PROGRAM ON THE GLOBAL DEMOGRAPHY OF AGING

Working Paper Series

A Practical Introduction to Stata

Mark E. McGovern

August 2012

PGDA Working Paper No. 94
http://www.hsph.harvard.edu/pgda/working.htm

 The Program on the Global Demography of Aging receives funding from the National Institute on Aging, Grant No. 1 P30 AG024409-06.

A Practical Introduction to Stata*

Mark E. McGovern

Harvard Center for Population and Development Studies
Geary Institute and School of Economics, University College Dublin

August 2012

Abstract

This document provides an introduction to the use of Stata. It is designed to be an overview rather than a comprehensive guide, aimed at covering the basic tools necessary for econometric analysis. Topics covered include data management, graphing, regression analysis, binary outcomes, ordered and multinomial regression, time series and panel data. Stata commands are shown in the context of practical examples.

Contents

*I gratefully acknowledge funding from the Health Research Board. This document is based on notes for the UCD MA econometrics module and a two day course in the UCD School of Politics. Preliminary, comments welcome.
Email: mcgovern@hsph.harvard.edu

List of Tables

List of Figures

Objective

The aim of this document is to provide an introduction to Stata, and to describe the requirements necessary to undertake the basics of data management and analysis. This document is designed to complement rather than substitute for a comprehensive set of econometric notes, no advice on theory is intended. Although originally intended to accompany an econometrics course in UCD, the following may be of interest to anyone getting started with Stata. Topics covered fall under the following areas: data management, graphing, regression analysis, binary regression, ordered and multinomial regression, time series and panel data. Stata commands are shown in red. It is assumed the reader is using version 11, although this is generally not necessary to follow the commands.

1 Introduction

1.1 Opening Stata

Stata 11 is available on UCD computers by clicking on the "Networked Applications". Select the "Mathematics and Statistics" folder and Stata v11. It is also possible to run Stata from your own computer. Log into UCD connect and click "Software for U" on the main page. You will first need to download and install the client software, then you will be able to access Stata 11, again in the "Mathematics and Statistics" folder. For further details see http://www.ucd.ie/itservices/teachinglearningit/applications/softwareforu/d.en.21241

Stata 11 is recommended, however Stata 8.0 may also be available on the NAL (Novell Application Launcher). Click Start and open the NAL. Open the Specialist Applications folder and click into Economics. Open wsestata.exe, or right-click and add as a shortcut to your desktop. Alternatively, click Start > Run, paste in `Y:\nalapps\W95\STATASE\v8.0` and click enter.

1.2 Preliminaries

Before starting, we need to cover a very important principle of data analysis. It is vital that you keep track of any changes you make to data. There is nothing worse than not knowing how you arrived at a particular result, or accidentally making a silly mistake and then saving your data. This can lead to completely incorrect conclusions. For example you might confuse your values for male and female and conclude that men are more at risk of certain outcomes, etc. These mistakes are embarrassing at best, and career threatening at worst. There are three simple tips to avoid these problems. Firstly keep a log of everything. Secondly, to ensure you don't embed any mistakes you've made in future work, most econometricians never save their datasets. Generally people initially react badly to this suggestion. However you don't need to saves changes to the dataset itself if you implement all manipulations using do files. The final tip therefore, is to use do files. We will cover each of these in what follows.

The first thing we need to do is open our data. If we have a file saved somewhere on our hard disk we could use the menus to load it. FILE, OPEN. Or we could write out the full path for the file, e.g. "h:\Desktop\". The path for your desktop will differ depending on the computer your are using, however, if you are on a UCD machine this should be it. This is awkward, and we will also need somewhere to store results, and analysis. So we will create a new folder on our desktop called "Stata". Right click on your desktop, and select NEW, FOLDER. Rename this to "Stata". We will also create a new folder within this called "Ado" which we will use to install new commands. Save the files for this class into the "Stata" folder. Stata starts with a default working directory, but it is well hidden and not very convenient, so we want to change the working directory to our new folder. First we check the current working directory with `pwd`. Now we can change it `cd ''h:\Desktop\Stata''`. If you are unsure where your new "Stata" folder is, right click on it and go to PROPERTIES. You will see the path under LOCATION. Add "\Stata " to this. Now we can load our data files. One final piece of housekeeping, because we can only write to the personal drive ("h:\") on UCD computers we need to be able to install user written commands here. So we set this folder with `sysdir set PLUS ''h:\Desktop\Stata\Ado''`. This is only necessary if you are running Stata from a UCD computer.

Now we have this set up, accessing files saved in Stata format (.dta) is straightforward. `use icecream2`. If you make changes to the data, you will not be allowed to open another dataset without clearing Stata's memory first. `gen year=2010`. We will encounter the gen command later. Now if we try and load the data again `use icecream2` we get the error message "no; data in memory would be lost". We need to use the command `clear` first, then we can reload the dataset `use icecream2`. Alternatively, using the clear option automatically drops the dataset in current use `use icecream2, clear`. This raises a very important point, we need to keep track of our analysis and our changes to the data. Never ever save changes to a dataset. If you have no record of what you have done not only will *you* get lost and not be able to reproduce your results, neither will anyone else. And you won't be able to prove that you're not just making things up. This is where do files come in. A do file (not to be confused with an ado file)[1] is simply a list of commands

[1]This is a do file which contains a programme. Stata uses these to run most of its commands. This is also how we are able

that you wish to perform on your data. Instead of saving changes to the dataset, you will run the do file on the original data. You can add new commands to the do file as you progress in your analysis. This way you will always have a copy of the original data, you will always be able to reproduce your results exactly, as will anyone else who has the do file. You will also only need to make the same mistake once. The top journals require copies of both data and do files so that your analysis is available to all. It is not uncommon for people to find mistakes in the analysis of published papers. We will look at simple example. Do files have the suffix ".do". You can execute a do file like this `do intro`. [2] `do tutorial1` would run all of the analysis for this particular tutorial. There are several ways to open, view and edit do files. The first is through Stata. Using the menus go to WINDOW DO-FILE EDITOR, NEW DO-FILE. Or click on the notepad icon below the menus. Or type `doedit` in the command window. Or press CTRL F8. Each of these will open the dofile editor. Alternatively you can write do files in notepad or word. They must be saved as .do files however. You don't have to execute a whole do file, you can also copy and paste commands into the command window. Here we will create our own do file using the commands in this document.

As well as using do-files to keep track of your analysis, it is important to keep a log (a record of all commands and output) in case Stata or your computer crashes during a session. Therefore you should open a log at the start of every session. `log using newlog, replace`. To examine the contents of a log using the menus go to FILE, VIEW. Alternatively type `view logexample`. Also useful is `set more off`, which allows Stata to give as much output as it wants. This setting is optional but otherwise Stata will give only one page of output at a time. Finally, you must have enough memory to use your data. You can set the amount of memory Stata uses. By default, it starts out with 10 megabytes which is not always enough. If you run out of memory you will get the error message "no room to add more observations". For most data files 30 megabytes will be enough, so we will start by setting this as the memory allocation. `set mem 30m`. To check the current memory usage type `memory`. You could set memory to several hundred megabytes to ensure that Stata will never run out, but this makes your computer slow (especially if you have a slow computer) and so is not recommend. None of the files we will be examining require more than this. Note that if you run out of memory you will have to clear your data, set the memory higher and re-run your analysis before proceeding.

In general all of these items are things you will want to place at the start of every do file.

```
clear
set mem 30m
cd  "h:\Desktop\Stata"
sysdir set PLUS "h:\Desktop\Stata\Ado"
set more off
capture log close
local x=c(current_date)
log using "h:\Desktop\Stata\`x'", append
```

Lines 7 and 8 require some explanation. The outcome of this is that Stata will record all analysis you conduct on a particular day in a log file, the name of which will be that day's date. We will explain how this works when we discuss macros. Note that Stata ignores lines with begin with "*", so we will use this to write comments. The command "capture" is also important. If you are running a do file and it encouters and error, the analysis will stop. The "capture" command tells Stata to proceed even if it encounters a mistake.

If you are running Stata on your own computer, there is a way to alter the default settings that Stata starts with. When it launches, Stata looks for a do file called "profile.do" and runs any commands it contains. You can create this file so that these changes are made automatically every time you launch Stata. (i.e. memory is set, directory is set and a log is started). As well as a working directory, Stata also has other directories where programmes are stored. We need to put our "profile.do" into the "personal" folder. To find it, type `sysdir`. We now paste the following into a text file (either using notepad or Stata), and save it as "profile.do" into that directory.

to install new user written commands. Usually we will be able to install these automatically, however sometimes we need to do this manually. All that is involved here is saving the appropriate ado file into the appropriate directory which you can locate with `sysdir`.

[2] `run intro` executes the do file but suppresses any output.

1.3 Audit Trails

1. Remember to keep a record of everything
2. Never alter the original dataset
   (a) Place the original dataset in a separate folder
   (b) Make a backup of the dataset
   (c) Use .dta files for Stata use
3. Before completion, do a test run on a backup

1.4 Getting Help

Stata has an inbuilt help function. In fact you can easily access the help file for any command. Suppose we are interested in the "tabstat" command, we can type `help tabstat`. However these are often aimed at experienced users and you may have difficulty understanding them. The syntax for this command is given as "tabstat varlist [if] [in] [weight] [, options]". Items in square brackets are optional. So all we require to run this command is the command itself followed by at least one variable. The various options are explained, followed by some examples. Looking at the examples is often the best way of getting to grips with how a command works. If you cannot solve the problem using the help file, Stata has an extensive online support system. It is more than likely that someone else has encountered the same problem. A google search usually throws up several items of interest. There are also several excellent websites which detail how to deal with various aspects of data analysis. The best of these are the UCLA (http://www.ats.ucla.edu/stat/stata/sk/default.htm) and Princeton

(http://dss.princeton.edu/online_help/stats_packages/stata/stata.htm) Stata websites. We are now ready to begin analysing our data.

