Inferential I: Correlations, t-tests, and weights

EDH7916

Benjamin Skinner

After your data have been wrangled from raw values to an analysis data set and you've explored it with summary statistics and graphics, you are ready to model it and begin making inferences. As one should expect from a statistical language, R has a powerful system for fitting statistical and econometric models.

Because the purpose of this course is to learn and practice using a good quantitative work flow, we won't spend time much time interpreting the results from our inferential models. Instead, this lesson will give you the basic understanding you need to run correlations and t-tests in R as well as give a quick overview of how to use survey weights. (We'll cover regressions and predictions in a later lesson.) That said, the lessons you've learned in other courses about proper statistical practice still stand — this lesson should help you apply what you've learned before.

Data

In this lesson, we'll use data from the NCES Education Longitudinal Study of 2002[1]. Much like HSLS, ELS is a nationally representative survey that initially surveyed students in their early high school career (10th grade in 2002) and followed them into college and the workforce. We'll again use a smaller version of the data, so be sure to get the full data files if you decide to use ELS in a future project. One additional benefit of ELS is that the public use files contain the weights we'll use to properly account for the survey design later in the lesson.

Here's a codebook with descriptions of the variables included in our lesson today:

variable	description
stu_id	student id
sch_id	school id
strat_id	stratum
psu	primary sampling unit
bystuwt	student weight
bysex	sex-composite
byrace	student's race/ethnicity-composite
bydob_p	student's year and month of birth
bypared	parents' highest level of education
bymothed	mother's highest level of education-composite
byfathed	father's highest level of education-composite
byincome	total family income from all sources 2001-composite
byses1	socio-economic status composite, v.1
byses2	socio-economic status composite, v.2
bystexp	how far in school student thinks will get-composite
bynels2m	els-nels 1992 scale equated sophomore math score
bynels2r	els-nels 1992 scale equated sophomore reading score

[1] https://nces.ed.gov/surveys/els2002/

variable	description
f1qwt	questionnaire weight for f1
f1pnlwt	panel weight, by and f1 (2002 and 2004)
f1psepln	f1 post-secondary plans right after high school
f2ps1sec	Sector of first postsecondary institution
female	== 1 if female
moth_ba	== 1 if mother has BA/BS
fath_ba	== 1 if father has BA/BS
par_ba	== 1 if either parent has BA/BS
plan_col_grad	== 1 if student plans to earn college degree
lowinc	== 1 if income < $25k

Let's load the libraries and data!

...but first! You'll most likely need to install the `survey` package[2] first, using `install.packages("survey")`.

```
## ----------------------------
## libraries
## ----------------------------

library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.1     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.2     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(haven)
library(survey)
```

```
## Error in library(survey): there is no package called 'survey'
```

```
## ----------------------------
## directory paths
## ----------------------------

## assume we're running this script from the ./scripts subdirectory
dat_dir <- file.path("..", "data")
```

```
## ----------------------------
## input data
## ----------------------------

## assume we're running this script from the ./scripts subdirectory
df <- read_dta(file.path(dat_dir, "els_plans.dta"))
```

[2] https://CRAN.R-project.org/package=survey

Correlations

In prior lessons, we've visually checked for correlations between variables through plotting. We can also produce more formal tests of correlation using R's `cor()` function. By default, it gives us the Pearson correlation coefficient[3], though we can also use it to return other versions.

First, let's check the correlation between math and reading scores. Because this is a base R function, we'll need to use `df$` notation. We'll also need to be explicit about removing missing values with the argument, `use = "complete.obs"` (one of the few annoying function options that doesn't really match how other functions work).

```
## correlation between math and reading scores
cor(df$bynels2m, df$bynels2r, use = "complete.obs")
```

```
## [1] 0.7521538
```

As we might have hypothesized, there is a strong positive correlation between math and reading scores.

We can also build a correlation matrix if we give `cor()` a data frame. Our data set, though smaller than the full ELS data set, is still rather large. We'll only check the correlations between a few variables. In this next example, I'll show you how you might do this using the tidyverse way we've used in other lessons.

```
## correlation between various columns, using pipes
df %>%
    ## select a few variables
    select(byses1, bynels2m, bynels2r, par_ba) %>%
    ## use a . to be the placeholder for the piped in data.frame
    cor(., use = "complete.obs")
```

```
##             byses1  bynels2m  bynels2r    par_ba
## byses1   1.0000000 0.4338441 0.4315718 0.7041740
## bynels2m 0.4338441 1.0000000 0.7490148 0.2980608
## bynels2r 0.4315718 0.7490148 1.0000000 0.2865717
## par_ba   0.7041740 0.2980608 0.2865717 1.0000000
```

Per our expectations, the main diagonal is all 1s — every variable is perfectly correlated with itself — and the mirror cells on either side of the line are the same: [2,1] == [1,2] because the correlation between `bynels2m` and `byses1` is the same as `byses1` and `bynels2m` (the order doesn't matter).

Quick exercise *Run a correlation of both math and reading scores against whether the father has a Bachelor's degree (`fath_ba`) or the mother has a Bachelor's degree (`moth_ba`). In what ways, if any, is this different than what you get when using the combined `par_ba` indicator variable?*

t-test

One common statistical test is a t-test for a difference in means across groups (there are, of course, other types of t-tests that R can compute[4]). This version of the test can be computed using the R formula syntax: `y ~ x`. In our example, we'll compute base-year math scores against parent's college education level. Notice that since we have the `data = df` argument after the comma, we don't need to include `df$` before the two variables.

[3] https://en.wikipedia.org/wiki/Pearson_correlation_coefficient
[4] https://stat.ethz.ch/R-manual/R-devel/library/stats/html/t.test.html

```r
## t-test of difference in math scores across parental education (BA/BA or not)
t.test(bynels2m ~ par_ba, data = df)
```

```
##
##  Welch Two Sample t-test
##
## data:  bynels2m by par_ba
## t = -38.341, df = 13269, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##  -8.671327 -7.827819
## sample estimates:
## mean in group 0 mean in group 1
##        41.97543        50.22501
```

Looking at the bottom of the output, we see the mean math scores across the two groups:

- Math score of `41.9754334` for students for whom neither parent has a Bachelor's degree or higher
- Math score of `50.2250064` for students for whom at least one parent has a Bachelor's degree or higher

Are these differences statistically meaningful? Looking in the third line, we see a large t stat of `-38.34` which is highly statistically significant at conventional levels. Taken together, we can say that among this sample of students, those from households in which at least one parent had a Bachelor's degree or higher tended to score higher on the math exam than their peers whose parents did not have a Bachelor's degree or higher. Furthermore, this result is *statistically significant*, meaning that we can reject the null that there is no difference between the groups, that is, the difference we observe is just sampling noise.

Is the difference *practically significant*? Maybe, but to really know that we need to understand more about the design and scaling of the math test. Once we do, we can apply our content-area knowledge and place our findings in their proper context.

Quick exercise *Run a t-test of reading scores against whether the father has a Bachelor's degree (`fath_ba`).*

Using survey weights

So far we haven't used survey weights, but they are very important if we want to make population-level inferences using surveys with complex sampling designs. Yes, we can produce means, compute t-tests, and fit regressions without weights, but our results may not be externally valid due the fact that observations in our sample are almost certainly out of proportion to their occurrence in the population. Otherwise stated, it's likely that our sample under-represents some groups while over-representing others. The upshot is that any unweighted estimates will be a function of what we observe (sample) and not necessarily what we want (population). Furthermore, our standard errors — which we use when determining statistical significance — may be incorrect if we don't use survey weights, meaning that we may be more likely to commit Type I or Type II errors.

So that we can make population-level inferences, survey designers often include weights that allow us to adjust the amount each observation contributes to our estimates. With this adjustment our estimate should better reflect the population value. To use survey weights, you'll need to use the survey package (which we've already loaded above).

As a first step, you need to set the survey design using the `svydesign()` function. You could do this in the `svymean()` or `svyglm()` functions we'll use to actually produce our weighted estimates, but it's easier and

clearer to do it first, store it in an object, and then re-use that object.

ELS has a complex sampling design that we won't get into[5], but the appropriate columns from our data frame, `df`, are set to the proper arguments in `svydesign()`:

- `ids` are the primary sampling units or `psus`
- `strata` are indicated by the `strat_ids`
- `weight` is the base-year student weight or `bystuwt`
- `data` is our data frame object, `df`
- `nest = TRUE` because the `psus` are nested in `strat_ids`

Finally, notice the ~ before each column name, which is necessary in this function.

```
## subset data
svy_df <- df %>%
    select(psu,                          # primary sampling unit
           strat_id,                     # stratum ID
           bystuwt,                      # weight we want to use
           bynels2m,                     # variables we want...
           moth_ba,
           fath_ba,
           par_ba,
           byses1,
           lowinc,
           female) %>%
    ## go ahead and drop observations with missing values
    drop_na()

## set svy design data
svy_df <- svydesign(ids = ~psu,
                    strata = ~strat_id,
                    weight = ~bystuwt,
                    data = svy_df,
                    nest = TRUE)
```

```
## Error in svydesign(ids = ~psu, strata = ~strat_id, weight = ~bystuwt, : could not find function "svydesign"
```

Now that we've set the survey design, let's compare the unweighted mean with one that accounts for the survey design.

```
## compare unweighted and survey-weighted mean of math scores
df %>% summarise(bynels2m_m = mean(bynels2m, na.rm = TRUE))
```

```
## # A tibble: 1 × 1
##   bynels2m_m
##        <dbl>
## 1       45.4
```

```
svymean(~bynels2m, design = svy_df, na.rm = TRUE)
```

```
## Error in svymean(~bynels2m, design = svy_df, na.rm = TRUE): could not find function "svymean"
```

It's a little different!

We can also use the survey package to properly weight our t-tests and regression models — again, using the object `svy_df` in the `design` argument in place of our unset `df` data frame.

[5] https://nces.ed.gov/training/datauser/ELS_04.html

```r
## get svymeans by group
svyby(~bynels2m, by = ~par_ba, design = svy_df, FUN = svymean, na.rm = TRUE)
```

```
## Error in svyby(~bynels2m, by = ~par_ba, design = svy_df, FUN = svymean, : could not find function "svyby"
```

```r
## t-test using survey design / weights
svyttest(bynels2m ~ par_ba, design = svy_df)
```

```
## Error in svyttest(bynels2m ~ par_ba, design = svy_df): could not find function "svyttest"
```

Notice that unlike the base R `t.test()` function, `svyttest()` gives the difference between the groups. That's part of the reason we began by using `svyby()` function with `FUN = svymean`. Looking at the full output between the two functions, our results are a little different from what we got earlier without weights.

QUICK EXERCISE *Compare this to the output from `t.test()` above. How is it different?*

Two notes:

First, the **survey** library has a ton of features and is worth diving into if you regularly work with survey data. We've only scratched the surface of what it can do. But keep in mind that whatever statistical test you want to perform, there's likely a version that works with survey weights via this library.

Second, you'll have to spend **a lot** of time reading the survey methodology documents in order to understand which weights you should you use (context always matters) so that you can properly set up your survey design with `svydesign()`. But be warned: even then it won't always be clear which weights are the "correct" weights. That said, do as you always do: (1) your due diligence, and (2) be prepared to defend your choice as well as note potential limitations.