INTRODUCTION TO STATA II

OBJECTIVES: In this module, you will learn how to:

1. Explore the data and their summary statistics using basic STATA commands
2. Describe the difference between continuous and discrete variables; and individual-level and household-level variables
3. Generate useful new variables from old variables
4. Identify one observation to represent information for the household
5. Examine subsets of your data
6. Summarise data using scatter plots and line graphs
7. Introduction to regression with STATA
8. Saving your data

1. STARTING UP

Before you start this lab session remember to open BOTH a log file (to record your results) and a cmdlog file to record your commands. Remember to create your log file as a text file so that you can easily read it in Word. Stata's default is to save log files as scml files which are very long and impossible to read in a text editor. The simplest way is to open your log file using the command line and specifying a log extension. For example

```
log using F:\yourlastname_LAB2.log
```

Start a command log file by typing:

```
cmdlog using F:\yourlastname_LAB2.do
```

Open up the data set **personworker.dta**. This dataset was created using the Person and Worker data sets.

2. RECODING VARIABLES

Sometimes we want to recode a variable. This could be because we do not like the way the variable was constructed originally and/or because we need to have the variable in an alternative form to analyse it.

EXAMPLE 1: RECODING MISSING VALUES

In STATA a missing value is usually recorded as a full stop/period. If you look at the metadata for the GHS you will notice that Stats SA generally uses 9 (or 99, 999 etc) for "Unspecified" and 8 (or 88, 888 etc) for "Not applicable". When we start to analyse the data we will need to recode the 9's to missing so that STATA knows to treat these observations as missing.

We use the **replace** command.

```
replace E_Race= . if E_Race ==9
replace Q15Read= . if Q15Read ==9
```

Regenerate the cross tabulation of race and ability to read

```
tab E_Race Q15Read
```

Notice how the table only includes valid (non missing) observations. If we want to include the missing values we use the missing option after the tab command.

```
tab E_Race Q15Read, missing
```

When we summarize variables, and when we start to analyse relationships between variables, we need to be very careful about missing data for numeric variables – the period is interpreted as positive infinity by STATA.

3. BASIC SUMMARY STATISTICS: using **tabsum** and **sum**

We briefly looked at the tabulation command in the last lab. While the **tab** command provides a summary of the distribution of the data in the form of frequency tables, the **summarize** command gives more detailed summary statistics, like the mean, standard deviation and range of the values that the variable takes on. If we type

```
sum E_Race
```

we find that the mean of this variable is 1.41, and the range is from 1 to 4. But what does it mean to say that the 'average' race in our data set is 1.41? It doesn't mean anything because race is a categorical, or discrete, variable. The numbers which represent African or White do not mean anything numerically, so it doesn't make sense to have the mean of this variable. We'll look at variable types in the next section. If we want to find some meaningful summary statistics, we should use the **summarize** command with continuous/numerical variables. For example,

```
sum D_Age
```

will generate the mean, sample standard deviation and range of this variable. The mean age is 28.3 years. Notice that the maximum age is 999. We need to recode this so that STATA knows that 999 is actually missing and does not use it as a valid observation in the calculation of the mean.

```
replace D_Age=. if D_Age==999
sum D_Age
```

The average age is now 27.3 and the oldest person is 130. The sum command may also be used with any of the qualifying, comparison or logical operators above, and may be used to generate sample statistics for multiple variables at a time. Think about what each of the following commands will give you, and then execute them:

```
sum D_Age edyears salary
sum D_Age if C_Gender==2
sum edyears if D_Age>=18
sum edyears if D_Age>=18&D_Age!=.
```

Look at the number of observations for the last two commands. Notice how STATA regards missing values as positive infinity – we need to be extremely careful about missing values. The **summarize, detail** command will give even more detailed statistics such as the value of the variable at each of a number of percentiles, the variance, skewness and kurtosis of the data. For example,

```
summarize D_Age, detail
```

tells us that 75% of the sample have age values less than 40 years, the skewness of the age data is 0.766 and the kurtosis is 2.89.[1]