1.5 Importing Data

We have already seen that importing data in Stata format (.dta files) is simple. The command is "use filename". We need to load the icecream dataset for this analysis. `use icecream`.

> **Note:** We're opening a dataset. Remember, do not save any changes you make.

Often you will have to make do with Microsoft Excel (.xls) files. Fortunately Stata can import these quite easily. If you have an Excel file named `myfile.xls`, you can import it using Stata's `insheet` command. First, in Microsoft Excel, click File > Save As. Now instead of saving as a Microsoft Office Excel file, save the file as a CSV (Comma Delimited) file using the dropdown menu. You can then load the data using the command `insheet using myfile.csv`. The "infile" command is an alternative for loading other types of data. If a direct import with this command fails, try opening it in excel and following the instructions above. We will discuss how to import SPSS files when discussing an example of a user written command.

1.6 User Written Commands

One of the major advantages of Stata is the manner in which users can write their own commands. In the unlikely event that you are trying to do something that doesn't have an official command, you are practically guaranteed that someone else has had the same problem, and can be reasonably confident that someone has written their own code to deal with the issue. Finding the answer to your particular problem is not always straightforward, but it can be as easy as a google search. Knowing exactly what to look for can be the main problem. If you can find the name of your programme it can be relatively straightforward to install. Later we will encounter two other user written commands ("tabout" and "catplot"), now we will consider the

programme "usespss" which is used to import data saved in SPSS format.[3]. If we try to import the SPSS file in our directory, `use spssfile.sav` we get the error message "file SPSSfile.sav not Stata format". You would imagine that it should be relatively easy to transfer files between different statistics packages but this is not the case. Without this command you might need to use the expensive StatTransfer programme. As we know what we're looking for, the process of installing the programme is easy. We use the "findit" command. `findit usespss`. In fact if you're sure of the name you can simply type `ssc install usespss`. If you are not exactly sure of the name, but have a general idea of what it's called you can use the command `search usespss, all`. A new window will appear, and clicking on the blue link will take you to a new page. Click on CLICK HERE to install. This programme is now ready to run. You can also access the help file, `help usespss`. We can now open the SPSS file in our folder. `usespss using spssfile.sav`. We could now save this in Stata format for future use. `save, replace`. It is saved as "SPSSfile.dta". However as you can see, this is just the same as our icecream dataset.[4] We will re-load the Stata version. `use icecream2, clear`.

1.7 Menus and Command Window

Across the top of the Stata programme are a number of menus, and you may be tempted to use these to carry out your analysis. For example, going to 'FILE', 'OPEN', and selecting your file will open the dataset. As we will explain later on, this is not ideal. Not only is it slow, but it makes reproducing your results very tricky. The alternative is to type commands directly into the command window, which requires becoming familiar with the language of Stata, but is ultimately much more efficient and reliable. One benefit of using the menus is that when you run a command this way, it appears in the main window, which is where results of analysis are displayed. So this is a useful way to learn commands if you are unsure of them. Copying this from the main window into the command window and pressing enter will reproduce the command. This is in fact how to keep track of your analysis, by recording every change you make to the data. The Review window provides you with a list of all entered commands. Clicking on a command in the review window will cause it to appear in the command window. Failed commands are shown in red. The variable window gives a list of all the variables in the dataset, their labels, and their attributes.

1.8 Data browser and editor

Stata holds the data in memory like an excel file. To see the actual data select 'DATA' then 'DATA BROWSER' from the menus. You will see the variables ordered horizontally with the observations ordered vertically. Selecting 'DATA EDITOR' instead provides you with the same view, except you are now able to edit the data, like you would a spreadsheet. In fact this is an alternative way of entering data, you can simply paste it in. As before this is not recommended for reasons of reproducibility. Clicking on a particular case gives you the exact entry for that particular variable and observation. The data editor and browser windows must be closed before you can enter any new commands into Stata.

1.9 Syntax

Getting to grips with how to communicate with Stata is perhaps the most daunting aspect of starting out. Generally programmes and commands take the form of "command name" "variable name(s)" ", options." We will shortly see examples with tab and later regress. The exact syntax for a particular command is detailed in the help file. For example, `help tab`. Here the aim is to introduce you to some of the most important commands. As you become more familiar with them you will be able to use the various options available, depending on the particular task you wish to perform.

Stata understands abbreviations, once the abbreviation can only be interpreted one way. For example, the full command to run a regression is "regress", however Stata understands what you mean if you only type "reg". The same principle applies to variable names. Using the icecream dataset, typing `tab ti` is equivalent

[3] SPSS is a statistics package popular in the other social sciences

[4] An alternative for transferring SPSS files into Stata is to download SPSS which is available from UCD Connect, open the SPSS file and save in Stata format.

Table 1: Logical Operators in Stata

And	&
Or	|
Not	! or ~
Multiplication	*
Division	\
Addition	+
Subtraction	-
Less Than	<
Greater Than	>
Less Than or Equal	=<
More Than or Equal	=>
To The Power Of	^
Wildcard	*

to `tab time`. However, the command `tab t` will return the error message "t ambiguous abbreviation." In the help file for a command, the shortest acceptable abbreviated version is underlined.

1.10 Types of Variables

There are essentially two types of variables in Stata, words (referred to as strings) and numbers. Each is handled slightly differently when you are manipulating your data. Within numerical variables, there are two further types; continuous data, such as income or height, and categorical data (such as level of education or gender). In the second case a value will take on a particular meaning, e.g. 1=male and 2=female. As we will see these variables are often labelled in the data. In the Icecream dataset, the first 5 are obviously continuous numerical variables (these appear in black in the data browser), however county, var7 and weekend are different. County is a string variable, with the entries appearing as words in the dataset. Strings such as this appear in red. If you click on the county variable for observation 1, you will see "Antrim" appear as the entry for that case. On the other hand, for var7 the entry for observation 1 appears as "Ulster" in the data browser, but "1" when you click on it. This means the data is in numerical form (in this case 1-4), but has been labelled so that each number refers to a different province. We will discuss manipulating each of these types of variables in the next section.

2 Data Manipulation

Here we introduce the basic commands for manipulating your data. The most important logical operators in Stata are outlined in table 1. The most frequently used are & (and), | (or) and ! (not). These are essential for manipulating the data correctly. We can illustrate some of these using the "display" command, which we can shorten to "di": `di 10*20` and `di 6/(5-2) +18`. Notice that strings require double quotes `di Welcome` does not work but `di ``Welcome''` does. You can also access system values and programme results with this command, for example today's date `di ```c(current_date)'''`. Note again that `di `c(current_date)'` returns an error message. We need single quotes because "c(current date)" is a macro. This is how we were able to name our log file, see section 2.10 for more details on macros. We will explain the use of the wildcard in section 2.9. Stata will know whether you mean multiplication or the wildcard depending on the situation.

2.1 Describing Data

Stata provides several ways of investigating and describing data. It is generally a good idea to `browse` the data once you have it loaded. This allows you to view the data in spreadsheet format. For a broader look at the variables in the dataset, use the `sum` command, which summarises all the variables. Alternatively, specifying a variable after the command (e.g. `sum price`) will summarize the price variable on its own. If you would like more precise information (e.g. percentiles) then you can add the `detail` option to the end of that command, i.e. `summ price, detail`. A very useful command is `inspect`. This summarizes a variable's missing values (if any) and provides a simple plot of the variable's distribution.

Sometimes you may wish to break down summaries by a particular variable. For example, you might like to see how some variable changes over time. The `bysort` command is very useful here. It is best explained with an example. Suppose you want to see how consumption changes as temperature changes. The command here would be: `bysort temp: summ cons`. This means, in English, "summarize consumption for every distinct value of temperature." Other variables that you may use again include `describe` and `list`.

To investigate basic correlations, use the `corr` command followed by the variables you want to correlate. For example, `corr price temp cons` will provide a matrix of Pearson correlation coefficients between price, temperature and consumption. You may want to use the `pwcorr` command instead. It is essentially the same as `corr` but it allows for a little bit more detail. If you type `pwcorr temp cons, sig` it will provide p-values of the test with the null of the correlation being zero.

2.2 Generating Variables

You will regularly have to generate variables to aid econometric analysis. For example, you may want to create dummy variables or run log-log regressions. To create a variable named `loggedprice` equal to the natural log of price, the command is `gen loggedprice = ln(price)`. Similarly, to generate a variable equal to twice the square root of temperature, use the command `gen twice_root_temp = 2 *sqrt(temp)`. Note that variable names cannot contain spaces.

The `egen` ("extended gen") command works just like `gen` but with extra options. For example, `egen avg_price = mean(price)`. With `egen` we can also break commands down into percentiles very easily. For example, to create a variable equal to the 99th percentile of price, enter `egen high_price = pctile(price), p(99)`. Changing the 99 to 50 in that command would produce a variable equal to the median price. "egen" is often used to obtain a breakdown of a particular statistic by another variable. For example, we could obtain a variable containing the minimum income value for that particular province `egen minincome=min(income), by(var7)`.

2.3 `if` Commands

Oftentimes we want Stata to run a command conditional on some requirement. For example, the correlation between price and consumption if the temperature is greater than 25°C. This is easily achieved: `corr price cons if temp>25`. To add more conditions to a command, for example to examine the correlation

if temperature is both greater than 25°C *and* less than 35°C, we use the & operator: `corr price cons if temp>25 & temp<35`.

We can also use `if` to investigate data more closely: `summ cons if price >.275`. We can create dummy variables with `if` commands. Typically two steps are needed. First we create a variable set equal to zeros: `gen expensive = 0`. Now we `replace` it: `replace expensive = 1 if price >avg_price`. See the tutorial on regression analysis for more details on dummy variables. Similarly we can control for outliers using `if` commands. For example if you want to eliminate the most expensive 5% of observations, the following would work:

```
egen top_fivepercent_prices = pctile(price), p(95)
drop if price > top_fivepercent_prices
```

We remove these variables from the data with the "drop" command[5] as we do not need them in this analysis. `drop loggedprice twice_root_temp avg_price high_price expensive`.

