Package 'dotwhisker'

October 6, 2015

Type Package

Title Dot-and-Whisker Plots of Regression Results

Version 0.2.0.1

Date 2015-10-06

Author Frederick Solt <frederick-solt@uiowa.edu>, Yue Hu <yue-hu-1@uiowa.edu>

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

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

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

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

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

Suggests mfx, ordinal, knitr

License MIT + file LICENSE

LazyData TRUE

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2015-10-06 22:17:27

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.draw.

To save, wrap the grid.draw command with pdf or png, etc., and dev.off. Alternately, the next release of ggplot2 (>1.0.1.9002) will enable ggsave for gtables; one can install the development version using devtools::install_github("hadley/ggplot2").

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

p <- dwplot(m1_df) +
   scale_y_discrete(breaks = 4:1, labels=c("Intercept", "Weight", "Cylinders", "Displacement")) +
    theme_bw() + xlab("Coefficient") + ylab("") +
    geom_vline(xintercept = 0, colour = "grey50", linetype = 2) +
    theme(legend.position="none")

two_brackets <- list(c("Engine", "cyl", "disp"), c("Not Engine", "(Intercept)", "wt"))

g <- p %>% add_brackets(two_brackets)
```

```
grid.draw(g)  # to display

#pdf("plot.pdf")  # to save (not run)
#grid.draw(g)
#dev.off()
```

---

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.

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 = 0.15)
```

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.

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). 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).

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) +
   scale_y_discrete(breaks = 4:1, labels=c("Intercept", "Weight", "Cylinders", "Displacement")) +
    theme_bw() + xlab("Coefficient") + ylab("") +
    geom_vline(xintercept = 0, colour = "grey50", linetype = 2) +
    theme(legend.position="none")

# Plot regression coefficients from multiple models in a tidy data.frame
library(dplyr)
by_trans <- mtcars %>% group_by(am) %>%
    do(tidy(lm(mpg ~ wt + cyl + disp, data = .))) %>% rename(model=am)

dwplot(by_trans, dodge_size = .05) +
   scale_y_discrete(breaks = 4:1, labels=c("Intercept", "Weight", "Cylinders", "Displacement")) +
    theme_bw() + xlab("Coefficient Estimate") + 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(1,0), legend.position=c(1,0),
          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(df, replace = NULL)
```

Arguments

df	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.

See Also

relabel_y_axis to relabel the predictors on the y-axis of a dot-whisker plot after using dwplot

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)
```

---

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

Description

relabel_y_axis is a convenience function for relabeling the predictors on the y-axis of a dot-whisker plot created by dwplot

Usage

```
relabel_y_axis(x)
```

Arguments

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

See Also

relabel_predictors to relabel the predictors in a tidy data.frame before using dwplot

Examples

```
data(mtcars)
m1 <- lm(mpg ~ wt + cyl + disp, data = mtcars)
dwplot(m1) + relabel_y_axis(c("Intercept", "Weight", "Cylinders", "Displacement"))
```

`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)
```

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.

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 = 0.06, alpha = 0.05)
```

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.

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 desireable 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.

Note

Ideally, the y-axes of small multiple plots would vary by predictor, but small_multiple does not currently support this behavior.

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

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)))
}

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