Applications of Data Analysis (EC969)

Simonetta Longhi and Alita Nandi (ISER)
Contact: slonghi and anandi; @essex.ac.uk

Week 1 Lecture 1: Inspecting and Manipulating Cross-section Data

---

Use data on individual respondents from the first wave of the British Household Panel Survey (BHPS). The file is called **`aindresp.dta`**.

1. Upload the data
2. Inspect the data
3. Create variables, recode values, retain and delete variables and observations
4. Estimate cross-sectional wage regressions, test the regression coefficients, save the results in output tables

---

Upload the Data

There are different ways to upload the data in Stata. You can do it interactively using the drop-down menu (file → open) or by clicking on the 'open' icon. Note that this works only if the data is in Stata format (.dta).
However, it is always best to write down the appropriate command in a do-file. The command in this case is **`use`**. Again, the command **`use`** works only for datasets that are already in Stata format, and can be used in two different versions.
If you want to upload the **whole dataset** (this option is the best e.g. if you do not know the name of the variables, or are not sure which variables you might need for your analysis):

```
use filename [, clear nolabel]
```

in our case:

```
use $dirdata/aindresp, clear
```

We use the dollar symbol ($) to tell Stata that **`dirdata`** is a global macro, and that its value is specified in that macro variable whose name follows the dollar. Remember, this is the macro we normally define at the beginning of our do file.
The option **`clear`** tells Stata to clear the memory of any data currently open, and to open the specified file. If a dataset is already open and you do not type **`clear`** (either as stand-alone command or as option to **`use`**) you will get an error message. Stata will not open the **`aindresp`** file because the data currently open will be lost.

If the dataset is very large and you know in advance which variables you need for your analysis, you can upload only a **subset of the dataset**. The advantage is that the file will upload more quickly:

```
use [varlist] [if] [in] using filename [, clear nolabel]
```

in our case:

```
use ahid pid asex amastat alknbrd apaygu ///
      using $dirdata/aindresp, clear
```

where **[varlist]** is a list of the variables you want to upload. If you want to upload only data for men, you could add **if asex == 1** after the list of variables. Note that after **if** we identify equalities by typing the equality sign twice: '**==**' instead of only once '**=**'. For more information look for 'logical operators' in the Stata help.

What if I want to upload a file which is not in Stata format (.dta)?

You can go to the drop-down menu: File → Import → ASCII data created by a spreadsheet.
Better, use the command **insheet**, which reads a dataset that is not in Stata format, e.g. data that have been generated by a spreadsheet or a database program. Regardless of the creator of the file, insheet reads text (ASCII) files in which there is one observation per line and the values are separated by tabs or commas.

```
insheet [varlist] using filename [, options]
```

filename can include the extension of the file. If filename is specified without an extension, **.raw** is assumed. If the path to your file or your filename contain spaces, remember to enclose the whole path/file name in double quotes.

Among the options you can choose:

tab	for tab-delimited data
comma	for comma-delimited data
delimiter("char")	if the data use char as delimiter
case	to preserve variable name's case
[no]names	if variable names are included on the first line of the file (note that [no]names is not shown in the dialog box)

In some cases the command **insheet** might not do what you are looking for. If this is the case, you might want to use the command **infile** instead. Check the Stata help for more details. If you want to save your data in spreadsheet-style format, you can use the command **outsheet** (or **outfile**). Files stored in this format usually require less disk space.

Inspect and Recode the Data

Let's now inspect the **aindresp** file. Open the data set, reading in only the variables of interest: **ahid pid asex amastat aage aqfachi alknbrd apagyu**. Use the global macro **dirdata** to specify the file path to the data set.

Then describe the data:
– list the first 20 observations;
– describe the data to find out what variables are contained in the dataset;
– summarise the data to get information on means and sample sizes:

```
use ahid pid asex amastat aage aqfachi alknbrd apaygu ///
   using $dirdata/aindresp, clear

list in 1/20
describe
sum
```

Looking at the output of the commands **list** and **sum** you will notice that some numeric variables have attached *value labels* to each numeric value. Use the command **label list** to list the names and contents of the value labels. If you want to focus on one variable only (e.g. **amastat**), try tabulating it with and without labels:

```
tab alknbrd
tab alknbrd, nolabel

sum alknbrd, detail
inspect alknbrd
```

How can we identify missing values in the BHPS? How does Stata identifies missing values?