2.4 Summarising with tab and tabstat

One of the first things you will want to do with your data is to summarise its main features. Crosstabs are a useful pre-regression tool, and are also useful for presenting the main points of your data succinctly. The two most important commands for this are "tab" and "tabstat". "Tab" tells you how many times each answer is given in response to a particular variable. This is only suitable for variables with relatively few entries, such as categorical data. If you try to use "tab" on a variable which has hundreds of different entries you will get an error message. Typing `tab var7` will show the how many entries there are for each province. As with all commands, it can also be accessed through the menus via: STATISTICS, SUMMARIES, TABLES. It is also easy to obtain crosstabs which give a breakdown of a variable by another. For example typing `tab var7 weekend` will show how many weekend and weekday entries there are for each province. It is often useful to know the percentages as well as the actual numbers. We need to add an option to our "tab" command. `tab var7 weekend, col` will give us the percentage with each column. Typing `tab var7 weekend, row` will give us the percentage within each row.

The second command which is useful here is "tabstat". This is used for continuous variables, and its main use is to provide mean values. For example `tabstat price` will give the average price in the dataset. Using the command options we can also access other useful statistics such as the median `tabstat price, stats(med)`, or the variance `tabstat price, stats(var)`. For a full list of the available statistics, type `help tabstat`. As before we can obtain these statistics according to different levels of a second variable. `tabstat price, by(var7)` gives the average price for each province.

2.5 Introduction to Labels

Labels are designed to help the user of a dataset understand and present their findings. These are often essential, for example if you have a categorical variable gender with two values 1 and 2, and no label you are in trouble as you will not know which refers to male and which to female. Generally these will be provided in your data, but not necessarily. Often the variable name itself will be self explanatory, for example in the icecream dataset the meaning of the variable "income" is obvious, although we do not know the unit it is measured in. It is not so obvious for var7, but a look at descriptives or the data browser makes it clear that it refers to Irish provinces. It is easy to rename a variable. Type `rename var7 province` into the command window.

As well as their actual names, you can also label a variable. These can be used to provide additional information that is not apparent from the variable name. So far none of the variables in the dataset have these. Adding a variable label to a variable is also straightforward. Type `label variable temp ''Temperature Degrees C''` and `label variable cons ''The number of ice-creams purchased.''`. You can see the

[5] We can also select the variables we wish to remain in the data, with the "keep" command.

result of this in the variable window where this label will appear. The label will also be used in tables and other output generated by Stata.

We have already encountered the other type of labels, value labels. These are labels attached to particular values of a variable and are mainly used with categorical data. In the case of the variable province, we have already seen how this works. For a quick way of seeing exactly which labels are attached to each value, type `codebook province` to obtain the name of the value label. Then `lablebook province1` will show all the values and their labels. From this we can see that in the variable province, the value "1" is labelled with the name "Ulster", 2 "Leinster" etc. If there are many value labels you may need to use the option all. `codebook province, all`.

You will often need to either create your own value labels, or else modify existing ones. The procedure is almost the same in both cases. If you are starting from scratch, you will need to pick a name for your label. In this case we will create value labels for the weekend variable, and call the label weekend1. Suppose we know that "1" refers to "weekend" and "2" refers to "weekday". We use the "label define" command, followed by the values and their labels in double quotation marks. `label define weekend1 1''Weekend'' 2''Weekday''`. We then need to attach our value label to the existing variable using `label values weekend weekend1`. We `tab weekend` to confirm the change. The only difference in the case of modifying an existing set of values labels is that you need to obtain the name of the value label ("codebook" will supply you with this). Then you use the "label define" command with the "modify" option to change one or more of the labels. You do not need to reattach the modified value label to the variable, this is done automatically. You also do not need to write out the full set of labels, only the one you want to change. For example, if we wanted to change the label on the province variable for the value "2" from "Leinster" to "Dublin", we would use `label define prov 2''Dublin'', modify`.

2.6 Joining Datasets

There are two different situations when you will need to join datasets, each requires a different command. Broadly these involve adding more observations, or adding more variables. We will demonstrate each using our icecream dataset. For the first situation we will add two extra observations. In the second case we will add an extra variable (rainfall).

The first is easier. Suppose you want to join two rounds of a survey into a single dataset. This often happens with the likes of the European Social Survey, the QNHS or other surveys which involve repeated cross sections. If you think about what you would do in excel, all you need to do in this case is stack the two datasets on top of each other. But do not do it in excel because there will be no record of what you have done or how you have done it. We will use the "append" command to add our new observations which are in the dataset icecream3. It is as straightforward as `append using icecream3`. Now we have a dataset with extra observations. We confirm this with `tab county` and the data browser.

Adding new variables is slightly more tricky. We can't just add the information at random, we need to make sure that each observation is matched in each dataset. This often arises in the context of surveys such as SHARE which have several modules (and datafiles) dealing with different domains such as health, income, demographics etc. If we want to join modules we have to make sure that the extra variables are really the responses which that particular individual gave. So you need a variable in each dataset that uniquely identifies each individual (or firm, country etc. whatever your level of observation is).[6] In this case we will assume we are interested in time periods, and add an extra variable (rainfall) for each time period. In this case our unique identifier is time, and we will need this variable to be the same in the new dataset. The first thing we need to do is make sure the data is ordered correctly, so we need to sort by our identifier. `sort time`. The new dataset will also need to be sorted by this variable. Now we can use the merge command with the dataset containing the new variable icecream4. `merge time using icecream4`. We can now check that we have an additional rainfall variable for each time period. `tabstat rainfall, by(time)`. And in the data browser. A new variable, "_merge" is created, which tells us whether the variables were present in

[6] In certain situations you may need to merge across more than one variable, for example merge mergeid year if you have a panel dataset.

the master dataset, the dataset we merged with , or both. See `help merge` for details. For no we will drop this variable with `drop _merge`.

2.7 Tabout

We have already discussed tabulating variables and looking at crosstabs. Here we will examine how to extract these results in a way that can be easily used in presentations or papers. To do this we need the user written command "tabout". To find and install, we first search for it. `search tabout, all`. The option all allows us to search the internet. We find an entry in blue for tabout, with the description:

'TABOUT': module to export publication quality cross-tabulations / tabout is a table building program for oneway and twoway / tables of frequencies and percentages, and for summary tables. It / produces publication quality tables for export to a text file.

2.7.1 Tabout with Stata 9/10/11

You can install tabout from this link, or else by typing `ssc install tabout`.

As before, there are essentially two cases involved, one analogous to "tab", and the other analogous to "tabstat". `tabout county using filename.xls, replace` gives us an excel table with the numbers in each county. The replace option is important (and an option which is available in most programmes which involve exporting results) and tells Stata to overwrite the file if it already exists. The alternative is append, which tells Stata to add new results to the same file. If we want percentages as well as numbers in each category we can use the following option: `tabout county using filename.xls, append cells(freq co)`. We may also want to display crosstabs which we can do by adding a second variable, for example `tabout county province using filename.xls, append`. If we want percentages as well as numbers we can use the following: `tabout county province using filename.xls, append cells(freq co)`. "Tabout" only works with variables which have labels, as these kind of tables do not really make sense for continuous variables such as income. However, we can make tables of means (or other statistics) like with "tabstat". `tabout province using filename.xls, append cells(mean income) sum` gives us the average income for each province.

2.7.2 Tabout with Stata 8

Clicking on the link after running `search tabout, all` will open a page which allows you to install the programme. You will notice that Stata version 9 is required. We need a version which is compatible with the verision of Stata we are running. Here we need "tabout8", but a search is unsuccessful. `search tabout8, all`. We will need to install it ourselves. If we google "tabout8" we will find the website "http://www.ianwatson.com.au/stata.html" as the first entry. Here we find a link to the .ado file, which we download and save in our personal directory which we can locate with the command `sysdir`. We also save the corresponding help file in this directory. This may seem like a lot of effort, however this only needs to be done once, and will save you a lot of time if you are handling datasets with large numbers of variables. It is possible to copy and paste from the Stata window into excel after using the "tab command", but this is a painstaking process, especially if you are constantly updating your analysis. "Tabout" automates this process.

If we want percentages as well as numbers in each category we can use the following option: `tabout8 county using filename.xls, append cells(fcount fper)`. We may also want to display crosstabs which we can do by adding a second variable, for example `tabout8 county province using filename.xls, append`. If we want percentages as well as numbers we can use the following: `tabout8 county province using filename.xls, append cells(double)`. "Tabout8" only works with variables which have labels, as these kind of tables do not really make sense for continuous variables such as income. However, we can make tables of means (or other statistics) like with "tabstat". `tabout8 province using filename.xls, append cells(mean income)` gives us the average income for each province.

Table 2: Tabout Example 1 - Crosstabs

	Province									
	Ulster		Dublin		Munster		Connacht		Total	
Weekend	Num	%	Num	%	Num	%	Num	%	Num	%
Weekend	5	55.6	7	58.3	3	60.0	3	50.0	18	56.3
Weekday	4	44.4	5	41.7	2	40.0	3	50.0	14	43.8
Total	9	100.0	12	100.0	5	100.0	6	100.0	32	100.0

Table 3: Tabout Example 2 - Variable Averages

	Mean Income
By Weekend	
Weekend	84
Weekday	85
Total	85
By Province	
Ulster	83
Dublin	85
Munster	85
Connacht	86
Total	85

"Tabout" is at its most powerful when used in conjunction with LaTeX, which is the software which was used to create this document. Note that the more recent version has a slightly different syntax, but the general idea is the same. You can also get confidence intervals using the latest version.

