple is a function for plotting regression results of multiple models as a 'small multiple' plot

Usage

```
small_multiple(x, dodge_size = 0.4, alpha = 0.05, show_intercept = FALSE,
  dot_args = list(size = 0.3))
```

Arguments

x	Either a tidy data.frame including results from multiple models (see 'Details') or a list of model objects that can be tidied with tidy
dodge_size	A number (typically between 0 and 0.3; the default is .06) indicating how much horizontal separation should appear between different submodels' coefficients when multiple submodels are graphed in a single plot. Lower values tend to look better when the number of models is small, while a higher value may be helpful when many submodels appear on the same plot.
alpha	A number setting the criterion of the confidence intervals. The default value is .05, corresponding to 95-percent confidence intervals.
show_intercept	A logical constant indicating whether the coefficient of the intercept term should be plotted
dot_args	A list of arguments specifying the appearance of the dots representing mean estimates. For supported arguments, see geom_pointrangeh.

Details

Kastellec and Leoni (2007) small_multiple takes a tidy data.frame of regression results or a list of model objects and generates a dot-and-whisker plot of the results of a single variable across the multiple models.

Tidy data.frames to be plotted should include the variables term (names of predictors), estimate (corresponding estimates of coefficients or other quantities of interest), std.error (corresponding standard errors), and model (identifying the corresponding model). In place of std.error one may substitute lb (the lower bounds of the confidence intervals of each estimate) and ub (the corresponding upper bounds).

Alternately, small_multiple accepts as input a list of model objects that can be tidied by tidy.

Optionally, more than one set of results can be clustered to facilitate comparison within each model; one example of when this may be desirable is to compare results across samples. In that case, the data.frame should also include a variable submodel identifying the submodel of the results.

Value

The function returns a ggplot object.

Examples

```
library(broom)
library(dplyr)

# Generate a tidy data.frame of regression results from six models

m <- list()
ordered_vars <- c("wt", "cyl", "disp", "hp", "gear", "am")
m[[1]] <- lm(mpg ~ wt, data = mtcars)
m123456_df <- m[[1]] %>% tidy %>% by_2sd(mtcars) %>%
  mutate(model = "Model 1")

for (i in 2:6) {
 m[[i]] <- update(m[[i-1]], paste(". ~ . +", ordered_vars[i]))
 m123456_df <- rbind(m123456_df, m[[i]] %>% tidy %>% by_2sd(mtcars) %>%
   mutate(model = paste("Model", i)))
}

# Generate a 'small multiple' plot
small_multiple(m123456_df)


## Using submodels to compare results across different samples
# Generate a tidy data.frame of regression results from five models on
# the mtcars data subset by transmission type (am)
ordered_vars <- c("wt", "cyl", "disp", "hp", "gear")
mod <- "mpg ~ wt"
by_trans <- mtcars %>% group_by(am) %>%  # group data by transmission
  do(tidy(lm(mod, data = .))) %>%        # run model on each group
  rename(submodel = am) %>%              # make submodel variable
  mutate(model = "Model 1") %>%          # make model variable
 ungroup()

for (i in 2:5) {
   mod <- paste(mod, "+", ordered_vars[i])
   by_trans <- rbind(by_trans, mtcars %>% group_by(am) %>%
                         do(tidy(lm(mod, data = .))) %>%
                         rename(submodel = am) %>%
                         mutate(model = paste("Model", i)) %>%
                         ungroup())
}

small_multiple(by_trans) +
theme_bw() + ylab("Coefficient Estimate") +
    geom_hline(yintercept = 0, colour = "grey60", linetype = 2) +
    theme(axis.text.x  = element_text(angle = 45, hjust = 1),
          legend.position=c(0, 0), legend.justification=c(0, 0),
          legend.title = element_text(size=9),
```

```
      legend.background = element_rect(color="gray90"),
      legend.margin = unit(-3, "pt"),
      legend.key.size = unit(10, "pt")) +
scale_colour_hue(name = "Transmission",
breaks = c(0, 1),
labels = c("Automatic", "Manual"))
```

Index