NB: As in most datasets, in the BHPS negative values identify different types of missing values.
Often we are not interested in the reason why the variable is missing; if this is the case, we can recode all the negative values into (Stata) missings. There are different ways to recode missing values:

```
recode alknbrd -9/-1 = .
// This is one option
mvdecode apaygu, mv(-9/-1)
// This is another option
```

mvdecode decodes the missing values specified in parantheses, in this case all values between -9 and -1, and sets them to system missing. System missing values in Stata are represented by a full stop (.).

Remember:
`///` indicates that the command continues on the next line. You will use `///` in a Stata do file, but not if you are working interactively. If you are working interactively you will need to type the whole command in the same line.

How does the proportion of those who like/dislike their neighbourhood vary by gender? (Hint: you can use the **`tab`** command to tabulate up to two variables simultaneously).

Note that you can also ask Stata to add percentages to the table. The percentages can be computed by row or by column. You can also tabulate missing values!
Compare the different options:

```
tab alknbrd asex
tab alknbrd asex, col
tab alknbrd asex, row
tab alknbrd asex, col miss
```

How do they differ?

The name of the variable **`alknbrd`** is difficult to remember. Let's change it to something easier to remember:

```
rename alknbrd LikesNeighbourhood
```

Also change the name of the variable **`asex`** to **`sex`**.

Generate and Label New Variables

The variable **`apaygu`** is a continuous variable. In some cases we might want to summarise pay into few groups. For example, let's create a new variable, called **`newpay`**, that has value 1 (which we will call 'low') if **`apaygu`** is less than 200; 2 ('medium') if it is between 201 and 400; and 3 ('high') if it is more than 400.

```
generate newpay = 1 if apaygu <= 200
* equivalent to:  g newpay = 1 if apaygu <= 200
replace newpay = 2 if apaygu > 200 & apaygu <= 400
replace newpay = 3 if apaygu > 400 & apaygu ~= .
```

Note that Stata considers missing values as the highest values. Hence, if we only type **`replace newpay = 3 if apaygu > 400`** then when **`apaygu`** is missing **`newpay`** would have value 3. By adding **`& apaygu ~= .`** we make sure that missing values in **`apaygu`** are missing also in **`newpay`**.

An alternative way to generate the new variable (let's call it now **`newpay2`**) consists in recoding **`apaygu`**:

```
recode apaygu (0/200 = 1) (200/400 = 2) (400/max = 3), ///
      gen(newpay2) test
```

The option **`test`** tells us if there are overlaps between the new group formed. Compare with:

```
recode apaygu (0/200 = 1) (200.01/400 = 2) ///
      (400.01/max = 3), gen(newpay3) test
tab newpay3
```

Note that while recoding you can also attach labels to the new values. Try:

```
recode apaygu (0/200 = 1 low) (200.01/400 = 2 medium) ///
      (400.01/max = 3 high), gen(newpay4) test
tab newpay4
```

How can we attach labels to the values of the variable **`newpay`** (which we have created using the command **`generate`**)?
First we have to define the list of labels with the command **`label define`**; then we attach these labels to the values of the variable with the command **`label value`**:

```
tab newpay
label define newpay_labels 1 "low" 2 "medium" 3 "high"
label value newpay newpay_labels
tab newpay
label list newpay_labels
```

Note that in most cases in the BHPS the label name coincides with the name of the variable. After you have defined the label you can attach it to many variables.
Finally, we can attach a short description to the variable using the command **`label var`**:

```
label var newpay "Whether pay is low, medium, or high"
```

A useful extension to the command **`generate`** is the command **`egen`**, which allows more complex computations. We can use the command **`egen`** to compute means, medians, standard deviations, total sums, running sums, and much more. For example, we can easily compute the average pay:

```
egen MeanPay = mean(apaygu)
```

We could easily compute mean pay by gender by adding **`bysort`** at the beginning of the command (we will use this later in the course). Alternatively, we can compute

mean pay for men and women separately, using **`if`**. Let's call these two new variables **`MeanPayMen`** and **`MeanPayWomen`**.