2.8 Recoding and Strings

Sometimes we may need to recode particular values in order to carry out our analysis. For example we have information on all four provinces, but we may only be interested in comparing Ulster to the other provinces. We will generate a new variable province2, which at first is exactly the same as our province variable. `gen province2=province`. Now we will recode the Ulster value to be equal to one, and the other provinces to be equal to zero. In case we've forgotten, `codebook province` and then `lablebook province1` will tell us the value labels. Ulster is already coded as 1, so we leave that as it is. There's more than one way to change the value for the other countries. We could use the replace command. Either of the following would do the job: `replace province2=0 if province>1` or `replace province2=0 if province2!=1` or `replace province2=0 if province2==2 | province2==3 | province2==4`. The recode command often accomplishes the same task with less effort. `recode province2 (2/4=0)`. The "/" tells Stata to recode values 2 through 4 to be equal to 0.[7] "Recode" is also important if you want to change values within a variable simultaneously. For example suppose we want to reverse the coding on the variable "province", we could do so with `recode province (1=4)(2=3)(3=2)(4=1)`. This would not be possible using the "replace" command. If you do this, don't forget to change the value labels. Now if we `tab province2` we see only two values as required. To be sure that we're not confused when we come back to the data later, or someone else is using it, we should label this new variable properly. We define a new label, `lab def prov2 0''Leinster Munster or Connacht'' 1''Ulster''`. Then attach this to the variable province2. `label values province2 prov2`. Now `tab province2` is self explanatory.

[7] An even quicker command would be `gen province2=(province==1)`.

Several issues arise with trying to manipulate string variables. For example, if we try `replace prov2=15 if county==Armagh`, we get an error message. This is because in general Stata requires strings to be surrounded with double quotations marks. `replace province2=15 if county==‘‘Armagh’’` works. But we should undo that change. `replace province2=1 if county==‘‘Armagh’’`. As we will see later on, single quotation marks have other uses. In general, dealing with strings can be tricky, for example, we cannot replace a string value with a numerical value. We also want to avoid having to type out a string every time we want to manipulate the data. Expressions involving >and <clearly don't work with strings. Fortunately, there is a command which converts string variables into categorical variables with the appropriate value labels (e.g. the province variable). Here we use this on the county variable. `encode county, gen(county2)`. Now we have a categorical numerical variable county2 which is appropriately labelled. We can check this in the data browser. `encode county, replace` would have given us almost the same result, except with the original county variable replaced rather than generating a new variable. There is a command, "decode" which reverses the process.

2.9 Missing Values

Missing values are practically unavoidable, particularly in micro surveys. Individuals may not know how to respond to a question, or may simply refuse. In well established surveys such as Share or Living in Ireland, these will be coded as some value such as "99", and be appropriately labelled. However this is not always the case. Often there will just be a blank entry. This appears in Stata as ".". Stata actually equates this with infinity, so if you try something like `replace var9=100 if var9>100` then your missing values will be (unintentionally) included in your recode. So let's reverse this `replace var9=.  if var9==100`. For most commands, observations which have missing values (for any of the variables which are involved in running that command) are excluded. This is particularly important to remember when we come to look at regressions. For example we know there are now 32 observations in our dataset. If we `tab income` we find a total of 32 as expected. If we look at var9 in the data browser we notice it contains some missing values. So if we `tab var9`, we will only see a total of 29. We notice that there are some entries labelled as "Refusal". Depending on the circumstances we may or may not want to exclude these from our analysis. To do this we need to find the value for "Refusal" with `codebook var9`. We find it is "99". To recode this as missing we could do as above `replace var9=.  if var9==99` or `recode var9 (99=.)`. An alternative is the "mvdecode" command. `mvdeocde var9, mv(99)`. An advantage of using this command is that we can simultaneously recode all missing values for several variables at the same time, e.g. `mvdecode time income weekend, mv(99)`. But of course in this case the variables time and income do not contain any observation with the value "99". It is also possible to reverse this process with the command "mvencode". You may also want to recode existing non-missing values as missing, for example to deal with outliers. One way to do this is `replace income=.  if income>100`. Again, an alternative is "mvencode". But here there are no observations with income greater than 100. Note that dealing with missing values is a very important topic in applied econometrics, and can have a major impact on your results. Earlier we mentioned the wildcard "*". Suppose we wanted to drop all variables with beginning with "var", we could type `drop var*`. Or if we wanted to recode missing values for all variables and the value "99", we could type `mvdecode *, mv(99)`.

2.10 Macros, Looping and Programming

An important thing to remember about Stata is that there is nearly always an easier and quicker way to do things, especially if you find yourself having to repeat the same task over and over. This could be recoding, generating variables etc. Stata has features designed to automate this processes, and as you become more familiar with the programme you will literally be able to save yourself hours (by using loops, for example).

We will start with macros. These are simply shortcuts which stand in for something else, and can be used to store everything from strings (words) to values, variable lists and results from programmes. We define a

macro with the "local" command, and access it using single quotation marks ‘’.[8][9] First we define a macro x which takes the numerical value 10. `local x 10`. Now every time we call the macro ‘x’ we have the value 10. To check this we type `di ‘x’`. We can now use this macro in expressions, for example `di 100-‘x’`. We can also use it to manipulate variables. `gen income2=income*‘x’`. We can also store words in macros. Suppose you wanted to add the word "icecream" to each variable name. You could type `rename price icecreamprice` and `rename cons icecreamscons` etc. To save time you could store the word "icecream" in a macro. `local y icecream`. Then `rename price ‘y’price` would give the same result. You may wonder as to how useful this is, and in these cases it is probably not particularly helpful. A better example is when we want to store a list of variables. Rather than typing out the whole list every time, we can save the variables in a macro. `local z price income temp`. Then suppose we wanted to recode all missing values in for all of these, instead of typing `mvdecode price income temp, mv(100)` we can type `mvdecode ‘z’, mv(100)`. This is a small dataset so it's not a particularly big deal here, but it's a different matter when you have 100s of variables.

Macros are also important for accessing results stored by programmes.[10] The macros saved by a programme are listed in the help file. For example, if we look at `help summarize`, we see the list for this command.[11] Suppose we are interested in constructing new versions of our variables which are in the form of a z score (standardised deviation from a variable's mean: $z_i = \frac{(\mu - x_i)}{sd(x)}$. We can see from the help file that the command "summarize" stores the two results we need, the mean and standard deviation in the macros "r(mean)" and "r(sd)" respectively.[12] We can use these to form our new variables. To access the stored results we need an "=" that we didn't when we were defining our own macros. First we run the command `sum time`. Then we define our macros `local a=r(mean)` and `local b=r(sd)`. To check we have the correct results `di ‘a’` and `di ‘b’`. Now we can generate our new variable. `gen ztime=(‘a’-time)/‘b’`. If we now `sum ztime`, we see that the mean is effectively zero (it's actually a very small number due to rounding), as it should be seeing as the average variation around a mean is zero by definition. The standard deviation is also as expected. We will later write our own programme which will allow us to transform all our variables in this way in a single line of code. Think how long it would take to do this in excel by hand.

Loops are another time saving device which employ the use of macros. For example in our icecream dataset, we notice from the data browser that the variable hour has been badly inputted. These should all be in the format of the 24 hour clock, however you can see that the final two zeros are missing from some of the entries. This makes analysis difficult, for example `sum hour` will give misleading results. In order to correct this we make use of a loop. This involves the forvalues command. Essentially we want to add two zeros to every entry less than 100.

```
forvalues i=1(1)24 {
replace hour=hour*100 if hour==‘i’
}
```

Notice the syntax, we are creating a macro "i" which will start at the value 1, execute the command for that value, move on to the next value ("2"), execute the command for that value etc until the loop ends. The first number refers to the starting value, the number in brackets is the increment, and the final number is the end value. We need a curly brace at the end of this line, the command on a separate line, and another curly brace again on a separate line. Executing this command will give us a well behaved variable with every entry in the same format. `sum hour`.

The other type of loop (the "foreach" command) is generally used when you want to perform the same task on a number of different variables. We will write our own programme to transform every variable into a z score. We will call it "zscore". We then use the foreach loop, with "i" being the macro that corresponds to

[8] Note that ‘ is the inward pointing single quotation mark, and is usually the button to the left of the number 1 on your keyboard. You may need to press it twice.

[9] There are also "global" macros which are rarely used.

[10] This is what we are doing at the beginning of our do file. Stata is told to access the date, and open a log file under that name.

[11] Type `mac list` to see which macros are currently in use.

[12] For more on accessing macros see `help extended_fcn`.

every individual variable we wish to transform. The macro "0" refers to the list of variables we are interested in, essentially everything we type after our new command.[13] Like forvalues, we need a curly brace at the end of this line, and each of our commands also on a separate line. For every variable in "0" (our variable list) we are running the sum command, obtaining the macros for the mean and standard deviation, and then generating a new variable which will have the prefix "zscore". We also summarise this new variable. After the final curly brace we need "end" to tell Stata the programme is finished.

```
capture program drop zscore
program define zscore
foreach i in `0' {
sum `i'
local m=r(mean)
local n=r(sd)
gen zscore`i'=(`m'-`i')/`n'
sum zscore`i'
}
end
```

We can now run the programme on whichever variables we are interested in. For example, `zscore time cons price`. The summary statistics confirm we have what we wanted. Note that if you try to call a new programme by the same name as an existing programme you get an error message. `program define zscore`. So if you want to modify an existing programme you will first need to drop it. `program drop zscore`. But if there is no program called zscore this will produce an error message. Hence the use of "capture" just like when we were opening our log file. Also, you will need to define your programme again each time you start a new Stata session, unless you save it as an ado file.[14]

2.11 Counting, sorting and ordering

It is possible to access individual observations with your data using subscripts. This takes the form of square brackets containing the observation number after the variable of interest. For example, `di time[2]` displays the value of the time variable for the second observation. This can be very important if you want to access information in the responses of other observations. For example you may have data on households and may want to use information provided by parents to analysis the outcomes of children. We will illustrate this with our province variable. Two special cases of subscripts are _n and _N. The latter is used to count the total number in each case. For example, `gen totalno=_N` gives us a new variable which is the total number of observations in the dataset. It is obviously the same for each observation. `tab totalno`. On the other hand, `gen totalno2=_n` tells us the ranking of each observation in the dataset, and runs from 1 to 32. `tab totalno2`. These are most useful when used with the by command.

