# Package ‘dotwhisker’

August 22, 2021

<table>
<thead>
<tr>
<th>Type</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Dot-and-Whisker Plots of Regression Results</td>
</tr>
<tr>
<td>Version</td>
<td>0.7.3</td>
</tr>
<tr>
<td>Date</td>
<td>2021-08-21</td>
</tr>
<tr>
<td>Maintainer</td>
<td>Yue Hu <a href="mailto:yuehu@tsinghua.edu.cn">yuehu@tsinghua.edu.cn</a></td>
</tr>
<tr>
<td>Description</td>
<td>Quick and easy dot-and-whisker plots of regression results.</td>
</tr>
<tr>
<td>Encoding</td>
<td>UTF-8</td>
</tr>
<tr>
<td>BugReports</td>
<td><a href="https://github.com/fsolt/dotwhisker/issues">https://github.com/fsolt/dotwhisker/issues</a></td>
</tr>
<tr>
<td>Depends</td>
<td>R (&gt;= 3.2.0), ggplot2 (&gt;= 2.2.1)</td>
</tr>
<tr>
<td>Imports</td>
<td>grid, stats, parameters, margins, dplyr, stringr, ggstance, rlang, purrr, gtable</td>
</tr>
<tr>
<td>Suggests</td>
<td>ordinal, tibble, gridExtra, knitr, rmarkdown, broom</td>
</tr>
<tr>
<td>License</td>
<td>MIT + file LICENSE</td>
</tr>
<tr>
<td>VignetteBuilder</td>
<td>knitr</td>
</tr>
<tr>
<td>RoxygenNote</td>
<td>7.1.1</td>
</tr>
<tr>
<td>NeedsCompilation</td>
<td>no</td>
</tr>
<tr>
<td>Author</td>
<td>Frederick Solt [aut], Yue Hu [aut, cre], Os Keyes [ctb], Ben Bolker [ctb], Stefan Müller [ctb], Thomas Leeper [ctb], Chris Wallace [ctb], Christopher Warshaw [ctb]</td>
</tr>
<tr>
<td>Repository</td>
<td>CRAN</td>
</tr>
<tr>
<td>Date/Publication</td>
<td>2021-08-22 08:40:09 UTC</td>
</tr>
</tbody>
</table>
R topics documented:

- add_brackets
- by_2sd
- dwplot
- relabel_predictors
- relabel_y_axis
- secret_weapon
- small_multiple

Index

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, fontSize = 0.7, face = "italic", ...)

Arguments

- **p**
  A plot generated by dwplot. Any 'ggplot' customization should be done before passing the plot to add_brackets. To pass the finalized plot to add_brackets without creating an intermediate object, simply wrap the code that generates it in braces (`{` and `}`).

- **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 bottom most variable to be enclosed by the bracket.

- **fontSize**
  A number defining the size of the bracket label. The default value is .7.

- **face**
  A typeface for the bracket labels; options are "plain", "bold", "italic", "oblique", and "bold.italic".

- **...**
  Extra arguments to pass to gpar.

Details

The brackets are drawn by 'grid' functions. Apart from font size and typeface, users can customize the appearance of the bracket labels by setting 'gpar' arguments in 'add_brackets'.

Value

The function returns a ggplot object.
Examples

```r
library(dplyr)
m1 <- lm(mpg ~ wt + cyl + disp, data = mtcars)
two_brackets <- list(c("Engine", "Cylinder", "Displacement"),
                      c("Not Engine", "Intercept", "Weight"))

{dwplot(m1, show_intercept = TRUE) %>%
  relabel_predictors("(Intercept)" = "Intercept",
                    wt = "Weight",
                    cyl = "Cylinder",
                    disp = "Displacement") +
  theme_bw() + xlab("Coefficient") + ylab("") +
  theme(legend.position="none") +
  geom_vline(xintercept = 0, colour = "grey50", linetype = 2) %>%
  add_brackets(two_brackets)
```

by_2sd

Rescale regression results by multiplying by 2 standard deviations

by_2sd

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, or (preferably) the `model matrix` used by the models in df; the information required for complex models can more easily be generated from the model matrix than from the original data set. In many cases the model matrix can be extracted from the original model via `model.matrix`.

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 each other and 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 from 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 \texttt{arm::standardize} before passing the results to \texttt{tidy} and then on to \texttt{dwplot}. The advantages of \texttt{by_2sd} are that (1) it takes a tidy data frame as its input and so is not restricted to only those model objects that \texttt{standardize} accepts and (2) it is much more efficient because it operates on the parameters rather than refitting the original model with scaled data.

