Hypothesis Testing and Stored Results

Hypothesis Testing

- One of the most common exercises for new users to statistical software is to perform basic hypothesis testing
- Stata has very easy to use commands to perform t-tests, anovas, linear regressions, and logistic regressions
- Let's begin with some t-tests concerning the mean price of cars from our `auto` data

ttest

```
ttest cont_variable == value
```

- This command performs a one-sample t-test for whether the mean of a sample continuous variable (`cont_variable`) is equal to a fixed `value`

```
ttest cont_variable, by(dich_variable)
```

- This command performs a two-sample t-test for the equality of means of a (`cont_variable`) across two values of the dichotomous variable (`dich_variable`)

- There are many other options available via **`help ttest`**

ttest

```
ttest cont_variable == value
```

- Let's check example output from the following:

```
sysuse auto
```

```
ttest mpg == 30
```

and

```
ttest mpg, by(foreign)
```

Stored Results

- As we just saw with **`ttest`** Stata presents a lot of output information in response to a statistical command
- Often the user will want to use some aspect of these results later on in a script
- The *wrong way* to do this is to manually write down the result we want
- Luckily, Stata saves the outcome of your last command as results. The *right way* is to access these `returns` using an `r()` call
- Let's walk through an example using **`summarize`**

summarize r()

DON'T DO THIS:

```
summarize mpg

generate above_average_mpg = 1 if mpg > 21.2973
```

- Prone to copy/paste errors
- Not reproducible — what if the next time you run your script the mean changes and you forget this line?

summarize r()

INSTEAD, DO THIS!

```
summarize mpg

generate above_average_mpg = 1 if mpg > r(mean)
```

- Instead of manually copying down what we read from the summarize command call, we are accessing it directly in Stata's memory

- Stata saves the results of your last command in `returns`

return

return list

- This command will list out all the available saved returns available from a particular statistical command
- All of these values are available via `r()` results, but they only represent the last **command** issued
  - This can be important to remember when running multiple loops

ereturn

- Many statistical commands include estimated results, which are stored in `ereturn`:

```
ereturn list
```

- `ereturn` stored values are generally connected to model fitting and parameters

- Moving back to our **`ttest`** example we can see what results are available via `r()` calls and `e()` calls

regress

```
regress dependent_var ind_var1 ind_var2 ...
```

- The `regress` command will construct a linear model with a continuous outcome variable (`dependent_var`) and independent predictors (`ind_var1 ind_var2` etc)

- The output of this command contains what we would expect from a linear model, with easily interpretable model coefficients

```
regress price mpg headroom
```

regress

ereturn list

- Since regression is a modeling command the majority of its results are stored in `ereturn`, or estimated returns
- The difference between `return` and `ereturn` can be subtle, so if you are looking for a particular return, it is good to check both (along with the **`help`** file for the command you want to use)
- For many commands, there will be output that you might also want to use that are not stored in `return` or `ereturn`
- I was often interested in running regressions and recording the estimated coefficients for predictor variables

regress _b[var]

- It turns out, Stata holds on to both regression coefficients and their standard errors in `_b[var]` and `_se[var]` macros

```
regress price mpg headroom
```

- In this example we can access `_b[_cons]`, `_b[mpg]` and `_b[headroom]` values.

- This would allow us to estimate a particular value based on our model if we were interested:

```
display _b[_cons] + _b[mpg]*25 + _b[headroom]*2
```

- This displays a model estimate for the price of a car with 25 miles per gallon and 2 inches of headroom

Stored values in strings

- When writing a string, such as a note or title text in a graph, or a string to display, you can use stored values
- However, Stata needs to know that when you are writing r(mean) in a string you intend for it to reference a stored value, and not actually r(mean)!
- To indicate your intent to Stata, use the familiar backtick + apostrophe combination:

```
summarize mpg

notes mpg: The mean mpg is r(mean)

notes mpg: the mean mpg is `r(mean)'
```

Summary

- Using stored results help reduce human error when copying and pasting specific values
- Using stored `results` makes your scripts more reproducible and flexible — if the underlying data changes your code immediately adapts
- Check both `return` and `ereturn` for command results and estimated results
- Check **`help`** files and additional documentation for stored results (like `_b[var]` in **`regress`**) that may be useful for your particular purposes

Exercises (1)

1. Titanic Data

   A. Perform a one sample t-test to test the hypothesis that the mean age on the titanic is different from 32 years of age. Use stored results to add the two-sided p-value and t-statistic to a note on the age variable.

   B. Perform a two-sample t-test to test the hypothesis that age differs between those who survived the titanic and those who did not. Use stored results to add the two-sided p-value and t-statistics to a note on the survival variable.

Exercises (2)

2. Auto Data

   A. Write a loop that creates three new standardized variables for mpg, price, and headroom (subtract the mean and divide by the standard deviation). These should be named *var*_standard.

   B. Perform a linear regression with standardized price as the dependent variable and standardized mpg and standardized headroom as predictors. Create new variables beta_mpg and beta_headroom to hold the coefficients for mpg and headroom.

   C. Run the same regression as B, but with unstandardized variables – this time include the option **beta** in your regress command. Compare the Beta column of your output to your beta_mpg and beta_headroom variables. *Check the documentation and return lists – is there some way to pull out these standardized coefficients?*