We will generate two variables which gives us the total number in each province, and also the rank of each observation in each province. First we need to sort our data `sort province`. Then `by province: gen provinceno=_N`[15] and `bysort province: gen provinceno2=_n`.

We can use the data browser to confirm that this generated the variables we expected. In this case we have some province level data in var10, however it is only present for one observation in each province, and we need it in all observations.

This is a case where you may be tempted to use excel, but apart from the replication issue, this will simply not be possible if you have a datatset with 1000s of observations and hundreds of variables. Instead we can use subscripts and a loop to make the change:

```
forvalues i=1(1)10 {
by province: replace var10=var10[`i'] if var10==. & var10[`i']!=.}
```

[13] Within the macro "0", the macro "1" refers to the first variable, "2" the second etc.

[14] It so happens that Stata already has a way of creating standardised variables with `egen newvar=std(var)`.

[15] These two steps could be combined with "bysort": `bysort province: gen provinceno=_N`

Within each province we are simply replace var10 with the value of the first observation in each province, provided that variable is missing. We are then looping over ten values as there are 10 at most 10 observations. The variable is only replaced if there is a missing value for that observation. Another useful command in this context is gsort, which orders the variables according to the values of some other variable.

2.12 Reshaping Datasets

Sometimes data will be in the wrong "shape". This is difficult to explain without an example. If we open the file cyear `use cyear2`, and look in our data browser we will see that we have several variables that refer to repeated measures over a number of years. In fact there are three outcomes: u5m (under 5 mortality), gdppc (GDP Per Capita) and hivp (Proportion of Population infected with HIV/AIDS), and 9 years. Depending on the analysis you wish to conduct, this may be awkward with the data in this "shape". For example, tracking an indicator across time is difficult with this format. We can use the reshape command to transform the data into something more useable. The syntax is a little tricky, but the most important part is to identify our outcome variables, our time variable, and our country (or firm or individual) identifier. The command then takes the form "reshape" "outcome variables", i("identifier") j("time variable"). So in this case we have `reshape long u5m gdppc hivp, i(country) j(year)`. Now we can track an indicator across time, e.g. `tabstat gdppc, by(year)`. Of course we now essentially have a panel dataset, which is a whole other topic. See section 7 for more details.

2.13 Graphs

Like the tables we discussed above, graphs are a powerful tool for exploring, summarising and presenting your data. The basic graph commands for Stata are straightforward, however getting to grips with all the available options is tricky. It would be impossible to discuss all the different types of graph, however we will discuss the most common types. Like with tables we will divide graphs into two types, those that deal with continuous variables and those that deal with categorical data. We will load our icecream data again. `use icecream2, clear`.

For continuous data, the easiest way to visualise the relationship between two variables is to produce a scatterplot of them, e.g. `scatter cons temp`. If, instead, we want a graph of the line of best fit between the two variables, the relevant command is `graph twoway lfit cons temp`. We can also combine multiple plots in one graph using the `twoway` command. For example, try using `twoway (scatter cons temp) (lfit cons temp)`. We can add some complexities very easily. For example you may wish to add confidence interval "bands" around your line of best fit. To achieve this, use `lfitci` instead of `lfit`.[16] Stata also can produce several graphs in the one chart. For example we can create a 3x3 matrix of scatterplots with by inputting `graph matrix cons temp price, scheme(s1mono)`. To investigate the distribution of a single variable, we can create a histogram of it using the `histogram` command. [17] For example, `histogram temp`.

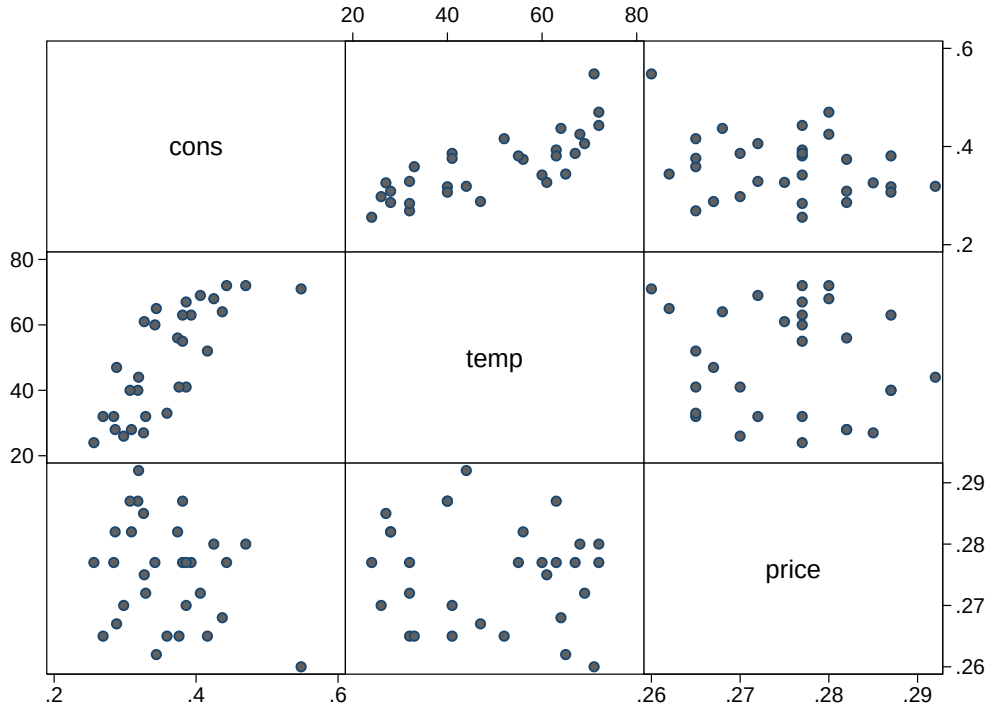
We can produce a bar chart showing the mean of our variables using `graph bar time cons price income temp`. We can display this breakdown for values of a particular variable in the same graph `graph bar time cons price income temp, by(province)` or in different graphs `graph bar time cons price income temp, over(province)`. We may also want to graph categorical variables like province, with the aim of showing the per cent in each category. The best way to do this is with the user written command "catplot". As before we use the search command `search catplot, all` or `ssc install catplot`, and click on the blue link to install. Now we can use this to graph our province variable by itself `catplot province`, or by another variable `catplot province weekend`.

We will discuss this further in the time series section, but if you have time series data then line graphs can be important. First we need to sort our data by the time variable, in this case time. `sort time`. Then we use the "line" command to graph the variables. The last variable needs to be our time variable. So in this case we could have `line cons income time`.

[16] Note that the easiest way to change the overall look of a graph is with schemes. See `help schemes`. We will use "scheme(s1mono)" to generate the graphs in this document as we want them in black and white.

[17] A similar chart is produced with `graph7`.

Figure 1: An example of a `graph matrix` chart



We can save any graph we produce in Stata using the graph export command. After drawing our graph, Stata will open it in a new window. It is possible to save it using the menus FILE, SAVE AS. We can also type `graph export filename.png, replace`. Stata can save graphs in various different formats, but .png is the most straightforward. [18]. We can then use the graphs in other documents and presentations.[19]

It would take too long to go through all of the available options, but some of the most important ones refer to the title and axis labels. For example:

```
line income temp time, title(Time Series Graph of Income and Tempreature Over Time) ///
xtitle(Time Period) ytitle(Euro(Income) and Degrees(Temp)) caption(Source: icecream.dta)
```

[18].wmf is best for word documents, .eps is best for LaTeX.
[19]There is a useful graph editor available in Stata version 10 onwards.

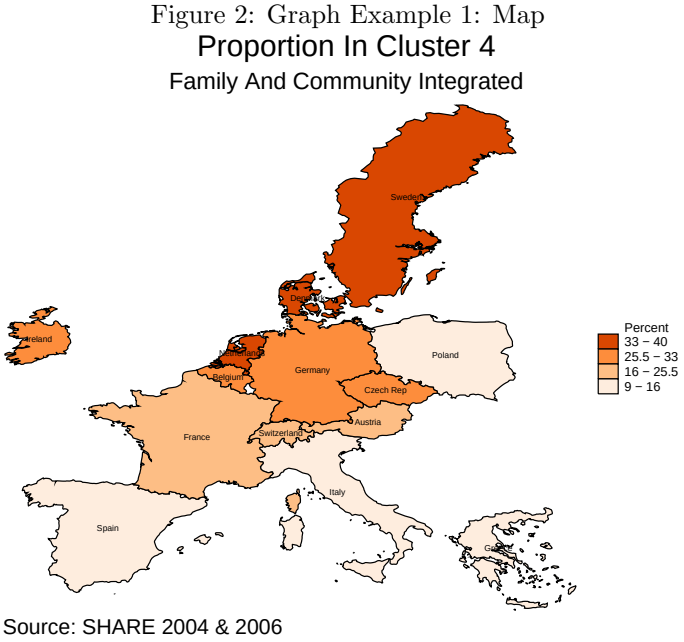
The "///" tells Stata to read the next line as part of the same command. Two examples of the kind of graphs that are possible are provided below. The first graph was made using the user written command "spmap". The second graph was generated using the cyear dataset with the following code:

```
twoway(scatter gdppc u5m if gdppc<1000 & year==2005, mlabel(country2) mlabsize(tiny)) ///
(lfit gdppc u5m if gdppc<1000 & year==2005), ///
caption(''Source: WHO and Penn World Tables'', size(tiny) span) legend(order(2 ''Linear Fit'')) ///
title(''National Income Per Head and Under 5 Mortality'', span) ///
ytitle(''GDP Per Capita in US Dollars'', size(small)) note(''Note: Correlation=-0.1737'') ///
plotregion(style(none)) legend(pos(4) col(1)) ///
xtitle(''Under 5 Mortality, Deaths Per 1,000 Births'') ///
graphr(lwidth(thin) ilwidth(vvthin) ilcolor(black) ilpattern(solid)) ///
subtitle(''Developing Countries in 2005'', span)
```