### Value

A tidy data frame

### References


### Examples

```r
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

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

### Usage

```r
dwplot(
  x,
  ci = 0.95,
  dodge_size = 0.4,
  vars_order = NULL,
  show_intercept = FALSE,
  margins = FALSE,
  model_name = "model",
  model_order = NULL,
  style = c("dotwhisker", "distribution"),
  by_2sd = FALSE,
)```

vline = NULL,
dot_args = list(size = 1.2),
whisker_args = list(size = 0.5),
dist_args = list(alpha = 0.5),
line_args = list(alpha = 0.75, size = 1),
...
)

dw_plot(
x,
ci = 0.95,
dodge_size = 0.4,
vars_order = NULL,
show_intercept = FALSE,
margins = FALSE,
model_name = "model",
model_order = NULL,
style = c("dotwhisker", "distribution"),
by_2sd = FALSE,
vline = NULL,
dot_args = list(size = 1.2),
whisker_args = list(size = 0.5),
dist_args = list(alpha = 0.5),
line_args = list(alpha = 0.75, size = 1),
...
)

Arguments

x Either a model object to be tidied with tidy, or a list of such model objects, or a tidy data frame of regression results (see 'Details').

ci A number indicating the level of confidence intervals; the default is .95.

dodge_size A number 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; the default is 0.4.

vars_order A vector of variable names that specifies the order in which the variables are to appear along the y-axis of the plot. Note that the order will be overwritten by relabel_predictors, if the function is following called.

show_intercept A logical constant indicating whether the coefficient of the intercept term should be plotted.

margins A logical value indicating whether presenting the average marginal effects of the estimates. See the Details for more information.

model_name The name of a variable that distinguishes separate models within a tidy data frame.
model_order  A character vector defining the order of the models when multiple models are involved.

style  Either "dotwhisker" or "distribution". "dotwhisker", the default, shows the regression coefficients’ point estimates as dots with confidence interval whiskers. "distribution" shows the normal distribution with mean equal to the point estimate and standard deviation equal to the standard error, underscored with a confidence interval whisker.

by_2sd  When x is model object or list of model objects, should the coefficients for predictors that are not binary be rescaled by twice the standard deviation of these variables in the dataset analyzed, per Gelman (2008)? Defaults to FALSE. Note that when x is a tidy data frame, one can use by_2sd to rescale similarly.

vline  A geom_vline() object, typically with xintercept = 0, to be drawn behind the coefficients.

dot_args  When style is "dotwhisker", a list of arguments specifying the appearance of the dots representing mean estimates. For supported arguments, see geom_point.

whisker_args  When style is "dotwhisker", a list of arguments specifying the appearance of the whiskers representing the confidence intervals. For supported arguments, see geom_linerangeh.

dist_args  When style is "distribution", a list of arguments specifying the appearance of normally distributed regression estimates. For supported arguments, see geom_polygon.

line_args  When style is "distribution", a list of arguments specifying the appearance of the line marking the confidence interval beneath the normal distribution. For supported arguments, see geom_linerangeh.

...  Extra arguments to pass to parameters.

Details

dwplot visualizes regression model objects or regression results saved in tidy data frames 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 tidy_parameters, parameters (with proper formatting)), or a list of such model objects.

By default, the plot will display 95-percent confidence intervals. To display a different interval when passing a model object or objects, specify a ci argument. When passing a data frame of results, include the variables conf_low and conf_high describing the bounds of the desired interval.

Because the function can take a data frame as input, it is easily employed for a wide range of models, including those not supported by broom, broomExtra, or parameters. 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.
dwplot provides an option to present the average marginal effect directly based on `margins`. Users can alter the confidence intervals of the margins through the `ci` argument. See the full list of supported functions in the document of the package `margins`. The ‘margins’ argument also works for `small_multiple` and `secret_weapon`.

**Value**

The function returns a ggplot object.

**References**


**Examples**

```r
library(dplyr)
# Plot regression coefficients from a single model object
data(mtcars)
m1 <- lm(mpg ~ wt + cyl + disp, data = mtcars)
dwplot(m1, vline = geom_vline(xintercept = 0, colour = "grey50", linetype = 2)) +
xlab("Coefficient")
# using 99% confidence interval
dwplot(m1, ci = .99)
# Plot regression coefficients from multiple models
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 on the fly
mtcars %>%
  split(.$am) %>%
purr::map(~ lm(mpg ~ wt + cyl + disp, data = .x)) %>%
dwplot() %>%
relabell predictors(c(wt = "Weight", cyl = "Cylinders", disp = "Displacement")) +
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.position = c(.995, .99),
  legend.justification = c(1, 1),
  legend.background = element_rect(colour="grey80"),
  legend.title.align = .5) +
scale_colour_grey(start = .4, end = .8,
  name = "Transmission",
  breaks = c("Model 0", "Model 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` or a plot generated by `dwplot`.

**Usage**