Write the commands in your do file (and compare with the one at the end of this worksheet)

Note that **`MeanPayMen`** is missing for women, and **`MeanPayWomen`** is missing for men. Now combine the two variables (**`MeanPayMen`** and **`MeanPayWomen`**) into one single variable which contains the mean pay for the gender of the respondent. (Hint: use the commands **`generate`** and **`replace`**) Let's call this variable **`MeanPay2`**

Write the commands in your do file (and compare with the one at the end of this worksheet)

If we had used **`bysort`** we would have obtained the same result in only one step.

Let's now create two dummy variables: the first one (**`Men`**) is 1 for men and 0 for women; the second one (**`Women`**) is 1 for women and 0 for men. (Hint: use the commands **`generate`** and **`replace`**)

Write the commands in your do file (and compare with the one at the end of this worksheet)

There is a faster way to generate such kinds of dummy variables:

```
tab sex, gen(Sex)
```

This command tabulates the variable **`sex`**, and generates two new variables: **`Sex1`**, which is the same as our variable **`Men`**, and **`Sex2`**, which is the same as our variable **`Women`**. Try tabulating **`Men`** and **`Sex1`**; and **`Women`** and **`Sex2`**.

We have now a number of redundant variables: **`newpay, newpay2, newpay3, newpay4`** they are all the same. Let's eliminate **`newpay2, newpay3, newpay4`** (and some of the other duplicated variables) from the dataset:

```
drop newpay2 newpay3 newpay4 MeanPayMen MeanPayWomen Men Women
```

The command **`drop`** eliminates variables or observations. Note that if we type **`drop newpay*`** we also lose the variable **`newpay`**.
The command **`keep`** keeps observations or variables. Hence, if we want to eliminate from the data all variables except **`newpay sex MeanPay MeanPay2`** and those variable starting with **`Sex…`**, we can type:
`keep newpay sex Sex* MeanPay MeanPay2`.

If we are only interested in those cases where **`apaygu`** is not missing, we can drop from the dataset all those cases where the variable is missing using **`drop if`** or **`keep if`**:

```
drop if apaygu == .
      * drops obs with missing apaygu
```

same as

```
keep if apaygu ~= .
      * keeps obs where apaygu is not missing
```

Saving Datasets for Future Use

The command is called **save**:

```
save [filename] [, save_options]
```

where **filename** includes only the name of the file if you want to save it in the current directory; while it includes the whole path if you want to save it in a different directory. You can either type the path, or save it in a global macro, as we did for the path where the original dataset is stored.

Useful options are:

replace	overwrite existing dataset
emptyok	save dataset even if zero observations and zero variables

(There is no need to save the file you worked on today. Next time we will create the dataset that we will then use for analysis)

Estimating a Wage Regression on Cross-section Data

We want to estimate a wage regression of the kind:

$$\text{Log wage}_i = \alpha_0 + \alpha_1 \text{Age}_i + \alpha_2 \text{Age}_i \text{ square} + \alpha_3 \text{Female dummy}_i + \alpha_4 \text{Married dummy}_i + \alpha_5 \text{Qualification dummies}_i + \text{error term}_i$$

First, generate the relevant variables: age square; female, married, and qualification dummies. Then, generate the dependent variable: log wage. Note that the variable **apaygu** refers only to wages of employed people; earnings of self-employed people are stored in a different variable.

Write the commands in your do file (and compare with the one at the end of this worksheet)

We can estimate the wage regression using the command **regress**:

```
regress LnW age age2 Female Married Q1-Q5
```

We can compute different types of standard errors, for example, if you want to compute 'robust' standard errors, just include the option **robust**.

Other options are available. For example, if one of your variables is aggregate, like the unemployment rate at the regional level, you might want to use the option **`cluster(region)`**, which accounts for correlation of observations within the same region (note that **`region`** is the name of the variable we want to cluster by).

How do the results change if we use robust standard errors?

We can compute linear tests using the command: **`test`**.
Syntax 1: tests that coefficients are (not statistically different from) zero

```
test coeflist
```

Syntax 2: tests that linear expressions are equal

```
test exp=exp[=...]
```

Syntax 3: tests that coefficients in eqno are zero (useful when you estimate systems of equations using e.g. **`sureg`** or **`reg3`**)

```
test [eqno] [: varlist]
```

Syntax 4: tests equality of coefficients between equations (useful when you estimate systems of equations using e.g. **`sureg`** or **`reg3`**)

```
test [eqno=eqno[=...]] [: varlist]
```

Now look at the results of your regression with robust standard errors. Are women paid on average as much as men? And are married people paid as much as non married ones?