For more examples see:
http://www.survey-design.com.au/Usergraphs.html

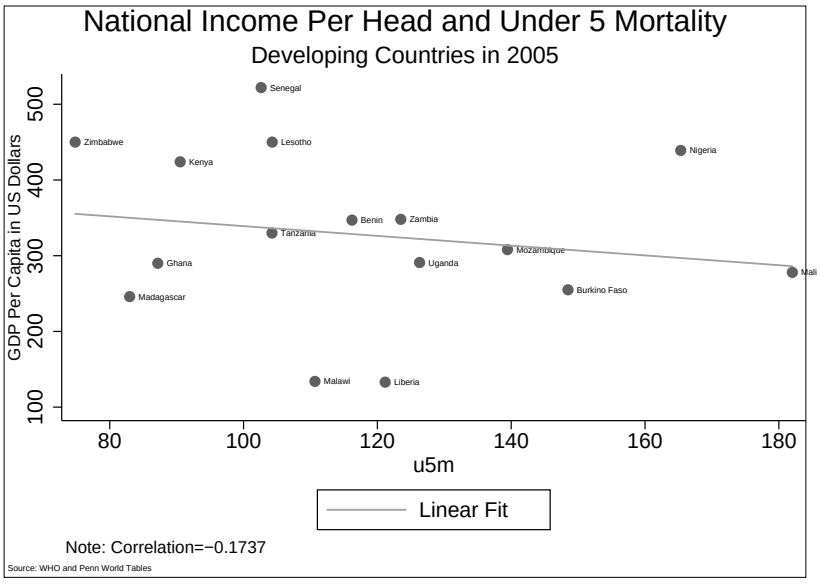
And:
http://www.ats.ucla.edu/stat/stata/library/GraphExamples/default.htm

Figure 2: Graph Example 1: Map



Source: SHARE 2004 & 2006

Figure 3: Graph Example 2: Labelled Scatterplot



3 Regression Analysis

The purpose of this lab is to provide an introduction to regression analysis using Stata, particularly the interpretation of the output produced by this and other statistical packages.

If we type `help regress` we see that the basic command takes the form regress y x1 x2 x3. Load the icecream data. `use icecream2, clear`. If we want to regress consumption on income temperature and price from our icecream dataset we type `reg cons price temp income`.[20] The output from this is presented in table 4. Stata will present the output almost immediately as this is a small dataset. We see that there are 30 observations used in the regression. Also note that Stata automatically includes a constant term, the variable _cons, not to be confused with your y variable, cons. You do not need to create a constant term yourself. [21] Interpreting output like this needs to become second nature if you are interested in pursuing research in economics. The most important of these numbers are the coefficients and their p values so we start with these. The coefficients (second column) tell us the effect of each X on Y (dy/dx). In this case none of the variables are in log form so the interpretation is straightforward. The coefficients tell us how Y changes if X changes by one unit. So in this case if price goes up by one unit, then consumption changes by -1.044, i.e. consumption falls by 1.044. Likewise, if income goes up by one unit, consumption will go up by .003. This is not the whole story however, some of these coefficients may be indistinguishable from zero (in a statistical sense). It is important to remember the difference between a coefficient and a t-stat. Coefficients are the "real" effects of variables but aren't very useful without knowledge of the variance. Each coefficient is assigned a t statistic in the 4th column. $t = \frac{\text{coefficient}}{\text{se(coefficient)}}$. The standard error of the coefficient is displayed in the third column.

As we know the distribution of the t statistic, so we are able to assess the probability that the population coefficient is equal to zero. This probability is the number displayed in the 5th column. So in this case we cannot reject the possibility that price has no effect on consumption, however we can reject the possibility that temperature and income have no effect. Another way of saying this is that the 95% confidence interval for price includes 0. Also included in the output are the Residual Sum of Squares, the Model (or Explained) Sum of Squares, and the Total Sum of Squares. We also have the R^2, which describes the proportion of variation in the outcome explained by the model. Remember that $R^2 = MSS/TSS$. We also have the F statistic, which is a test of whether the coefficients are all jointly equal to zero. An F test is analogous to a t test, except it examines several hypotheses at the same time. In this case F=22. As we know the F distribution we can assign it a p value in the same way we did with the t values. In this case we reject the possibility that the model coefficients are all zero. The F stat can also be seen as a test of whether the R^2 value has arisen at random. The R^2 term can only increase as more variables are added to the regression and can therefore be misleading, as the model may contain several redundant variables which have little explanatory power. The adjusted R^2 takes account of the number of variables used.

Interpretation of coefficients when some of the variables are in log form is a little different. If both X and Y variables are in logs, then the X coefficient tells you the percentage change in Y when X goes up by 1%. If Y is in log form and X is not, then the X coefficient tells you the percentage change in Y when X goes up by one unit. If Y is not in log form but X is, then the coefficient tells you the unit change in Y if X goes up by 1%. For example, suppose we generate a new variable which is the log of income `gen logincome=ln(income)`, and include this in our regression instead of our original variable. `reg cons price temp logincome`. Now the coefficient on logincome is .3 and we interpret this as meaning that a 1% rise in income results in consumption rising by .3 units.

3.1 Dummy Variables

Suppose we also want to include the province variable in our regression. As before we rename our province variable `rename var7 province`. It would be incorrect to run the model `reg cons price temp income province` as province is not a continuous variable (we can remind ourselves that it is a categorical variable

[20] Remember that Stata understands abbreviations.

[21] The constant term can be omitted with the option noconstant, but you need good reasons to justify this.

Table 4: OLS Regression Output

```
      Source |       SS       df       MS              Number of obs =      30
-------------+------------------------------           F(  3,    26) =   22.17
       Model |  .090250523     3  .030083508           Prob > F      =  0.0000
    Residual |  .035272835    26  .001356647           R-squared     =  0.7190
-------------+------------------------------           Adj R-squared =  0.6866
       Total |  .125523358    29  .004328392           Root MSE      =  .03683

------------------------------------------------------------------------------
        cons |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       price |  -1.044413    .834357    -1.25   0.222    -2.759458    .6706322
        temp |   .0034584   .0004455     7.76   0.000     .0025426    .0043743
      income |   .0033078   .0011714     2.82   0.009     .0008999    .0057156
       _cons |   .1973149   .2702161     0.73   0.472    -.3581223     .752752
------------------------------------------------------------------------------
```

which runs from 1-4 with `tab province`). Instead we need to include a separate dummy variable for each category.[22] We could create 3 new variables, taking the value one for each category and zero otherwise.[23] `gen leinster=(province==2)` and `gen munster=(province==3)` and `gen Connacht=(province==4)`.[24] Now we can include these 3 variables in our regression. `reg cons price temp income leinster munster connacht`. There is an easier way of doing this however. We do not actually need to generate the separate dummy variables ourselves if we use the xi command, which expands categorical variables for us. `xi: reg cons price temp income i.province`. Remember to include "i." before the categorical variable you wish to expand. Note that these dummies will appear in your variable window.

It is important to be able to interpret the coefficients on these dummy variables correctly. Notice again that there are only three dummies, despite the fact that there are four categories in the province variable. Each of these refers to the effect of each category relative to the omitted category. We can clearly see that the omitted category is province 1 (Ulster). In any case, Stata tells us which one is omitted. If we want to change the omitted category, for example if we wanted to compare the effect of the other provinces to being in Leinster, we can set which group Stata omits. `char province[omit] 2`. If we run the regression again we see this is indeed the case `xi: reg cons price temp income i.province`. The output is presented in table 5.

3.2 Outreg2

You will encounter dozens if not hundreds of regressions, and if you need to present these results copying and pasting from Stata is not really an option. Fortunately there is a user written command, similar to tabout, which automates the export of regression results. First we need to install the programme. `findit outreg2` or `search outreg2, all`, or `ssc install outreg2`. Having done this, we can check the help file `help outreg2`. We run the outreg command after our regression to save our results in excel format (this is probably best, unless you are using LaTeX). We need to specify the filename. `outreg2 using test, excel replace`. This exports all coefficients, if required you can specify the variables for export, e.g. `outreg2 temp income using test, excel append`. The replace and append options operate as with tabout. If you choose append, new regression results will appear in different columns in the file. The excel option is also

[22] Including binary variables on the right hand side of a regression does not violate any of the assumptions which make OLS BLUE.

[23] We don't need 4 variables, this would be falling into the Dummy Variable Trap.

[24] A quicker way to do this is `tab province, gen(province)`.

Table 5: OLS Regression Output With Dummy Variables

```
i.province        _Iprovince_1-4      (naturally coded; _Iprovince_1 omitted)
      Source |       SS       df       MS              Number of obs =      30
-------------+------------------------------           F(  6,    23) =   10.27
       Model |  .091413007     6  .015235501           Prob > F      =  0.0000
    Residual |  .034110351    23  .001483059           R-squared     =  0.7283
-------------+------------------------------           Adj R-squared =  0.6574
       Total |  .125523358    29  .004328392           Root MSE      =  .03851

------------------------------------------------------------------------------
        cons |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       price |  -1.272991   .9760936    -1.30   0.205    -3.292195    .7462123
        temp |   .0034345   .0004815     7.13   0.000     .0024385    .0044306
      income |   .0030756   .0012691     2.42   0.024     .0004503     .005701
_Iprovince_2 |   .0093664   .0190921     0.49   0.628    -.0301285    .0488614
_Iprovince_3 |    .003358   .0225584     0.15   0.883    -.0433076    .0500236
_Iprovince_4 |   .0181066   .0210197     0.86   0.398     -.025376    .0615892
       _cons |   .2737497   .3113172     0.88   0.388     -.370259    .9177583
------------------------------------------------------------------------------
```

important as this ensures that excel will recognise the way the standard errors are formatted. An example is shown in table 7. It is also possible to export the results directly to a word document with the "word" option `outreg2 using test2, word replace`. In this case you will get a document with the extension ".rtf".

