

Week 11 (ha) R tutorial

Aaron Shaw

November 24, 2020

Contents

Leftovers!	1
Import data for the examples	1
Transformations	1
Interaction terms	2
Polynomial (square, cube, etc.) terms	2
Log-transformations	4
Interpreting regression results with model-predicted values	4

Leftovers!

There is a whole lot we will not have a chance to cover in this course related to regression analysis and interpretation, but I wanted to write out a brief R tutorial on some topics that are just beyond the reach of the material introduced by our textbook and problem sets. If you have any interest/need to apply these kinds of techniques in your final projects, I hope these examples will help and give you a sense of where/how to direct your questions. As always, please get in touch when you run into trouble.

Import data for the examples

For all of the examples this week, I'll work with the `population.tsv` dataset from back in Week 6. The following lines of code load it and clean it up a bit. Since we've worked with this dataset before I will not revisit the process of "getting to know" it.

```
d <- read.delim(url("https://communitydata.science/~ads/teaching/2020/stats/data/week_06/population.tsv"))
d$j <- as.logical(d$j)
d$l <- as.logical(d$l)
d$k <- factor(d$k,
              levels=c(1,2,3),
              labels=c("some", "lots", "all"))
d <- d[complete.cases(d),]
```

Transformations

It is often necessary to transform variables for modeling (we'll discuss some reasons why in class). Here are some common transformations with example code to perform them in R. Keep in mind that you also know some others (e.g., you know how to standardize a variable or take the square root).

Interaction terms

Interaction terms are best handled using the `I()` function (note the capitalization). I start with a “base” model and update it to add the interaction.

```
m.base <- formula(y ~ x + j)
summary(lm(m.base, data=d))

##
## Call:
## lm(formula = m.base, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5.2058 -2.5528 -0.0673  2.4728  5.1225
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.09192    0.12594   0.730   0.466
## x            2.97879    0.03058  97.420 <2e-16 ***
## jTRUE        0.10907    0.13753   0.793   0.428
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.995 on 1897 degrees of freedom
## Multiple R-squared:  0.8335, Adjusted R-squared:  0.8333
## F-statistic: 4748 on 2 and 1897 DF, p-value: < 2.2e-16

m.i <- update.formula(m.base, . ~ . + I(x*j))
summary(lm(m.i, data=d))

##
## Call:
## lm(formula = m.i, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5.2662 -2.5353 -0.0366  2.4658  4.9981
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.03985    0.14590   0.273   0.785
## x            2.99836    0.04124  72.713 <2e-16 ***
## jTRUE        0.22157    0.21026   1.054   0.292
## I(x * j)     -0.04348    0.06147  -0.707   0.479
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.996 on 1896 degrees of freedom
## Multiple R-squared:  0.8335, Adjusted R-squared:  0.8333
## F-statistic: 3164 on 3 and 1896 DF, p-value: < 2.2e-16
```

Polynomial (square, cube, etc.) terms

Polynomial terms can easily be created using `I()` as well:

```
m.poly <- update.formula(m.base, . ~ . + I(x^2))
summary(lm(m.poly, data=d))
```

```
##
## Call:
## lm(formula = m.poly, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5.1795 -2.5618 -0.0642  2.4480  5.2871
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.06051    0.14814   0.408   0.683
## x            3.01171    0.08725  34.519 <2e-16 ***
## jTRUE        0.10684    0.13767   0.776   0.438
## I(x^2)       -0.00449    0.01114  -0.403   0.687
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.996 on 1896 degrees of freedom
## Multiple R-squared:  0.8335, Adjusted R-squared:  0.8332
## F-statistic: 3164 on 3 and 1896 DF, p-value: < 2.2e-16
```

Need higher order polynomials? Try including $I(x^3)$ and so on...

