Regression Tables with huxreg

David Hugh-Jones

2023-12-09

Regression tables with huxreg

Huxtable includes the function `huxreg` to build a table of regressions.

You call `huxreg` 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.

```
data(diamonds, package = "ggplot2")
diamonds <- diamonds[1:100,]

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

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

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

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

Some of those variable names are hard to read. We can change them by providing a named vector of variables in the `coefs` argument.

```
color_names <- grep("factor", names(coef(lm2)), value = TRUE)
names(color_names) <- gsub(".*)(.)", "Color: \\1", color_names)

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

Or, since the output from `huxreg` is just a huxtable, we could just edit its contents directly.

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

# prints the same as above
```

Of course, we aren't limited to just changing names. We can also make our table prettier. Let's put our footnote in italic, add a caption, and highlight the cell background of significant coefficients. All of these are just standard huxtable commands.

```r
suppressPackageStartupMessages(library(dplyr))

diamond_regs %>%
      map_background_color(-1, -1, by_regex(
        "\\*" = "yellow"
      )) %>%
      set_italic(final(1), 1) %>%
      set_caption("Linear regressions of diamond prices")
```

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")
```

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")
```

You can change the default column headings by naming the model arguments:

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

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`:

```r
gl <- as_hux(broom::glance(lm1))

gl %>%
      restack_down(cols = 3, on_remainder = "fill") %>%
      set_bold(odds, everywhere)
```

Another value you can use 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
huxreg(lm1, lm3, statistics = c("N. obs." = "nobs",
      "R squared" = "r.squared", "F statistic" = "statistic",
      "P value" = "p.value"))
```

By default, `huxreg` displays significance stars. You can alter the symbols used and significance levels with the `stars` parameter, or set `stars = NULL` to turn off significance stars completely.

```r
huxreg(lm1, lm3, stars = c(`*` = 0.1, `**` = 0.05, `***` = 0.01)) # a little boastful?
```

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_format` option. Any column from `tidy` can be used by putting it in curly brackets:

```r
# Another useful column: p.value
huxreg(
        lm1, lm3,
        error_format = "[{statistic}]",
        note         = "{stars}. T statistics in brackets."
      )
```

Here we also changed the footnote, using `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`.

Alternatively, you can display confidence intervals. Use `ci_level` to set the confidence level for the interval, then use `{conf.low}` and `{conf.high}` in `error_format`:

```r
huxreg(lm1, lm3, ci_level = .99, error_format = "({conf.low} -- {conf.high})")
```

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. `huxreg` tries to be smart and to format summary statistics like `nobs` as integers.

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

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

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

Altering data

Sometimes, you want to report different statistics for a model. For example, you might want to use robust standard errors.

One way to do this is to pass a `tidy`-able test object into `huxreg`. The function `coeftest` in the "lmtest" package has `tidy` methods defined:

```r
library(lmtest)
library(sandwich)
lm_robust <- coeftest(lm1, vcov = vcovHC, save = TRUE)
huxreg("Normal SEs" = lm1, "Robust SEs" = lm_robust)
```

If that is not possible, you can compute statistics yourself and add them to your model using the `tidy_override` function:

```r
lm_fixed <- tidy_override(lm1, p.value = c(0.5, 0.2, 0.06))
huxreg("Normal p values" = lm1, "Supplied p values" = lm_fixed)
```

You can override any statistics returned by `tidy` or `glance`.

If you want to completely replace the output of tidy, use the `tidy_replace()` function. For example, here's how to print different coefficients for a multinomial model.

```r
mnl <- nnet::multinom(gear ~ mpg, mtcars)
```

```
## # weights:  9 (4 variable)
## initial  value 35.155593
## iter  10 value 23.131901
## final  value 23.129234
## converged
```

```r
tidied <- broom::tidy(mnl)
models <- list()
models[["4 gears"]] <- tidy_replace(mnl, tidied[tidied$y.level == 4, ])
models[["5 gears"]] <- tidy_replace(mnl, tidied[tidied$y.level == 5, ])
huxreg(models, statistics = "AIC")
```

	(1)	(2)	(3)
(Intercept)	981.607	900.067	6.269 ***
	(720.175)	(2431.815)	(0.782)
carat	4328.324 ***		3.531 ***
	(136.755)		(0.149)
depth	-27.785 *	-6.804	-0.019
	(11.656)	(39.293)	(0.013)
factor(color, ordered = FALSE)E		449.490	
		(239.388)	
factor(color, ordered = FALSE)F		391.705	
		(290.880)	
factor(color, ordered = FALSE)G		583.111	
		(308.513)	
factor(color, ordered = FALSE)H		126.916	
		(256.367)	
factor(color, ordered = FALSE)I		-47.220	
		(253.092)	
factor(color, ordered = FALSE)J		-123.430	
		(269.157)	
N	100	100	100
R2	0.912	0.123	0.854
logLik	-675.703	-790.788	6.822
AIC	1359.405	1599.576	-5.644

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

	(1)	(2)	(3)
Carat	4328.324 ***		3.531 ***
	(136.755)		(0.149)
Depth	-27.785 *	-6.804	-0.019
	(11.656)	(39.293)	(0.013)
Color: E		449.490	
		(239.388)	
Color: F		391.705	
		(290.880)	
Color: G		583.111	
		(308.513)	
Color: H		126.916	
		(256.367)	
Color: I		-47.220	
		(253.092)	
Color: J		-123.430	
		(269.157)	
N	100	100	100
R2	0.912	0.123	0.854
logLik	-675.703	-790.788	6.822
AIC	1359.405	1599.576	-5.644