THE TABSUM COMMAND

The tabulate and summarize commands may also be used together. For example, we might want to know the average age of each population group in the sample, or the average education level of the observations in each province, or the average level of household income for each population group. For these questions, we would type:

```
tab E_Race, sum(D_Age)
tab Prov, sum(edyears)
tab E_Race, sum(salary)
```

We can also use the tabulate and sum in a two way table. For example we might want to know the average age by population group and by location. Here we would type:

```
tab Prov E_Race, sum(D_Age)
```

It can be seen from the output that the table gives us means, standard deviations and frequencies for each cell. This can be overwhelming. If we are interested only in means, we can request this by typing:

```
tab Prov E_Race, sum(D_Age) means
```

If we want means and standard deviations but not frequencies, we type:

```
tab Prov E_Race, sum(D_Age)means standard
```
OR
```
tab Prov E_Race, sum(D_Age)nofreq
```

[1] Skewness tells us how asymmetrically distributed the values of a variable are. If the skewness measure is zero, it means the values in the variable are symmetrically distributed about the mean; if the skewness coefficient is >0 (<0), then the values are skewed to the right (left) of the mean. Kurtosis tells us about the thickness of the tails of the distribution, or what proportion of our sample has values for the variable that lie in the tails (end points) of the distribution.

Tabsum is a key tool in descriptive analysis. There are more options than we show here. Use the 'help tabsum' in Stata to find out about these options.

Generate a table that displays in each cell a mean salary by population group and province.

4. CONTINUOUS vs CATEGORICAL VARIABLES

In the GHS data set, there are several types of variables. Economists, sociologists, and psychologists use different language to describe the multiple types of variables. Here, we will divide variables into two types: Continuous variables and categorical variables. The statistical and graphical tools used to understand the distributions of the various types of variables are quite different, so it is important to understand the differences between these measures.

CONTINUOUS VARIABLES
Continuous variables have an infinite number of possible values that fall between any two observed values. For example, consider age. In our data, age is recorded in years. But it could have been recorded in months, days, minutes, or even seconds. A continuous variable is ordinal in the sense that its values have an inherent order. In the age example, an age of 16 years is one year older than the age of 15 years, thus the unit of measurement in between these two values is itself meaningful. (This may seem like common sense, but when we consider categorical variables, this will no longer be true). Examples of continuous variables in the GHS data set include age (**D_Age**) and salary (**Q27Salto**).

We are actually not being terribly careful with our definitions. Consider, for example, a variable that counts members of the household (**hhsizem**). Household size might be 4 or 5, but it will never be 4.34. Nonetheless, if we were told that the average number of members of a household was 4.34, this would be comprehensible. We would know that, on average, there are more than 4 members and less than 5. We are going to treat variables like household size and number of births as continuous variables: taking the average gives an answer that is readily interpreted. Taking the average of a categorical variable, on the other hand, yields nonsense.

CATEGORICAL VARIABLES
These are made up of separate and distinct categories which do not have an inherent order. To code these variables, each category is typically assigned a value, but this assignment is arbitrary. Take for example the race variable. Each racial group is assigned an arbitrary value. In the data set, if a person is African, the race variable for that person is set to 1. If the person is Coloured, the value is set to 2, Indian is 3, and White is 4. Gender is indicated by the variable **C_Gender**, where males are coded as 1 and females as 2. Other examples of what we will consider categorical variables include the relationship to the head of household (**q11relsh**), and province (**Prov**).

DUMMY VARIABLES
A special type of a categorical variable is a dummy/indicator variable. A dummy variable is a variable that typically takes on a value of one if the observation meets specified criteria and a value of zero if otherwise. Observations that don't have the required information are assigned a missing value. In the next section, we will learn about constructing dummy variables. Later in the course you will see that these dummy variables are very useful in regression analysis.

