

# Regression Tables with huxreg

David Hugh-Jones

2025-08-17

## 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)
```

|                                 | (1)                       | (2)                   | (3)                  |
|---------------------------------|---------------------------|-----------------------|----------------------|
| (Intercept)                     | 981.607<br>(720.175)      | 900.067<br>(2431.815) | 6.269 ***<br>(0.782) |
| carat                           | 4328.324 ***<br>(136.755) |                       | 3.531 ***<br>(0.149) |
| depth                           | -27.785 *<br>(11.656)     | -6.804<br>(39.293)    | -0.019<br>(0.013)    |
| factor(color, ordered = FALSE)E |                           | 449.490<br>(239.388)  |                      |
| factor(color, ordered = FALSE)F |                           | 391.705<br>(290.880)  |                      |
| factor(color, ordered = FALSE)G |                           | 583.111<br>(308.513)  |                      |
| factor(color, ordered = FALSE)H |                           | 126.916<br>(256.367)  |                      |
| factor(color, ordered = FALSE)I |                           | -47.220<br>(253.092)  |                      |
| factor(color, ordered = FALSE)J |                           | -123.430<br>(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.

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))
```

|          | (1)                       | (2)                   | (3)                  |
|----------|---------------------------|-----------------------|----------------------|
| Carat    | 4328.324 ***<br>(136.755) |                       | 3.531 ***<br>(0.149) |
| Depth    | -27.785 *<br>(11.656)     | -6.804<br>(39.293)    | -0.019<br>(0.013)    |
| Color: E |                           | 449.490<br>(239.388)  |                      |
| Color: F |                           | 391.705<br>(290.880)  |                      |
| Color: G |                           | 583.111<br>(308.513)  |                      |
| Color: H |                           | 126.916<br>(256.367)  |                      |
| Color: I |                           | -47.220<br>(253.092)  |                      |
| Color: J |                           | -123.430<br>(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.

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.

```
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")
```

Table 1: Linear regressions of diamond prices

|             | (1)                       | (2)                   | (3)                  |
|-------------|---------------------------|-----------------------|----------------------|
| (Intercept) | 981.607<br>(720.175)      | 900.067<br>(2431.815) | 6.269 ***<br>(0.782) |
| carat       | 4328.324 ***<br>(136.755) |                       | 3.531 ***<br>(0.149) |
| depth       | -27.785 *<br>(11.656)     | -6.804<br>(39.293)    | -0.019<br>(0.013)    |
| Color: E    |                           | 449.490<br>(239.388)  |                      |
| Color: F    |                           | 391.705<br>(290.880)  |                      |
| Color: G    |                           | 583.111<br>(308.513)  |                      |
| Color: H    |                           | 126.916<br>(256.367)  |                      |
| Color: I    |                           | -47.220<br>(253.092)  |                      |
| Color: J    |                           | -123.430<br>(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               |

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

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

```
huxreg(lm1, lm3, error_pos = "right")
```

|             | (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.

This will give column headings a column span of 2.

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

```
huxreg(lm1, lm3, error_pos = "same")
```

|             | (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.

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

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

|             | Price                     | Log price            |
|-------------|---------------------------|----------------------|
| (Intercept) | 981.607<br>(720.175)      | 6.269 ***<br>(0.782) |
| carat       | 4328.324 ***<br>(136.755) | 3.531 ***<br>(0.149) |
| depth       | -27.785 *<br>(11.656)     | -0.019<br>(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.

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`:

```
gl <- as_hux(broom::glance(lm1))
gl |>
  restack_down(cols = 3, on_remainder = "fill") |>
  set_bold(odds, everywhere)
```

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

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:

```
huxreg(lm1, lm3, statistics = c("N. obs." = "nobs",
  "R_squared" = "r.squared", "F statistic" = "statistic",
  "P value" = "p.value"))
```

|             | (1)                       | (2)                  |
|-------------|---------------------------|----------------------|
| (Intercept) | 981.607<br>(720.175)      | 6.269 ***<br>(0.782) |
| carat       | 4328.324 ***<br>(136.755) | 3.531 ***<br>(0.149) |
| depth       | -27.785 *<br>(11.656)     | -0.019<br>(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.

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.

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



|             | (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 < 0.1; \*\* p < 0.05; \*\*\* p < 0.01.

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:

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

|             | (1)                      | (2)                   |
|-------------|--------------------------|-----------------------|
| (Intercept) | 981.607<br>[1.363]       | 6.269 ***<br>[8.016]  |
| carat       | 4328.324 ***<br>[31.650] | 3.531 ***<br>[23.773] |
| depth       | -27.785 *<br>[-2.384]    | -0.019<br>[-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.

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`:

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

|             | (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.

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.

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

|             | (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.

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) | 981.607             | <b>6.269 ***</b> |
|             | (720.175)           | <b>(0.782)</b>   |
| carat       | <b>4328.324 ***</b> | <b>3.531 ***</b> |
|             | <b>(136.755)</b>    | <b>(0.149)</b>   |
| depth       | <b>-27.785 *</b>    | -0.019           |
|             | <b>(11.656)</b>     | (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.

## 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:

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

|             | Normal SEs                | Robust SEs                |
|-------------|---------------------------|---------------------------|
| (Intercept) | 981.607<br>(720.175)      | 981.607<br>(1117.654)     |
| carat       | 4328.324 ***<br>(136.755) | 4328.324 ***<br>(293.929) |
| depth       | -27.785 *<br>(11.656)     | -27.785<br>(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.

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

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

|             | 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.

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.

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

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

```
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")
```

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

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