3.3 Hypothesis Testing

We have already established the meaning of the t statistics and p values, these refer to a test of the hypothesis that the coefficient on that variable in the population is equal to zero. However, we can also test any linear hypothesis using the test command. For example, theory may tell us that the coefficient on the log of income should be equal to one. `test _b[income]=1`. We use _b and square brackets to refer to the regression coefficients on our variable(s) of interest. In this case we find a very large F statistic, and as the p value is $<.05$ or even .01 we reject the hypothesis that the coefficient on income is equal to one. We can also test several hypotheses jointly `test (_b[income]=1) (_b[temp]=.01)`. If we wish to test a non-linear hypothesis, such as price*temperature=income, we use the testnl command `testnl _b[price]*_b[temp]=_b[income]`.

Suppose you run a regression of wages that includes experience and experience squared as independent variables and find that neither have significant t-tests. You should do a joint-test of the significance of the two terms measuring experience. The distribution of multiple variables differs from that of a single variable, so you cannot use a t-test; you need to run an F-test and use the appropriate critical values. To test several parameters, use the `test` or `testparm` commands. For example, `testparm exper expersq` outputs the F- and p-values of the test that these coefficients are jointly-insignificant.

3.4 Post Regression Commands

There are a number of useful postregression commands involving the "predict" command. To obtain the predicted values after our regression `xi: reg cons price temp income i.province`, we use `predict xbhat`. We can them compare these with the actual outcome with a scatter diagram. `scatter cons xbhat`. If the model was perfect the correlation between the two variables would be 1. We can also obtain the residuals with the command `predict residuals, res`, and examine the residual corresponding to each value of our

Table 6: Outreg Example

VARIABLES	(1) cons
price	-1.273
	(0.976)
temp	0.00343***
	(0.000482)
income	0.00308**
	(0.00127)
_Iprovince_1	-0.00937
	(0.0191)
_Iprovince_3	-0.00601
	(0.0233)
_Iprovince_4	0.00874
	(0.0202)
Constant	0.283
	(0.318)
Observations	30
R^2	0.728

Standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1

x variables `scatter res temp` and `scatter res income`. Both of these plots show the residuals centred around zero which is not indicative of any misspecification, however we can test this more formally.

3.5 Interaction Effects

Interaction effects are an important tool in regression analysis. Suppose we have reason to believe that the effect of income differs depending on the province. We can examine this by interacting income with our province. `xi: reg cons price temp i.province*income`. Now we have three extra variables, income*province2, income*province3 and income*province4. These variables tell us the differential effect of income in each province relative to income in our base province. Note that the provincial dummy variables and income variables (which gives us the average effect of income) are still included. As before we compare the effects of the income interaction to the base category which is income*province1. In this case there is no statistically different effect of income in each province. We can test whether all of these coefficients are equal `test _b[_IproXincom_2]=_b[_IproXincom_3]=_b[_IproXincom_4]`. We fail to reject the hypothesis that all these coefficients are equal.

3.6 Specification and Misspecification Testing

It is important to test whether the errors have constant variance, i.e.the assumption of homoskedasticity, otherwise statistical inference becomes unreliable, even if the coefficients are unbiased. The Breusch-Pagan tests whether the residuals depend on the level of the X-variables (e.g. if the variance of income increases as the income increases.) This is activated in Stata with the `hettest` command. (If installed, you can conduct a White test with the command `whitetst`.)

If you have a problem with heteroskedasticity, you can use White (1980) heteroskedastic-consistent standard errors by adding the "robust" option, e.g. `regress y x1 x2 x3, robust`. Robust errors are not always appropriate. You should also consider whether you should cluster your standard errors with the "cluster(variable)" option. See chapter 8 in Mostly Harmless Econometrics for more details on both these issues. With regards to autocorrelation you can produce a correlogram with the `ac` command. The Durbin-Watson test is called with `dwstat`. If you'd like to run a Cochrane-Orcutt regression instead of OLS, you can use `prais` regression method. The standard test for model specification is the Ramsey RESET test. The relevant command is `ovtest`. A high F-test suggests your model is improperly specified. To test for multicollinearity, enter the `vif` command after a regression. This tests the variable inflation factor, which is basically the effect of collinearity. One suggestion is that values greater than ten warrant further investigation. You can use Stata's `sktest` to test normality. You could plot the residuals with `predict myresids, resid` which produces the residuals in a variable named `myresids`. Then you can produce a `histogram` of this variable, or use a `qnorm` plot. For example try `kdensity myresids, normal` to compare your graph to the normal bell-curve.

Table 7: Linear Probability Model Output

```
Linear regression                               Number of obs =    3296
                                                F(  3,  3292) =  135.69
                                                Prob > F      =  0.0000
                                                R-squared     =  0.1010
                                                Root MSE      =  .42809
------------------------------------------------------------------------------
             |               Robust
    highwage |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       exper |   .0101499    .003119     3.25   0.001     .0040345    .0162652
        male |   .1605543   .0150249    10.69   0.000     .1310953    .1900133
      school |   .0781009   .0044197    17.67   0.000     .0694352    .0867666
       _cons |  -.7894874   .0600989   -13.14   0.000    -.9073225   -.6716523
------------------------------------------------------------------------------
```

4 Binary Regression

Including categorical or binary measures on the right hand side of a regression does not pose any problems for estimation, however it is a different matter when the Y variable is categorical. We first examine the case of a binary dependent variable. We will use the wages data. `use wages1`. We have information on an individual's years of schooling, their current wage(in logs), their experience (in years), and their gender. We first summarise the data `describe` and `summarize`. So we have 4 variables and 3296 observations. We examine the wage variable with `histogram wage`. As there are clearly a number of outliers we restrict this graph to values less than 20, and add a kernel density plot. `histogram wage if wage <20, kdensity`. The variable appears roughly lognormal. Suppose we are interested in the determinants of wages. As an initial step, we could examine the correlation between our variables. `pwcorr wage exper male school, sig`. Wages are significantly positively correlated with being male and years of schooling. We could also examine the average wage by years of schooling `tabstat wage, by(school)`. Again we see evidence of positive correlation, however the relationship appears to be non-linear. The lowest average wage is for those in the 8 and 9 years of schooling. This is easier to see graphically `graph bar wage, over(school) title(Wages by Years of Schooling)`. We could run a standard regression with this data, but instead suppose we are interested in the determinants of earning a wage with a value greater than 7. We generate a variable that takes the value 1 if the individual earns this value or greater, and zero otherwise. `gen highwage=(wage>7)`. 28% of the sample are in this category `tab highwage`. Males are more likely to be in this category `tab highwage male, row`.

4.1 The Problem With OLS

We could run our analysis using OLS `reg highwage exper male school, r`, and outreg the results `outreg2 using test, replace excel ctitle(OLS)`.[25] We need robust standard errors as the binary nature of our outcome induces heteroskedasticty. We now interpret our coefficients in terms of the probability of earning a high wage (wage greater than 7).[26]

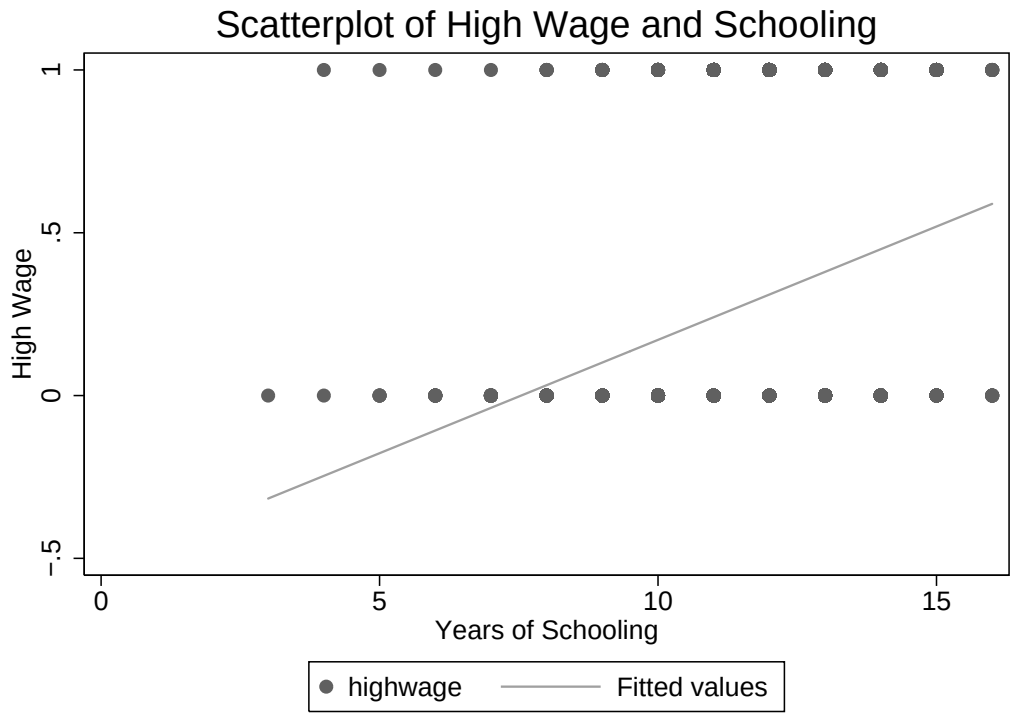
So for example, being male increases the probability of being in this category by 16%. Likewise an extra year of schooling raises the probability of earning a high wage by 7%. Although OLS is still unbiased in these situations, there is a difficulty as some groups may be predicted to have a negative probability of the outcome. To see this graphically, we examine a plot of wages and schooling, and fit an OLS regression line.

[25] The ctitle option places a title on the regerssion column.

[26] Using OLS on a binary outcome is often referred to as the linear probability model (LPM).