In general, it is important to know the types of variables you are using because some of the tools used to analyze variables differ depending on whether the variable is continuous or categorical. Another point to keep in mind is whether the variable you are using is an individual-level or household-level variable.

5. INDIVIDUAL vs HOUSEHOLD VARIABLES

INDIVIDUAL-LEVEL VARIABLES
These are made up of values that are unique to each person in the household. An example is the variable for age (**D_Age**). To see an example of an individual-level variable, type the following:

```
sort UqNr
list UqNr D_Age
```

As we can see, each person in the same household has an age value that is unique to them. Other examples in the GHS data sets would include the following individual-level variables: **C_Gender** and **edyears** (educational attainment level).

HOUSEHOLD-LEVEL VARIABLES
They have the same value for every person in the household. An example would be the variable for province (**Prov**). To see an example of a household-level variable, type the following commands:

```
sort UqNr
list UqNr D_Age edyears Prov
```

As we can see, each person in the same household has the same value for province.

6. GENERATING NEW VARIABLES

Often, we want to create new variables from the variables in the data set. This could be because we do not like the way the variable was constructed originally and/or because we need to have the variable in an alternative form to analyse it; because we need to create combined categorical variables or because we simply need another new variable. Commands we use are **generate, replace** and **egen**.

EXAMPLE 1: GENERATING A WHITE/NONWHITE VARIABLE
Suppose we want to create a variable to indicate whether an individual is white or nonwhite. We can use the **E_Race** variable to create a new variable called **white** that will be equal to 1 if the individual is white and equal to zero if individual is not white. We use the **generate** command, which can be shortened to **gen**.

```
gen white=0
replace white=1 if E_Race==4
replace white= . if E_Race ==.
```

Once a variable has been generated we cannot generate it again. So we have to use the **replace** command to make additional changes to the variable. Usually it will take a **generate** command followed by one or more **replace** commands to completely define the new variable. To confirm that the variable was created correctly, we can tabulate **E_Race** by **white,** including the missing option to confirm that the missing values in the original variable are also coded as missing for the new variable:

```
tab E_Race white, missing
```

Generate a dummy variable called *male* that equals one if the individual is male and 0 if not. If the gender variable is missing for the individual the *male* dummy variable should be set to missing.

Suppose we want to create a dummy variable for each race. We could create four variables, one at a time, but there is a quicker way using the **tab** command with the **gen** option:

```
tab E_Race, gen(race)
```

This generates a dummy variable for each category of **E_Race**. The dummy variables are named **race1** to **race4**. To verify that **race1** is a dummy variable for Africans, type:

```
tab E_Race race1
```

Let's rename the race1 variables so we don't get confused.

```
rename race1 African
rename race2 Coloured
rename race3 Indian
rename race4 White
```

EXAMPLE 2: TRANSFORMING A VARIABLE
Often it is more convenient to use the log of income variables, than to use the income variable itself in descriptive and analytical work. Taking the natural log of such a variable will reduce the effect of outliers on the income distribution. Take a look at the monthly wage variable (**salary**) using **codebook** and **sum**. How many of the observations are missing? Are there any negative values?

Type in the following:

```
gen lnsalary=ln(salary)
list salary lnsalary
```

This sequence will generate a new variable which is the log of total monthly income. Note that all the missing values for **salary** are still missing under **lnsalary**.

If you want to attach labels to your new variables, you can use the label command:
```
label variable lnsalary "Log of monthly wage"
```

EXAMPLE 3: THE EGEN COMMAND
Another powerful command for making new variables is **egen**. Let's say we want to create a household-level variable which contains the total number of people living in the household.
```
egen hhsize=count(PersonNr), by(UqNr)
tab hhsize
```

The egen command told STATA to count the number of people with non-missing values for the **PersonNr** variable for everyone with the same value of **UqNr**. So this is just a count of the total number of people in each household. This count is stored in the variable **hhsize**.We can see that some household/s have over 30 members! If we type:
```
list UqNr PersonNr D_Age C_Gender hhsize
```

we see that the value of **hhsize** is the same within households, for each member of the household. If we want to see a frequency table of how many households contain different total numbers of household members, we need to pick out ONE **hhsize** value to represent each household. We will do this in the next section.

