# Stata for Epidemiology, Lab 4

PH241, Spring 2018

In this lab, we'll cover tools you'll need to complete Homework 5, including:

- Hypothesis testing
- Confidence intervals
- Obtaining critical values and p-values from Gaussian and $\chi^2$ distributions
- How to create, modify, and delete variables

*Note*: To generate a nice record of the work you do during this week's lab, we recommend that you create a do-file and log, as described in previous lab handouts. From now on, we'll leave it up to you to take care of this during section. Please ask for help if you have trouble.

## 1 Hypothesis tests, confidence intervals, and $\chi^2$

We'll be using the WCGS dataset, as in lab 2. Load it into Stata as you'd like.

In this dataset, each man in the study is represented by a row. Each row contains, among other variables, the values of two binary (0/1 indicator) variables called `chd69` (Coronary Heart Disease) and `arcus0` (Arcus senilis, a ring in the corneal margin or around the iris that can indicate CHD).

### 1.1 The Normal approximation for proportions, exact confidence intervals, and Fisher's exact test

Using the Normal curve for inferences about proportions is actually an approximation. For confidence intervals for a proportion, the number of observed successes and failures should be at least 10. For hypothesis tests about a proportion, the expected numbers of successes and failures should be at least 10. For two-sample problems with binary outcomes, where the data are summarized in a 2 x 2 table, the criteria are a bit different – many authors say that the observed counts in each cell should be greater than 5 to use the Normal approximation, and for the two-sample test for proportion or the equivalent $\chi^2$ test, the expected counts should all be greater than 5.

What if our data don't satisfy the assumptions for the Normal or $\chi^2$ distributions? For these situations, there are exact tests based on the binomial distribution and its relatives. Stata allows us to request exact p-values for all the test commands we've been using (e.g., `tab`, `cs`, `cc`), and to construct "exact" confidence intervals.

*Note*: The defaults for confidence intervals and tests are not consistent in Stata. By default, Stata calculates **approximate** $\chi^2$ tests, but the default for confidence intervals is **exact**.

### 1.1.1 Using cc to construct confidence intervals for an odds ratio

There are 3 options:

1. Exact (the default)

   ```
   cc disease_var exposure_var
   ```

   For example,

   ```
   cc chd69 arcus0
   ```

2. Woolf (the formula we've been using with the normal z critical value and the standard error estimate for log(OR))

   ```
   cc disease_var exposure_var, wo
   ```

   For example,

   ```
   cc chd69 arcus0, wo
   ```

3. Cornfield (a more accurate approximation)

   ```
   cc disease_var exposure_var, cornfield
   ```

   For example,

   ```
   cc chd69 arcus0, cornfield
   ```

### 1.1.2 Specifying your confidence level (1 - $\alpha$)

- Stata's default for confidence intervals is 95%
- The `level` keyword can be used to override this default
  - You may specify the level after your individual command

    ```
    cc disease_var exposure_var, wo level(99)
    ```

    For example,

    ```
    cc chd69 arcus0, wo level(99)
    ```

  - You may set the default level for your Stata session with the `set` command

```
set level 99
```

- The values of the confidence level must be between 10.00 and 99.99

## 1.2 Obtaining critical values and confidence levels

We learned previously how to find the p-value associated with a particular $\chi^2$ test statistic by typing the following:

```
display 1 - chi2(1, 13.6382)
```

Or, equivalently, by typing the following:

```
display chi2tail(1, 13.6382)
```

where 1 is the number of degrees of freedom and 13.6382 is our test statistic (from the WCGS dataset examples above). Remember this value is the area in the right hand tail of the $\chi^2$ distribution with a specified number of degrees of freedom.

*Note*: these commands will not work if you insert a space between `chi2` and the open parentheses, between `chi2tail` and the open parentheses, or if you omit the `display` command preceding either `chi2` or `chi2tail`.

*Note*: though the `cc` and `cs` commands will automatically give you the p-value associated with your $\chi^2$ statistic, these commands round off the answer; `chi2tail` will give you more significant digits.

We can similarly find the p-value associated with a particular test statistic (z-value) based on the Standard Normal distribution. For a 2-sided test, this would look like:

```
display 2 * ( 1 - normal(1.96) )
```

The `chi2`, `chi2tail`, and `normal` functions also have inverses. These functions take a probability as input and will give you its associated critical value. Be careful about making adjustments for 2-tailed tests if you want a particular confidence level. For example, for a 2-tailed test at the 95% confidence level (where $\alpha = 0.05$), the critical z-value is found as follows:

```
display invnormal(.975)
```

Looking up two tails of the $\chi^2$ distribution for a hypothesis test is only appropriate when constructing a confidence interval. Recall that the $\chi^2$ distribution is not symmetric, so it is necessary to to look up both tails separately.

```
display invchi2(1, .025)        // 1 represents the degrees of freedom
display invchi2(1, .975)
```

## 2 Creating and modifying variables with Stata

*Note*: The code below assumes you still have the WCGS dataset loaded into Stata.

The `generate` command (or `gen` for short), followed by an equal sign (=), allows you to create new variables. For example, to create a variable called `beh_dum` that equals 0 for all observations, type the following:

```
gen beh_dum = 0
```

You can change the values that a variable takes using the `replace` command. By combining the replace command with an `if` statement, you can condition on the values of other variables. For example, we have another variable that's called `behpat0` (hint: this would be similar to `alcgp` in homework 5, problem 2), and this variable takes the values 1, 2, 3, and 4. If we want our `beh_dum` variable to take the value of 1 when `behpat0` is equal to 3 or 4, then we could type any **one** the following:

- ```
  replace beh_dum = 1 if behpat0 > 2
  ```

  Note that `beh_dum` will only change for the observations for which `behpat0 > 2`; for all other observations, `beh_dum` will still equal 0.

- Assign `beh_dum` the value of 1 if behpat0 is greater than or equal to 3:

  ```
  replace beh_dum = 1 if behpat0 >= 3
  ```

- Assign `beh_dum` the value of 1 if `behpat0` is equal to 3 or `behpat0` is equal to 4:

  ```
  replace beh_dum = 1 if behpat0==3 | behpat0==4
  ```

  That's right – Stata interprets the vertical straight line `|` as "or" and insists that you include two back-to-back equal signs if you want it to evaluate whether a certain statement is true. If you want your "if" statement conditional on two or more statements, Stata interprets the ampersand sign `&` as "and".

**For your reference: a complete list of comparison operators** (*these commands won't work in the current environment as the variables a and b don't exist*) :

```
a > b // greater than
a < b // less than
a >= b // greater than or equal to
a <= b // less than or equal to
a == b // equal to
a != b // not equal to
```

*Note*: Stata interprets missing values (.) as positive infinity for the purposes of deciding whether these comparisons are true or false. That means missing values will always be bigger (and will never be smaller) than any actual data value.

For example, a more complicated condition:
```

```
replace beh_dum = 1 if behpat0==3 & age0 > 45
```

*Note/Warning*: Stata will not let you "generate" a variable that already exists. That is, you will get an error message if you try to run `gen beh_dum = 0` twice (much like you get an error if you try opening a log that is already open). To re-generate a variable, you must first delete it using the `drop` command (or reload your data such that it no longer contains the variable):

```
drop beh_dum
```

For those who just can't get enough of Stata – a one line short-cut for creating dummy variables:

```
gen beh_dum = (behpat0>2)
```

How the above line works: when the statement `behpat0>2` evaluates to true, Stata will assign `beh_dum` a value of 1; for all other observations, `beh_dum` will get a value of 0.