Creating polynomials this way is intuitive, but can also create a little bit of a messy situation for reasons that go beyond the scope of our course. In these circumstances, using the `poly()` function is useful (look up “orthogonalized polynomials” online to learn more). Generally speaking, creating polynomials in this way impacts the interpretation as well as the model estimates, so you should only use it if you need to and once you’ve taken the time to actually learn what is happening. That said, here’s what the code could look like:

```
m.poly2 <- formula(y ~ j + poly(x,2))
summary(lm(m.poly2, data=d))
```

```
##
## Call:
## lm(formula = m.poly2, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5.1795 -2.5618 -0.0642  2.4480  5.2871
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.8241    0.0962  81.329 <2e-16 ***
## jTRUE         0.1068    0.1377   0.776   0.438
## poly(x, 2)1  291.9278    2.9973  97.398 <2e-16 ***
## poly(x, 2)2  -1.2080    2.9983  -0.403   0.687
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.996 on 1896 degrees of freedom
## Multiple R-squared:  0.8335, Adjusted R-squared:  0.8332
## F-statistic: 3164 on 3 and 1896 DF, p-value: < 2.2e-16
```

Higher order (to the n th degree) terms can be created by using higher values of n as an argument to (e.g. `poly(x, n)`).

Log-transformations

We covered log transformations (usually natural logarithms) before, but just in case, here they are again. I usually default to using `log1p()` because it is less prone to fail in the event your data (like mine) contains many zeroes. That said, if you have a lot of -1 values you may need something else:

```
m.log <- update.formula(m.base, . ~ log1p(x) + j)
summary(lm(m.log, data=d))

##
## Call:
## lm(formula = m.log, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7.120 -2.813 -0.008  2.489 14.603
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.74890    0.18217  -15.090  <2e-16 ***
## log1p(x)      9.85486    0.12838   76.766  <2e-16 ***
## jTRUE        -0.02979    0.16624   -0.179    0.858
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.621 on 1897 degrees of freedom
## Multiple R-squared:  0.7566, Adjusted R-squared:  0.7563
## F-statistic: 2948 on 2 and 1897 DF, p-value: < 2.2e-16
```

Keep in mind that you can use other bases for your logarithmic transformations. Check out the documentation for `log()` for more information.

Interpreting regression results with model-predicted values

When you report the results of a regression model, you should provide a table summarizing the model as well as some interpretation that renders the model results back into the original, human-intelligible measures and units specific to the study.

This was covered in one of the resources I distributed last week (the handout on logistic regression from Mako Hill), but I wanted to bring it back because it is **important**. Please revisit that handout to see a worked example that walks through the process. The rest of this text is a bit of a rant about why you should bother to do so.

When is a regression table not enough? In textbook/homework examples, this is not an issue, but in real data it matters all the time. Recall that the coefficient estimated for any single predictor is the expected change in the outcome for a 1-unit change in the predictor *holding all the other predictors constant*. What value are those other predictors held constant at? Zero! This is unlikely to be the most helpful or intuitive way to understand your estimates (for example, what if you have a dichotomous predictor, what does it mean then?). Once your models get even a little bit complicated (quick, exponentiate a log-transformed value and tell me what it means!), the regression-table-alone approach becomes arguably worse than useless.

What is to be done? Provide predicted estimates for real, reasonable examples drawn from your dataset! For instance, if you were regressing lifetime earnings on a bunch of different predictors including years of education, gender, race, age, and height, you would, of course, start by showing your readers the table that includes

all of the coefficients, standard errors, etc. Then you should also provide some specific predicted values for “prototypical” individuals in your dataset. Regression models usually incorporate earnings as a square-root or log-transformed measure, so the table of raw model results won’t be easy to interpret. It is probably far more helpful to translate the model results into estimates of how much more/less you would estimate 30 year old woman of average height with a college degree to change if they were white vs. asian/pacific islander. These prototypical predicted values (also sometimes referred to as “marginal effects”) may be presented as specific point-estimates in the text and/or using a visualization of some sort (e.g., lines plotting the predicted lifetime earnings by race over the observed range of age...). You can (and should!) even generate confidence intervals around them (But that’s a whole separate rant...).

The point that I hope you take away is that just because you produced a regression table with some p-values and stars in it, your job is not done. You should always do the work to convey your results in a human-intelligible manner by translating the model back into some model-predicted estimates for reasonable combinations of your predictor variables. Once you’ve got the hang of that, you should also work on conveying the uncertainty/confidence around your predictions given the data/model.