Does the wage impact of being married differ between men and women?

You can analyse this questions in two different ways: 1) by adding to your regression an interaction term between the two variables of interest; 2) by estimating your regression separately by gender. How does the answer to the previous question change depending on the model we estimate? Why?

Some of the explanatory variables in the wage regression might be endogenous (for example one or more of the explanatory variables might be correlated with the error term). Education might be endogenous to wages: it is plausible that unobserved factors such as ability have an influence both on the level of education and wages. More able people are more likely to reach higher levels of education, and more likely to obtain higher wages independently on their level of education.
In this case OLS would be inconsistent and a two-stage least square approach might be preferred, provided that you have good instruments.
To test for endogeneity:
1. Regress the variable which might be endogenous on all instruments (including the other non endogenous explanatory variables) and check that the instruments are good predictor for the dependent variable in this model
2. Obtain the residuals of the model
3. Include the residuals in the original equation. If the regression coefficient of the residuals is statistically significant, then the variable is endogenous.
4. Estimate your model using the **`ivreg`** command

Remember: if your dependent variable is not continuous, instead of using OLS (i.e. **`regress`**) it is better to use models for limited dependent variables, e.g. logit or probit models for binary, multinomial, ordered etc. variables. Stata allows an easy estimation of all these models with the commands: **`probit`**, **`mprobit`**, **`oprobit`**, **`logit`** etc.
Look at the Stata help for more information of these commands. We will deal with limited dependent variables – in the context of panel data – in the next weeks.

--- OPTIONAL ---
Producing Output Tables

Once the results of our models have been saved, we can produce output tables, which can be opened using a word processor. We can do this using the command **estout**.

```
estout Model1 Model2 using FileName.out, replace
```

Model1 and **Model2** are the names that we gave to the results of the models we have estimated. After estimating the regression we should type **estimates store** followed by the name we want to give to the estimation results, e.g. **Model1**. The command **estout** saves the estimated regression coefficients, their standard errors, and other statistics in an "out" file. Using Word you can easily convert this text to a properly formatted table (after selecting the text go to the menu Table → Convert → Text to Table; remember: in this process you should not select any note you might have added to your table, only select the coefficients, their standard errors and the R2, number of observations etc at the bottom of the table).

Useful options are:

keep(keeplist) keeps only the parameters of interest. If you don't specify this option, then all the regression coefficients will be saved.

cells(array|none) contents of the table cells. For example:
cells(b(star fmt(%9.3f))) specifies that we want to show only three decimals of the estimated coefficients, and we want Stata to include stars to identify the level of statistical significance. You can use the option **starlevels()** to specify how many stars (or other symbols) would identify which level of statistical significance.
cells(se(par fmt(%9.3f))) specifies that we want to show only three decimals of the estimated standard errors, and we want Stata to put them within parethesis.

stats(scalarlist[, subopts]) Displays summary statistics at the bottom of the table. For example, **N** would include the number of observations; **r2** would include the R-square; **r2_a** would show the adjusted R-square; **aic** would show the Akaike Information Criterion.

postfoot(stringlist) Adds text after the table footer

FINALLY, remember to close the log file at the end of the session:

```
log close
```