Counting observations is only one of the many functions **egen** is capable of. It can also create means, minima, maxima, medians, percentiles, to name just a few of the most commonly used functions. Here is one more example, where we use egen to expand values from one observation to other observations in the household. Suppose we have some theory which says the higher the level of education the head of household has, the more likely the other members will have high levels of education. Therefore, we want to create a variable which records for every individual in the household, the level of education of the head of the household, Try the following:
```
gen temp=edyears if q11relsh==1
egen headeduc=max(temp), by(UqNr)
drop temp
```

To create a variable in which contains the education of the household head, we first create a temporary variable equal to missing for everyone except household heads, for whom it equals the years of education. This means that there is only one non-missing value of **temp** per household. The second step is to use the **egen** command to calculate the maximum of **temp** across all observations with the same value of **UqNr.** Since there is only one non-missing value of **temp** in the household, which is equal to the head's years of education, the maximum of **temp** is simply the education of the head - but **egen** stores it in a variable called **headeduc** for everyone in the household. (If we had taken the minimum or mean we would have gotten the same result.)

Type **help egen** in STATA to see more. Note that every time you generate a new variable, its name should appear in your list of variables window.

7. IDENTIFYING & CREATING HOUSEHOLD-LEVEL VARIABLES

USING SORT
Suppose we want to sort the observations by gender. Type
```
list C_Gender
sort C_Gender
list C_Gender
```
The sort command can be used with several variables. For example, you might want to sort first on province, and then on gender and race. You would type:
```
sort Prov C_Gender E_Race
```

USING [_n]
If we want to see a frequency table of how many households contain different total numbers of household members, we need to pick out ONE **hhsize** value to represent each household. To accomplish this task, the [_n] option is very

helpful. Using the [_n] option allows us to treat an individual-level variable as a household-level variable. For this example, we would type:

```
sort UqNr
tab hhsize if UqNr~=UqNr[_n-1]
```

The qualifier **UqNr~=UqNr[_n-1]** is telling STATA to go to every UqNr and include it in the tabulation only if that UqNr is not equal to the one before it.

Use [_n] to generate a frequency table of the household head's education for each household.

8. EXAMINING SUBSETS OF YOUR DATA

Sometimes, we want to work with subsets of our data. For example, we may only be interested in working adults. We use the **keep** and **drop** commands to create subsets of our data.

Before we drop some of our observations lets preserve our current data set so that we will be able to recover the dropped observations at a later stage. The **preserve** command saves our current data set in memory. We can then recover this same data set at any point by typing **restore**. Type

```
preserve
```

Suppose we are only interested in female observations. To make our data set smaller, we can

```
drop if C_Gender==1
```

Note that when we DROP observations, all the information associated with the observations that meet the 'if' criteria will be erased from our data set, and if we want them back, the original read-only data set should be opened again (unless you have use the **preserve** command).

We can use the count command to see how many of these women are able to read in at least one language.

```
count if Q15Read==1
```

What percentage of these women are under 18 years of age?

Suppose we further narrow our sample of interest to adult (18 years and older) women. Restrict the data set by typing in

```
keep if D_Age>=18&D_Age~=.
```

Now using the **count** command, how many observations do we have left?

Now that we know how to select subsets of our data, let's go back to the original data set by typing **restore**. Type

```
restore
count
```

The original data set with 102,461 observations should be restored.