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

Table 1: Linear regressions of diamond prices

	(1)	(2)	(3)
(Intercept)	981.607	900.067	6.269 ***
	(720.175)	(2431.815)	(0.782)
carat	4328.324 ***		3.531 ***
	(136.755)		(0.149)
depth	-27.785 *	-6.804	-0.019
	(11.656)	(39.293)	(0.013)
Color: E		449.490	
		(239.388)	
Color: F		391.705	
		(290.880)	
Color: G		583.111	
		(308.513)	
Color: H		126.916	
		(256.367)	
Color: I		-47.220	
		(253.092)	
Color: J		-123.430	
		(269.157)	
N	100	100	100
R2	0.912	0.123	0.854
logLik	-675.703	-790.788	6.822
AIC	1359.405	1599.576	-5.644

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

	(1)		(2)	
(Intercept)	981.607	(720.175)	6.269 ***	(0.782)
carat	4328.324 ***	(136.755)	3.531 ***	(0.149)
depth	-27.785 *	(11.656)	-0.019	(0.013)
N	100		100	
R2	0.912		0.854	
logLik	-675.703		6.822	
AIC	1359.405		-5.644	

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

	(1)	(2)
(Intercept)	981.607 (720.175)	6.269 *** (0.782)
carat	4328.324 *** (136.755)	3.531 *** (0.149)
depth	-27.785 * (11.656)	-0.019 (0.013)
N	100	100
R2	0.912	0.854
logLik	-675.703	6.822
AIC	1359.405	-5.644

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

	Price	Log price
(Intercept)	981.607	6.269 ***
	(720.175)	(0.782)
carat	4328.324 ***	3.531 ***
	(136.755)	(0.149)
depth	-27.785 *	-0.019
	(11.656)	(0.013)
N	100	100
R2	0.912	0.854
logLik	-675.703	6.822
AIC	1359.405	-5.644

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

r.squared	adj.r.squared	sigma
0.912	0.91	211
statistic	**p.value**	**df**
504	5.65e-52	2
logLik	**AIC**	**BIC**
-676	1.36e+03	1.37e+03
deviance	**df.residual**	**nobs**
4.33e+06	97	100

	(1)	(2)
(Intercept)	981.607	6.269 ***
	(720.175)	(0.782)
carat	4328.324 ***	3.531 ***
	(136.755)	(0.149)
depth	-27.785 *	-0.019
	(11.656)	(0.013)
N. obs.	100	100
R squared	0.912	0.854
F statistic	504.082	283.881
P value	0.000	0.000

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

	(1)	(2)
(Intercept)	981.607	6.269 ***
	(720.175)	(0.782)
carat	4328.324 ***	3.531 ***
	(136.755)	(0.149)
depth	-27.785 **	-0.019
	(11.656)	(0.013)
N	100	100
R2	0.912	0.854
logLik	-675.703	6.822
AIC	1359.405	-5.644

*** $p < 0.01$; ** $p < 0.05$; * $p < 0.1$.

	(1)	(2)
(Intercept)	981.607	6.269 ***
	[1.363]	[8.016]
carat	4328.324 ***	3.531 ***
	[31.650]	[23.773]
depth	-27.785 *	-0.019
	[-2.384]	[-1.499]
N	100	100
R2	0.912	0.854
logLik	-675.703	6.822
AIC	1359.405	-5.644

*** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$. T statistics in brackets.

	(1)	(2)
(Intercept)	981.607	6.269 ***
	(-910.629 – 2873.844)	(4.214 – 8.324)
carat	4328.324 ***	3.531 ***
	(3969.004 – 4687.643)	(3.140 – 3.921)
depth	-27.785 *	-0.019
	(-58.411 – 2.842)	(-0.052 – 0.014)
N	100	100
R2	0.912	0.854
logLik	-675.703	6.822
AIC	1359.405	-5.644

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

	(1)	(2)
(Intercept)	981.61	6.27 ***
	(720.17)	(0.78)
carat	4328.32 ***	3.53 ***
	(136.75)	(0.15)
depth	-27.78 *	-0.02
	(11.66)	(0.01)
N	100	100
R2	0.91	0.85
logLik	-675.70	6.82
AIC	1359.41	-5.64

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

	(1)	(2)
(Intercept)	981.607	**6.269 *** **
	(720.175)	**(0.782)**
carat	**4328.324 *** **	**3.531 *** **
	(136.755)	**(0.149)**
depth	**-27.785 * **	-0.019
	(11.656)	(0.013)
N	100	100
R2	0.912	0.854
logLik	-675.703	6.822
AIC	1359.405	-5.644

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

	Normal SEs	Robust SEs
(Intercept)	981.607	981.607
	(720.175)	(1117.654)
carat	4328.324 ***	4328.324 ***
	(136.755)	(293.929)
depth	-27.785 *	-27.785
	(11.656)	(17.995)
N	100	100
R2	0.912	0.912
logLik	-675.703	-675.703
AIC	1359.405	1359.405

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

	Normal p values	Supplied p values
(Intercept)	981.607	981.607
	(720.175)	(720.175)
carat	4328.324 ***	4328.324
	(136.755)	(136.755)
depth	-27.785 *	-27.785
	(11.656)	(11.656)
N	100	100
R2	0.912	0.912
logLik	-675.703	-675.703
AIC	1359.405	1359.405

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

	4 gears	5 gears
(Intercept)	-9.502 **	-7.691 *
	(3.262)	(3.232)
mpg	0.475 **	0.358 *
	(0.168)	(0.168)
AIC	54.258	54.258

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