Do File

```
version 11
clear all
set more off
capture log close
set memory 20m

global dirdata "\\iserhome\conferencedata\final"
global dirresults " M:"

log using "$dirresults\Exercise.log", replace


*********************************************************************
* INSPECTING AND MANIPULATING BHPS DATA FOR WAVE 1
*********************************************************************


* Open the data *
*****************

use "$dirdata/aindresp", clear

describe
// Gives information on the variables contained in the dataset
sum
// Gives information about sample sizes, means, min, max

* Alternative method
use ahid pid asex amastat aage aqfachi alknbrd apaygu ///
      using "$dirdata\aindresp", clear


* Inspect and recode the data *
*******************************

list in 1/20
// Lists the first 20 observations

describe
sum

// The variables bsex, bpayuw have value labels, what does it mean?
// If you want to examime the values and labels of all variables:
codebook
labelbook

// If you want to focus on one variable only:
tab alknbrd
tab alknbrd, nolabel
sum alknbrd, detail
// Negative values (identify different types of) missing values
inspect alknbrd

// One way to recode missing values:
recode alknbrd -9/-1 = .
// Another way to recode missing values:
mvdecode apaygu, mv(-9/-1)
```

```
sum apaygu
sum apaygu, detail

sum alknbrd apaygu

* How does the proportion of those who like/dislike
* their neighbourhood vary by gender?
tab alknbrd asex
tab alknbrd asex, col
tab alknbrd asex, row

tab alknbrd asex, col miss

rename alknbrd LikesNeighbourhood
rename asex sex


* Generate and label new variables *
***********************************

* Generate the new variable newpay
generate newpay = 1 if apaygu <= 200
* equivalent to:  g newpay = 1 if apaygu <= 200
replace newpay = 2 if apaygu > 200 & apaygu <= 400
replace newpay = 3 if apaygu > 400 & apaygu ~= .

recode apaygu (0/200 = 1) (200/400 = 2) (400/max = 3), ///
      gen(newpay2) test
recode apaygu (0/200 = 1) (200.01/400 = 2) (400.01/max = 3), ///
      gen(newpay3) test
tab newpay3
recode apaygu (0/200 = 1 low) (200.01/400 = 2 medium) ///
      (400.01/max = 3 high), gen(newpay4) test
tab newpay4

* Label the values of the variable newpay
tab newpay
label define newpay_labels 1 "low" 2 "medium" 3 "high"
label value newpay newpay_labels
tab newpay
label list newpay_labels

egen MeanPay = mean(apaygu)
egen MeanPayMen = mean(apaygu) if sex == 1
egen MeanPayWomen = mean(apaygu) if sex == 2
generate MeanPay2 = MeanPayMen if sex == 1
replace MeanPay2 = MeanPayWomen if sex == 2

generate Men = 0 if sex == 2
replace Men = 1 if sex == 1
generate Women = 0 if sex == 1
replace Women = 1 if sex == 2
* Alternative way:
tab sex, gen(Sex)

drop newpay2 newpay3 newpay4 MeanPayMen MeanPayWomen Men Women

drop if apaygu == .
* drops cases when pay is missing; same as
* keep if apaygu ~= .
* keeps only observations where pay is not missing
```

```
* Generate variables for wage regression *
*****************************************

generate age2 = aage^2

generate Female = 1 if sex == 2
replace Female = 0 if sex == 1
label var Female "Dummy for women"

generate Married = 1 if amastat == 1 | amastat == 2
replace Married = 0 if amastat >= 3 & amastat <= 6
label var Married "Whether married or cohabiting"

recode aqfachi (-9/-1 = .) (1 = 2)
tab aqfachi, gen(Q)
label var Q1 "1st degree or higher"
label var Q2 "hnd,hnc,teaching"
label var Q3 "a level"
label var Q4 "o level"
label var Q5 "cse"
label var Q6 "none of these qualif"

generate LnW = ln(apaygu)


* WAGE REGRESSIONS AND TESTS *
*****************************

regress LnW aage age2 Female Married Q1-Q5

regress LnW aage age2 Female Married Q1-Q5, robust
estimates store Model1

* Are women paid as much as men?
test Female
* Are married people paid as much as non married ones?
test Married
* Note that in this case the test is not strictly necessary:
* In this case it is enough to look at the regression coefficients

* Does the impact of being married differ between men and women?
*---------------------------------------------------------------
* Method 1
*---------
generate MarriedWoman = Married * Female
regress LnW aage age2 Female Married MarriedWoman Q1-Q5, robust
estimates store Model2
* Method 2
*---------
regress LnW aage age2 Married Q1-Q5 if Female == 0, robust
regress LnW aage age2 Married Q1-Q5 if Female == 1, robust
```

```
* Producing estimation tables *
*******************************

estout Model1 Model2          ///
      using "$dirresults\TableREsults.out",     ///
      keep(Female Married)    ///
      cells(b(star fmt(%9.3f)) se(par fmt(%9.3f)))
      style(tab) stats(aic N r2_a, fmt(%9.3f %9.0g)   ///
      labels(AIC Observations)) label collabels(, none)     ///
      starlevels(* 0.10 ** 0.05 *** 0.01) ///
      postfoot("Robust standard errors in parenthesis") ///
      replace


log close
```