9. ANALYZING EDUCATION AND AGE USING SCATTERPLOTS

We are going to compare the mean years of completed education (**edyears**) by race against age. First, generate the mean years of completed education by age and race.

```
egen meaned=mean(edyears) , by(D_Age E_Race)
```

For graphing purposes it will be convenient to generate two mean income variables, one for Africans and one for whites.

```
gen edafrican=meaned if E_Race==1
gen edwhite=meaned if E_Race==4
```

It will be nice to have variable labels for these new variables:

```
label variable edafrican "Mean Years of Education – African"
label variable edwhite "Mean Years of Education – Whites"
```

To generate the line graphs of mean years of completed education against age for Africans, we would use the following command:

```
scatter edafrican D_Age, c(l) sort
```

The last part of this syntax says: connect with a line. We can also put multiple line graphs on the same graph space:

```
scatter edafrican edwhite D_Age, c(l l) sort
```

Since there are very few observations for people over the age of 80, we could exclude those observations using an if statement:

```
scatter edafrican edwhite D_Age if D_Age<80, c(l l) sort
```

We could add in x and y-axis titles and other additional details to the graph. Note that we can copy the graph and paste it into Word using by right-clicking on the graph. We can also save a graph in Stata format by right-clicking on the graph and giving a file name for the graph.

When we create a graph Stata will plot a point for every observation. This may mean that there are as many as 30,000 points on your graph, many of them describing the same x,y coordinate. This can make the graphs slow to display on the screen and difficult to print or to copy and paste. In the case of our meaned variable, all observations that have the same combination of age and race will be given the same value of meaned by the egen command. We therefore only need one point on the graph to represent each age/race cell. To eliminate this problem, we can tell STATA to plot only one point per group (age and race in this case). To do this, we can use another application of the egen command to create a variable that numbers the observations sequentially within each age/race cell.

```
egen number=seq(), by(D_Age E_Race)
scatter edafrican edwhite D_Age if number==1&D_Age<80, c(l l) sort
```

It may be easier to get a feel for the data if we look at children of school going age and adults separately, by using if statements:

```
scatter edafrican edwhite D_Age if number==1&D_Age>=6&D_Age<=18, c(l l) sort
scatter edafrican edwhite D_Age if number==1&D_Age>=18&D_Age<=80, c(l l) sort
```

10. SIMPLE LINEAR REGRESSION

In the scatter plots above we have seen there seems to be a linear relationship between age and years of complete education for adults. We go ahead and run a regression analysis:

```
reg edyears D_Age if D_Age>=18&D_Age<=80
```

```
      Source |       SS       df       MS              Number of obs =   60785
-------------+------------------------------           F(  1, 60783) =12046.11
       Model |  179136.396     1  179136.396           Prob > F      =  0.0000
    Residual |  903897.248 60783   14.870889           R-squared     =  0.1654
-------------+------------------------------           Adj R-squared =  0.1654
       Total |  1083033.64 60784  17.8177422           Root MSE      =  3.8563

------------------------------------------------------------------------------
     edyears |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       D_Age |  -.1107939   .0010095  -109.75   0.000    -.1127725   -.1088154
       _cons |    12.4363   .0417075   298.18   0.000     12.35455    12.51805
------------------------------------------------------------------------------
```

Make sure that you can interpret all coefficients and statistics in the regression output (if you are rusty - a good standard text book treatment of regression analysis is Gujarati's "Basic Econometrics").

We can use the fitted model to generate predicted values for each individual.

```
predict p0
```

Note that the predicted values are generated by the following equation:

$p0_i = a + bX_i$

So the mean of **p0** should just be $a+b\overline{X}$, which should be the same as $\overline{Y}$. Let's confirm this, using just the observations that were included in the regression (that is, restricting our analysis to the observations with non-missing values for **edyears**.

```
sum p0 edyears if D_Age>=18&D_Age<=80&edyears~=.
```

We can produce a scatter plot of the predicted values.

```
egen meaned2=meand(edyears), by(D_Age)
scatter p0 meaned2 D_Age if D_Age>=18&D_Age<=80, c(l)
```

11. REGRESSION WITH DUMMY VARIABLES

Now let's look at the difference in average years of completed education for adults (18 to 80 years old) men and women by regressing years of education on gender:

```
reg edyears male if D_Age>=18&D_Age<=80
```

How do we interpret the coefficient for male?