```r
relabel_predictors(x, ...)  
```

**Arguments**

- **x**  
  Either a tidy data frame to be passed to `dwplot` or a plot generated by `dwplot`.

- **...**  
  Named replacements, as in `recode`. The argument names should be the current values to be replaced, and the argument values should be the new (replacement) values. For backwards compatibility, a named character vector, with new values as values, and old values as names may also be used. The order of the named replacements will be preserved, so this function also serves the purpose of re-ordering variables.

**Value**

The function returns an object of the same type as it is passed: a tidy data frame or a plot generated by `dwplot`.

**Examples**

```r
library(broom)  
library(dplyr)  

data(mtcars)  
model <- lm(mpg ~ wt + cyl + disp, data = mtcars)  
model_df <- broom::tidy(model) %>%  
  relabel_predictors("(Intercept)" = "Intercept",  
  wt = "Weight",  
  disp = "Displacement",  
  cyl = "Cylinder")  

dwplot(model_df)

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

dwplot(m1_df, show_intercept = TRUE) %>%  
  relabel_predictors("(Intercept)" = "Intercept",  
  wt = "Weight",  
  disp = "Displacement",  
  cyl = "Cylinder")
```
**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, ci = 0.95, margins = FALSE, by_2sd = FALSE, ...)`

**Arguments**

- `x` Either a model object to be tidied with `tidy`, or a list of such model objects, or a tidy data frame of regression results (see ‘Details’).
- `var` The predictor whose results are to be shown in the ‘secret weapon’ plot
- `ci` A number indicating the level of confidence intervals; the default is .95.
- `margins` A logical value indicating whether presenting the average marginal effects of the estimates. See the Details for more information.
- `by_2sd` When x is a list of model objects, should the coefficients for predictors that are not binary be rescaled by twice the standard deviation of these variables in the dataset analyzed, per Gelman (2008)? Defaults to TRUE. Note that when x is a tidy data frame, one can use `by_2sd` to rescale similarly.
- `...` Arguments to pass to `dwplot`.
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 (or tidy_parameters, parameters (with proper formatting)), or a list of such model objects.

Value

The function returns a ggplot object.

Examples

library(dplyr)
library(broom)

# Estimate models across many samples, put results in a tidy data frame
by_clarity <- diamonds %>% group_by(clarity) %>%
  do(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,
  ci = 0.95,
  margins = FALSE,
small_multiple

dodge_size = 0.4,
show_intercept = FALSE,
model_order = NULL,
submodel_order = NULL,
axis_switch = FALSE,
by_2sd = FALSE,
dot_args = list(size = 0.3),
...
)

Arguments

x Either a model object to be tidied with tidy, or a list of such model objects, or a tidy data frame of regression results (see 'Details').

ci A number indicating the level of confidence intervals; the default is .95.
margins A logical value indicating whether presenting the average marginal effects of the estimates. See the Details for more information.
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.

show_intercept A logical constant indicating whether the coefficient of the intercept term should be plotted

model_order A character vector defining the order of the models when multiple models are involved.

submodel_order A character vector defining the order of the submodels when multiple submodels are involved.

axis_switch A logical constant indicating the position of variable labels and y axis ticks. Default is FALSE, when the variable label is on the right side, and y axis ticks is on the left size.

by_2sd When x is model object or list of model objects, should the coefficients for predictors that are not binary be rescaled by twice the standard deviation of these variables in the dataset analyzed, per Gelman (2008)? Defaults to TRUE. Note that when x is a tidy data frame, one can use by_2sd to rescale similarly.
dot_args A list of arguments specifying the appearance of the dots representing mean estimates. For supported arguments, see geom_pointrangeh.

... Arguments to pass to dwplot.

Details

small_multiple, following Kastellec and Leoni (2007), provides a compact means of representing numerous regression models in a single plot.

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 `conf.low` (the lower bounds of the confidence intervals of each estimate) and `conf.high` (the corresponding upper bounds).

Alternately, `small_multiple` accepts as input a list of model objects that can be tidied by `tidy` (or `tidy_parameters`, `parameters` with proper formatting), or a list of such model objects.

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.

**References**


**Examples**

```r
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)
```

```r
## 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) %>%
do(tidy(lm(mod, data = .))) %>%
rename(submodel = am) %>%
mutate(model = "Model 1") %>%
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.spacing = unit(-3, "pt"),
        legend.key.size = unit(10, "pt")) +
  scale_colour_hue(name = "Transmission",
                  breaks = c(0, 1),
                  labels = c("Automatic", "Manual"))
Index

add_brackets, 2
by_2sd, 3, 6, 9, 11

dw_plot (dwplot), 4
dwplot, 3, 4, 4, 8, 9, 11

tidy, 3-6, 9-12
tidy_parameters, 6, 10, 12