Regression Tables with huxreg

David Hugh-Jones

2017-04-27

Regression tables with `huxreg`

From version 0.2, huxtable includes the function `huxreg` to build a table of regressions.

`huxreg` can be called with a list of models. These models can be of any class which has a `tidy` method defined in the broom package. The method should return a list of regression coefficients with names `term`, `estimate`, `std.error` and `p.value`. That covers most standard regression packages.

Let's start by running some regressions to predict a diamond's price.

```r
data(diamonds, package = 'ggplot2')

lm1 <- lm(price ~ carat + depth, diamonds)
lm2 <- lm(price ~ depth + factor(color, ordered = FALSE), diamonds)
lm3 <- lm(log(price) ~ carat + depth, diamonds)
```

Now, we call `huxreg` to display the regression output side by side.

```r
huxreg(lm1, lm2, lm3)
```

	(1)	(2)	(3)
(Intercept)	4045.333 ***	6491.466 ***	7.313 ***
	(286.205)	(730.537)	(0.074)
carat	7765.141 ***		1.971 ***
	(14.009)		(0.004)
depth	-102.165 ***	-53.835 ***	-0.018 ***
	(4.635)	(11.815)	(0.001)
factor(color, ordered = FALSE)E		-95.142	
		(62.037)	
factor(color, ordered = FALSE)F		554.742 ***	
		(62.374)	
factor(color, ordered = FALSE)G		832.357 ***	
		(60.338)	
factor(color, ordered = FALSE)H		1324.183 ***	
		(64.296)	
factor(color, ordered = FALSE)I		1929.902 ***	
		(71.561)	
factor(color, ordered = FALSE)J		2164.044 ***	
		(88.144)	
N	53940	53940	53940
R2	0.851	0.032	0.847
logLik	-472488.441	-522908.139	-26617.649
AIC	944984.882	1045834.277	53243.298

*** p < 0.001; ** p < 0.01; * p < 0.05.

The basic output includes estimates, standard errors and summary statistics.

Some of those variable names are hard to read. We can change them by specifying a named list of variables in the `coefs` argument, like this:

```
color_names <- paste0('factor(color, ordered = FALSE)', LETTERS[5:10])
names(color_names) <- paste('Color:', LETTERS[5:10])

huxreg(lm1, lm2, lm3, coefs = c('Carat' = 'carat', 'Depth' = 'depth', color_names))
```

	(1)	(2)	(3)
Carat	7765.141 ***		1.971 ***
	(14.009)		(0.004)
Depth	-102.165 ***	-53.835 ***	-0.018 ***
	(4.635)	(11.815)	(0.001)
Color: E		-95.142	
		(62.037)	
Color: F		554.742 ***	
		(62.374)	
Color: G		832.357 ***	
		(60.338)	
Color: H		1324.183 ***	
		(64.296)	
Color: I		1929.902 ***	
		(71.561)	
Color: J		2164.044 ***	
		(88.144)	
N	53940	53940	53940
R2	0.851	0.032	0.847
logLik	-472488.441	-522908.139	-26617.649
AIC	944984.882	1045834.277	53243.298

*** p < 0.001; ** p < 0.01; * p < 0.05.

Alternatively, since the output from `huxreg` is just a huxtable, we could just edit its contents directly before we print it:

```
diamond_regs <- huxreg(lm1, lm2, lm3)
diamond_regs[seq(8, 18, 2), 1] <- paste('Color:', LETTERS[5:10])
diamond_regs
```

	(1)	(2)	(3)
(Intercept)	4045.333 ***	6491.466 ***	7.313 ***
	(286.205)	(730.537)	(0.074)
carat	7765.141 ***		1.971 ***
	(14.009)		(0.004)
depth	-102.165 ***	-53.835 ***	-0.018 ***
	(4.635)	(11.815)	(0.001)
Color: E		-95.142	
		(62.037)	
Color: F		554.742 ***	
		(62.374)	
Color: G		832.357 ***	
		(60.338)	
Color: H		1324.183 ***	
		(64.296)	
Color: I		1929.902 ***	
		(71.561)	
Color: J		2164.044 ***	
		(88.144)	
N	53940	53940	53940
R2	0.851	0.032	0.847
logLik	-472488.441	-522908.139	-26617.649
AIC	944984.882	1045834.277	53243.298

*** p < 0.001; ** p < 0.01; * p < 0.05.

Of course, we aren't limited to just changing names. We can also make our table prettier. Let's add the "article" theme, and a vertical stripe for background colour, tweak a few details like font size, and add a caption. All of these are just standard huxtable commands.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:huxtable':
##
##     add_rownames