(1) remember that the regression line always passes through the mean values of our data: $\overline{Y}$ = a + b $\overline{X}$1
(2) if we consider women only, then each women has `male==0`, which implies that predicted mean education for females = `a`.
(3) if we consider men only, each man has `male==1`, thus predicted mean education for males =`a + b`
(4) So, the difference in predicted mean education for men and women is b

Let's check that this is the case.

```
tab male if D_Age>=18&D_Age<=80, sum(edyears)
```

From our table above, generate the difference in male/female education (use the `display` command).

```
disp 8.3752-8.0388
```

The difference is 0.3364.

Note that this estimated coefficient from the regression and its associated t statistic give us the same result as a formal hypothesis test for the difference between means.

12. MULTIPLE REGRESSION

Regress years of education on age and gender.

```
reg edyears D_Age male if D_Age>=18&D_Age<=80
```

Make sure that you can interpret all coefficients and statistics.

In section 2 we observed distinct racial differences in the relationship between years of completed education and age for both adults and children. Let's revisit our simple regression model from section 3 by looking at a scatter plot of the fitted equation and the mean level of education at each age for Africans or Whites separately:

```
scatter p0 edafrican edwhite D_Age if D_Age>=18&D_Age<=80, c(l )
```

The fitted model is clearly inappropriate. Let's fit a new model including race as an explanatory variable.

```
reg edyears D_Age African Coloured Indian if D_Age>=18&D_Age<=80
```

Let's look at a scatter plot for our new model.

```
predict p1
gen africanp1=p1 if African==1
gen whitep1=p1 if White==1
scatter africanp1 whitep1 edafrican edwhite D_Age if D_Age>=18&D_Age<=80, c(l l)
```

In order to introduce race into our model we needed to create three dummy variables. We can do this using **gen** and **replace** or **tab** with the **gen** option. There is a quicker way to introduce categorical variables into your regression model using the **xi** command. The **xi:** command works by placing it at the beginning of the regression equation and then specifying the categorical variables you want STATA to expand into its constituent categories by "tagging" them with an "i." in front of each target variable. The command:

```
xi: reg edyears D_Age i.E_Race if D_Age>=18&D_Age<=80
```

produces the same results as:

```
reg edyears D_Age African Coloured Indian if D_Age>=18&D_Age<=80
```

The results are not exactly the same because when using **xi** STATA, by default, selects the first category in the specified variable as the reference category. In this instance Africans would be the reference category. If we wanted Whites to be the reference category we could preface the **regress** command with the **char varname[omit]** statement. To make category 4 (Whites) of the race variable the reference category type:

```
char E_Race[omit] 4
xi: reg edyears D_Age i.E_Race if D_Age>=18&D_Age<=80
```

The results should now be identical to those produced without the **xi** command.

13. INTERACTION

This model above allows for different intercepts for each race but from the scatter plot it seems that we should allow for different slopes for each race group. We need to introduce interaction terms between race and age. We could create an interaction for each race separately by **gen africanage=African*D_Age** and so on but it will be much easier to use the **xi** command.

```
char E_Race[omit] 4
xi: reg edyears i.E_Race*D_Age if D_Age>=18&D_Age<=80
predict p2
gen africanp2=p2 if African==1
gen whitep2=p2 if White==1
scatter africanp2 whitep2 edafrican edwhite D_Age if D_Age>=18&D_Age<=80, c(l l)
```

14. SAVING YOUR DATA

If we've created new variables we may want to save your Stata data set in your own data folder to work with again. Click on "File, Save" in the Stata window, and save the file in your folder. Stata will warn us if we are writing over a data set with the same name. We need to be careful about whether we want to write over the existing data set. If we make a mistake you can always go back to the original data sets.

LIST OF NEW STATA COMMANDS INTRODUCED IN MODULE 2

[_n]	char	count
disp	drop	egen
gen	keep	label
predict	preserve	reg
replace	restore	scatter
sort	sum	sum, detail
tab, gen	tab, sum	xi: