# In-class Lab 2

ECON 4223 (Prof. Tyler Ransom, U of Oklahoma)

January 25, 2022

The purpose of this lab is to practice using R to conduct hypothesis tests and run a basic OLS regression. The lab may be completed as a group. To get credit, upload your .R script to the appropriate place on Canvas. If done as a group, please list the names of all group members in a comment at the top of the file.

### For starters

Open up a new R script (named `ICL2_XYZ.R`, where `XYZ` are your initials) and add the following to the top:

```
library(tidyverse)
library(modelsummary)
library(broom) # you'll need to install this in the console first
library(wooldridge)
```

Load the dataset `audit` from the `wooldridge` package, like so:

```
df <- as_tibble(audit)
```

## A one-tailed test

The `audit` data set contains three variables: `w`, `b`, and `y`. The variables `b` and `w` respectively denote whether the black or white member of a pair of resumes was offered a job by the same employer. `y` is simply the difference between the two, i.e. `y=b-w`.

We want to test the following hypothesis:

$$H_0 : \mu = 0; H_a : \mu < 0$$

where $\mu = \theta_B - \theta_W$, i.e. the difference in respective job offer rates for blacks and whites.

### The `t.test()` function in R

To conduct a t-test in R, simply provide the appropriate information to the `t.test` function.

How do you know what the "appropriate information" is?

- In the RStudio console, type `?t.test` and hit enter.
- A help page should open in the bottom-right of your RStudio screen. The page should say "Student's t-Test"
- Under `Usage` it says `t.test(x, ...)`.
  - This means that *at minimum* we only have to provide it with is some object `x`. The `...` signals that we can provide it more than just `x`.
- Under `Arguments` it explains what `x` is: "a (non-empty) numeric vector of data values"
  - This means that R is expecting us to pass a column of a data frame to `t.test()`
- The other information in the help explains default settings of `t.test()`. For example:
  - `alternative` is `"two.sided"` by default

- `mu` is 0 by default
- ... other options that we won't worry about right now

Now let's do the hypothesis test written above. Add the following code to your script:

```
t.test(df$y,alternative="less")
```

R automatically computes for us the t-statistic using the formula

$$\frac{\overline{y} - \mu}{SE_{\bar{y}}}$$

All we had to give R was the sample of data (`y`, in our case) and the null value (0, which is the `t.test` default)!

### Interpreting the output of `t.test()`

Now that we've conducted the t-test, how do we know the result of our hypothesis test? If you run your script, you should see something like

```
> t.test(df$y,alternative="less")

	One Sample t-test

data:  df$y
t = -4.2768, df = 240, p-value = 1.369e-05
alternative hypothesis: true mean is less than 0
95 percent confidence interval:
        -Inf -0.08151529
sample estimates:
 mean of x
-0.1327801
```

R reports the value of the t-statistic, how many degrees of freedom, and the p-value associated with the test. R *does not* report the critical value, but the p-value provides the same information.

In this case, our p-value is approximatley 0.00001369, which is much lower than 0.05 (our significance level). Thus, we reject $H_0$.

## A two-tailed test

Now suppose instead we want to test if job offer rates of blacks are *different* from those of whites. We want to test the following hypothesis:

$$H_0 : \theta_b = \theta_w; H_a : \theta_b \neq \theta_w$$

This hypothesis test considers the case where there might be *reverse discrimination* (e.g. through affirmative action policies).

The code to conduct this test is similar to the code we used previously. (Add the following code to your script:)

```
t.test(df$b,df$w,alternative="two.sided",paired=TRUE)
```

You'll notice that the t-statistic is the exact same (-4.2768) for both of the tests. But the p-value for the two-tailed test is twice as large (0.00002739). This is because the two-tailed test must allow for the possibility of either direction of the $\neq$ sign. (In other words, that the job offer rate for blacks could be higher or lower than for whites.)

## Your first regression (of this class)

Let's load a new data set and run an OLS regression. This data set contains year-by-year statistics about counties in the US. It has counts on number of various crimes committed, as well as demographic characteristics about the county.

```
df <- as_tibble(countymurders)
```

A handy command to get a quick overview of an unfamiliar dataset is `glimpse()`:

```
glimpse(df)
```

`glimpse()` tells you the number of observations, number of variables, and the name and type of each variable (e.g. integer, double).[1]

### Regression syntax

To run a regression of $y$ on $x$ in R, use the following syntax:

```
est <- lm(y ~ x, data=data.name)
```

Here, `est` is an object where the regression coefficients (and other information about the model) is stored. `lm()` stands for "linear model" and is the function that you call to tell R to compute the OLS coefficients. `y` and `x` are variables names from whatever `tibble` you've stored your data in. The name of the `tibble` is `data.name`.

### Regress murder rate on execution rate

Using the `df` data set we created above, let's run a regression where `murders` is the dependent variable and `execs` is the independent variable:

```
est <- lm(murders ~ execs, data=df)
```

To view the output of the regression in a friendly format, type

```
tidy(est)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)     6.84     0.242      28.3 3.97e-174
## 2 execs          65.5      2.15       30.5 7.44e-202
```

In the `estimate` column, we can see the estimated coefficients for $\beta_0$—`(Intercept)` in this case—and $\beta_1$ (`execs`). `est` also contains other information that we will use later in the course.

You can also look at the $R^2$ by typing

```
glance(est)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic   p.value    df   logLik    AIC    BIC
##       <dbl>         <dbl> <dbl>     <dbl>     <dbl> <dbl>    <dbl>  <dbl>  <dbl>
## 1    0.0243        0.0243  46.6      930. 7.44e-202     1 -196508. 3.93e5 3.93e5
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Again, there's a lot of information here, but for now just focus on the $R^2$ term reported in the first column.

We can also view the output in a table format with `modelsummary()`:

[1] "double" means "double precision floating point" and is a computer science-y way of expressing a real number (as opposed to an integer or a rational number).

| | Model 1 |
|---|---|
| (Intercept) | 6.838 |
| | (0.242) |
| execs | 65.465 |
| | (2.146) |
| Num.Obs. | 37 349 |
| R2 | 0.024 |
| R2 Adj. | 0.024 |
| AIC | 393 021.2 |
| BIC | 393 046.8 |
| Log.Lik. | −196 507.599 |
| F | 930.365 |

```
modelsummary(est)
```