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
diamond_regs                                                         %>%
      theme_article                                                  %>%
      set_background_color(1:nrow(diamond_regs), evens, grey(.95)) %>%
      set_font_size(final(), 1, 9)                                   %>%
      set_bold(final(), 1, FALSE)                                    %>%
      set_top_border(final(), 1, 1)                                  %>%
      set_caption('Linear regressions of diamond prices')
```

Table 1: Linear regressions of diamond prices

	(1)	**(2)**	**(3)**
(Intercept)	4045.333 ***	6491.466 ***	7.313 ***
	(286.205)	(730.537)	(0.074)
carat	7765.141 ***		1.971 ***
	(14.009)		(0.004)
depth	-102.165 ***	-53.835 ***	-0.018 ***
	(4.635)	(11.815)	(0.001)
Color: E		-95.142	
		(62.037)	
Color: F		554.742 ***	
		(62.374)	
Color: G		832.357 ***	
		(60.338)	
Color: H		1324.183 ***	
		(64.296)	
Color: I		1929.902 ***	
		(71.561)	
Color: J		2164.044 ***	
		(88.144)	
N	53940	53940	53940
R2	0.851	0.032	0.847
logLik	-472488.441	-522908.139	-26617.649
AIC	944984.882	1045834.277	53243.298

*** p < 0.001; ** p < 0.01; * p < 0.05.

We could do more, like changing the `number_format` of N to not display decimals. But let's explore what else `huxreg` itself can do.

By default, standard errors are shown below coefficient estimates. To display them in a column to the right, use `error_pos = 'right'`:

```r
huxreg(lm1, lm3, error_pos = 'right')
```

	(1)		(2)	
(Intercept)	4045.333 ***	(286.205)	7.313 ***	(0.074)
carat	7765.141 ***	(14.009)	1.971 ***	(0.004)
depth	-102.165 ***	(4.635)	-0.018 ***	(0.001)
N	53940		53940	
R2	0.851		0.847	
logLik	-472488.441		-26617.649	
AIC	944984.882		53243.298	

*** p < 0.001; ** p < 0.01; * p < 0.05.

This will give column headings a column span of 2.

To display standard errors in the same cell as estimates, use `error_pos = 'same'`:

```r
huxreg(lm1, lm3, error_pos = 'same')
```

	(1)	(2)
(Intercept)	4045.333 *** (286.205)	7.313 *** (0.074)
carat	7765.141 *** (14.009)	1.971 *** (0.004)
depth	-102.165 *** (4.635)	-0.018 *** (0.001)
N	53940	53940
R2	0.851	0.847
logLik	-472488.441	-26617.649
AIC	944984.882	53243.298

*** p < 0.001; ** p < 0.01; * p < 0.05.

You can change the default column headings by giving names to your models:

```r
huxreg('Price' = lm1, 'Log price' = lm3)
```

	Price	Log price
(Intercept)	4045.333 ***	7.313 ***
	(286.205)	(0.074)
carat	7765.141 ***	1.971 ***
	(14.009)	(0.004)
depth	-102.165 ***	-0.018 ***
	(4.635)	(0.001)
N	53940	53940
R2	0.851	0.847
logLik	-472488.441	-26617.649
AIC	944984.882	53243.298

*** p < 0.001; ** p < 0.01; * p < 0.05.

To display a particular row of summary statistics, use the `statistics` parameter. This should be a character vector. Valid values are anything returned from your models by `broom::glance`. Another valid value is `"nobs"`, which returns the number of observations from the regression. If the `statistics` vector has names, these will be used for row headings:

```r
broom::glance(lm1)
```

```
##   r.squared adj.r.squared    sigma statistic p.value df    logLik      AIC
## 1 0.8506755     0.8506699 1541.649  153634.8       0  3 -472488.4 944984.9
##         BIC     deviance df.residual
## 1 945020.5 128191108498       53937
```

```r
huxreg(lm1, lm3, statistics = c('# observations' = 'nobs', 'R squared' = 'r.squared', 'F statistic' = '
  'P value' = 'p.value'))
```

	(1)	(2)
(Intercept)	4045.333 ***	7.313 ***
	(286.205)	(0.074)
carat	7765.141 ***	1.971 ***
	(14.009)	(0.004)
depth	-102.165 ***	-0.018 ***
	(4.635)	(0.001)
# observations	53940	53940
R squared	0.851	0.847
F statistic	153634.765	149771.327
P value	0.000	0.000

*** p < 0.001; ** p < 0.01; * p < 0.05.

You aren't limited to displaying standard errors of the estimates. If you prefer, you can display t statistics or p values, using the `error_style` option:

```r
huxreg(lm1, lm3, error_style = 'statistic')
```

	(1)	(2)
(Intercept)	4045.333 ***	7.313 ***
	(14.134)	(99.383)
carat	7765.141 ***	1.971 ***
	(554.282)	(547.305)
depth	-102.165 ***	-0.018 ***
	(-22.041)	(-14.936)
N	53940	53940
R2	0.851	0.847
logLik	-472488.441	-26617.649
AIC	944984.882	53243.298

*** p < 0.001; ** p < 0.01; * p < 0.05.

```r
huxreg(lm1, lm3, error_style = 'pvalue')
```

	(1)	(2)
(Intercept)	4045.333 ***	7.313 ***
	(0.000)	(0.000)
carat	7765.141 ***	1.971 ***
	(0.000)	(0.000)
depth	-102.165 ***	-0.018 ***
	(0.000)	(0.000)
N	53940	53940
R2	0.851	0.847
logLik	-472488.441	-26617.649
AIC	944984.882	53243.298

*** p < 0.001; ** p < 0.01; * p < 0.05.

Or you can display confidence intervals using 'ci'. Use `ci_level` to set the confidence level for the interval:

```r
huxreg(lm1, lm3, error_style = 'ci') # default is .95
```

```r
huxreg(lm1, lm3, error_style = 'ci', ci_level = .99)
```

	(1)	(2)
(Intercept)	4045.333 ***	7.313 ***
	(3484.381 – 4606.285)	(7.169 – 7.457)
carat	7765.141 ***	1.971 ***
	(7737.683 – 7792.599)	(1.964 – 1.978)
depth	-102.165 ***	-0.018 ***
	(-111.250 – -93.080)	(-0.020 – -0.015)
N	53940	53940
R2	0.851	0.847
logLik	-472488.441	-26617.649
AIC	944984.882	53243.298

*** p < 0.001; ** p < 0.01; * p < 0.05.

	(1)	(2)
(Intercept)	4045.333 ***	7.313 ***
	(3308.117 – 4782.549)	(7.123 – 7.502)
carat	7765.141 ***	1.971 ***
	(7729.055 – 7801.226)	(1.962 – 1.981)
depth	-102.165 ***	-0.018 ***
	(-114.105 – -90.226)	(-0.021 – -0.015)
N	53940	53940
R2	0.851	0.847
logLik	-472488.441	-26617.649
AIC	944984.882	53243.298

*** p < 0.001; ** p < 0.01; * p < 0.05.

If you choose more than one `error_style` option, the second one will be shown in square brackets:

```r
huxreg(lm1, lm3, error_style = c('stderr', 'ci'))
```

	(1)	(2)
(Intercept)	4045.333 ***	7.313 ***
	(286.205) [3484.381 – 4606.285]	(0.074) [7.169 – 7.457]
carat	7765.141 ***	1.971 ***
	(14.009) [7737.683 – 7792.599]	(0.004) [1.964 – 1.978]
depth	-102.165 ***	-0.018 ***
	(4.635) [-111.250 – -93.080]	(0.001) [-0.020 – -0.015]
N	53940	53940
R2	0.851	0.847
logLik	-472488.441	-26617.649
AIC	944984.882	53243.298

*** p < 0.001; ** p < 0.01; * p < 0.05.

To change the footnote, use `note`. If `note` contains the string `"%stars%"` it will be replaced by a description of the significance stars used. If you don't want a footnote, just set `note = NULL`.

```r
huxreg(lm1, lm3, note = 'Linear regressions on diamond price. %stars%.')
```

	(1)	(2)
(Intercept)	4045.333 ***	7.313 ***
	(286.205)	(0.074)
carat	7765.141 ***	1.971 ***
	(14.009)	(0.004)
depth	-102.165 ***	-0.018 ***
	(4.635)	(0.001)
N	53940	53940
R2	0.851	0.847
logLik	-472488.441	-26617.649
AIC	944984.882	53243.298

Linear regressions on diamond price. *** p < 0.001; ** p < 0.01; * p < 0.05.

To change number formatting, set the `number_format` parameter. This works the same as the `number_format` property for a huxtable - if it is numeric, numbers will be rounded to that many decimal places; if it is character, it will be taken as a format to the base R `sprintf` function; if it is a function, the function will be called to format the number. `huxreg` tries to be smart and to format summary statistics like `nobs` as integers.

```
huxreg(lm1, lm3, number_format = 2)
```

	(1)	(2)
(Intercept)	4045.33 ***	7.31 ***
	(286.21)	(0.07)
carat	7765.14 ***	1.97 ***
	(14.01)	(0.00)
depth	-102.17 ***	-0.02 ***
	(4.64)	(0.00)
N	53940	53940
R2	0.85	0.85
logLik	-472488.44	-26617.65
AIC	944984.88	53243.30

*** p < 0.001; ** p < 0.01; * p < 0.05.

Lastly, if you want to bold all significant coefficients, set the parameter `bold_signif` to a maximum significance level:

```
huxreg(lm1, lm3, bold_signif = 0.05)
```

	(1)	(2)
(Intercept)	**4045.333** ***	**7.313** ***
	(286.205)	**(0.074)**
carat	**7765.141** ***	**1.971** ***
	(14.009)	**(0.004)**
depth	**-102.165** ***	**-0.018** ***
	(4.635)	**(0.001)**
N	53940	53940
R2	0.851	0.847
logLik	-472488.441	-26617.649
AIC	944984.882	53243.298

*** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.