As our outcome is binary, all observations will lie either on 1 or 0. `twoway(scatter highwage school)(lfit highwage school), scheme(s1mono) title(Scatterplot of High Wage and Schooling) ytitle(High Wage) xtitle(Years of Schooling)`. Because there are a limited number of outcomes, each point could represent several hundred observations or 10, it is not possible to tell. To get a better idea of the distribution of our variables, we add some random noise to our plot using the jitter option `twoway(scatter highwage school, jitter(10))(lfit highwage school), title(Scatterplot of High Wage and Schooling) ytitle(High Wage) xtitle(Years of Schooling)`. This issue is clear from this graph, for individuals with less than 9 years of schooling, the predicted probability of earning a high wage is negative, which clearly does not make sense. To confirm this we obtain the predicted probabilities from our model. `predict xbhat` and check them against years of schooling `tabstat xbhat, by(school)`.

Figure 4: A Problem With OLS



4.2 Logit and Probit

Two of the most popular alternatives are the probit and logit estimators.[27] Both are maximum likelihood estimators which involve slightly different distributional assumptions, but should produce roughly the same results.[28] We run regressions with each of these `probit highwage exper male school` and `logit highwage exper male school`. We also verify that the predicted probabilities are with the 0-1 range. `predict xbhat2` and `tabstat xbhat2, by(school)`. We can also graph these `graph bar xbhat2, over(school)`. Figure 5 compares the predicted probabilities from each model. In this case logit and probit both give very similar results.

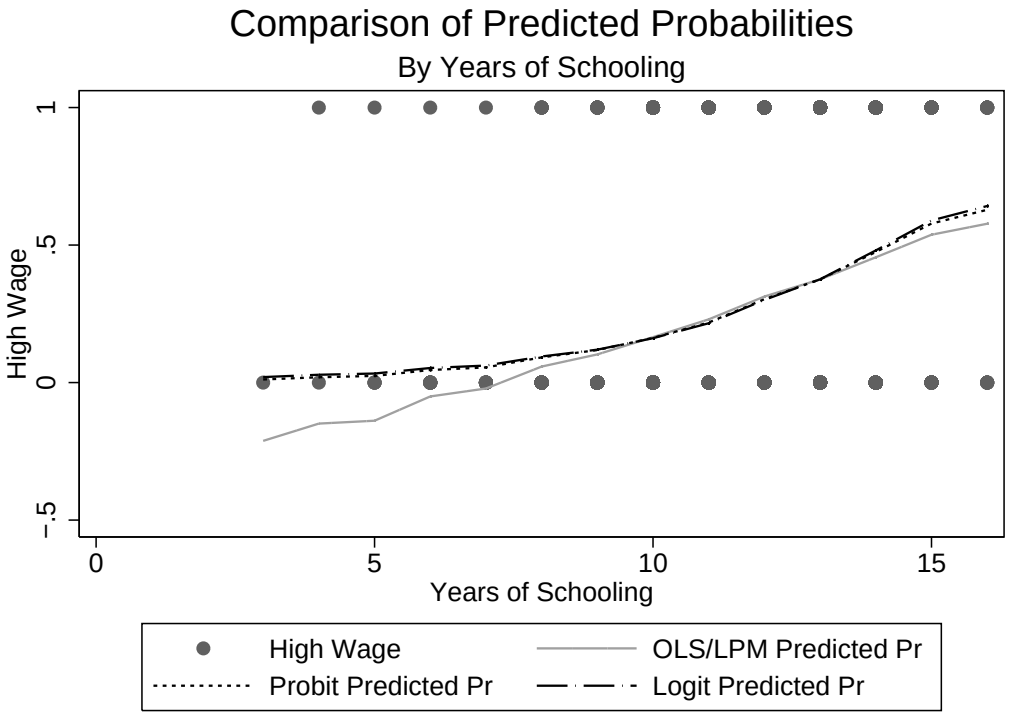
[27] A problem with the logit estimator is the independence of irrelevant alternatives assumption. The nested logit, "nlogit" command, is a designed to overcome this difficulty.

[28] See any standard econometrics text for further details, and how these relate to the concept of an underlying latent variable.

Table 8: Logit and Probit Output

```
Iteration 0:   log likelihood = -1968.4829
Iteration 1:   log likelihood = -1794.7776
Iteration 2:   log likelihood = -1792.4022
Iteration 3:   log likelihood = -1792.4006

Probit regression                                 Number of obs   =       3296
                                                  LR chi2(3)      =     352.16
                                                  Prob > chi2     =     0.0000
Log likelihood = -1792.4006                       Pseudo R2       =     0.0895

------------------------------------------------------------------------------
    highwage |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       exper |   .0320678   .0113016     2.84   0.005      .009917    .0542186
        male |   .5130279   .0496018    10.34   0.000     .4158102    .6102456
      school |   .2565379   .0161262    15.91   0.000     .2249311    .2881447
       _cons |  -4.138153   .2306573   -17.94   0.000    -4.590233   -3.686073
------------------------------------------------------------------------------

Iteration 0:   log likelihood = -1968.4829
Iteration 1:   log likelihood = -1796.4414
Iteration 2:   log likelihood = -1791.3207
Iteration 3:   log likelihood = -1791.3022

Logistic regression                               Number of obs   =       3296
                                                  LR chi2(3)      =     354.36
                                                  Prob > chi2     =     0.0000
Log likelihood = -1791.3022                       Pseudo R2       =     0.0900

------------------------------------------------------------------------------
    highwage |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       exper |   .0530249   .0194082     2.73   0.006     .0149855    .0910644
        male |   .8678092   .0847772    10.24   0.000     .7016489     1.03397
      school |   .4386614   .0283132    15.49   0.000     .3831685    .4941544
       _cons |  -7.028232   .4058646   -17.32   0.000    -7.823712   -6.232752
------------------------------------------------------------------------------
```

Figure 5: Problem Solved With Probit and Logit



4.3 Marginal Effects

It is important to recognise that these coefficients are not the same as the output generated by an OLS regression as they refer to the unobserved latent variable. They are not marginal effects, i.e. they do not tell us the average effect of a change in the X variable on Y (dy/dx). This is the case with OLS as it is a linear estimator. All we can interpret from probit or logit coefficients is the direction of the average effect, and the significance, but not the magnitude. Marginal effects are rarely reported in other disciplines in these models [29], however economists are usually interested in the magnitude of an effect, not just its statistical significance. We can calculate the marginal effects using the mfx compute command. Whereas the coefficients from logit and probit will differ due to scaling, the marginal effects should be almost identical. Outreg2 also works for exporting marginal effects, we will use this to compare different ways of calculating marginal effects. `logit highwage exper male school` then `mfx compute` and `outreg2 using test, mfx excel append ctitle(mfx logit)`. `probit highwage exper male school` then `mfx compute` and `outreg2 using test, mfx excel replace ctitle(mfx probit)`.

We see that this is indeed the case. Not only that, but these are also almost identical to the OLS results. For example, for the probit model, an increase in a year of schooling increases the probability of earning a high wage by 8%, and likewise for the logit model. As the marginal effects differ depending on the value of the x variables, there are a number of different ways of calculating these.

[29] Often odds ratios from logit models are reported in psychology, sociology and epidemiology.

Table 9: Marginal Effects Output

```
Marginal effects after probit
      y  = Pr(highwage) (predict)
         =  .26504098
------------------------------------------------------------------------------
variable |      dy/dx    Std. Err.     z    P>|z|  [    95% C.I.   ]      X
---------+--------------------------------------------------------------------
   exper |   .0105044       .0037    2.84   0.005   .003251  .017758   8.04187
    male*|   .1656721      .01561   10.61   0.000    .13508  .196264   .523968
  school |   .0840339      .00521   16.14   0.000   .073827   .09424   11.6302
------------------------------------------------------------------------------
(*) dy/dx is for discrete change of dummy variable from 0 to 1

Marginal effects after logit
      y  = Pr(highwage) (predict)
         =  .26010828
------------------------------------------------------------------------------
variable |      dy/dx    Std. Err.     z    P>|z|  [    95% C.I.   ]      X
---------+--------------------------------------------------------------------
   exper |   .0102047      .00373    2.74   0.006   .002892  .017517   8.04187
    male*|   .1645799      .01554   10.59   0.000   .134131  .195029   .523968
  school |   .0844213      .00525   16.09   0.000   .074139  .094704   11.6302
------------------------------------------------------------------------------
(*) dy/dx is for discrete change of dummy variable from 0 to 1
```

By default, Stata calculates these marginal effects at the mean of the independent variables, however it is also possible to evaluate them at other values. For example, suppose you suspect that the effect of experience and schooling on wages differs for men and women. Then you could evaluate the marginal effects for women. `mfx compute, at(male=0)` and `outreg2 using test, mfx excel append ctitle(mfx probit women)`. For men `mfx compute, at(male=1)` and `outreg2 using test, mfx excel append ctitle(mfx probit men)`. To reiterate, because OLS is a linear estimator the estimated marginal effect is the same at every set of X values. You can think about this in terms of the slope of the regression line in the bivariate case. OLS is a straight line, so the slope of the line is the same at every point, unlike with logit and probit.

The marginal effects of experience and schooling appear larger for men than women. There other ways of approaching how to estimate marginal effects. An alternative is to calculate the marginal effects for every value and then take the average to obtain the average partial effect. For this we use the user written command margeff. `ssc install margeff` to install. Then we run the command with the replace option as we wish to export the results. `margeff, replace`. We use outreg2 without the mfx option this time. `outreg2 using test, excel append ctitle(amfx probit)`. We can see that in this case both approaches give similar results. Mfx2 is another user written command which produces marginal effects. If you are using a probit model, you can obtain the marginal effects directly (without the coefficients) with the command `dprobit highwage exper male school`.

Table 10: Alternative Binary Estimators for HighWage

VARIABLES	(1) mfx probit	(2) mfx probit women	(3) mfx probit men	(4) amfx probit	(5) mfx hetprob
exper	0.0105***	0.00856***	0.0119***	0.00987***	0.0101***
	(0.00370)	(0.00304)	(0.00418)	(0.00347)	(0.00368)
male	0.166***	0.166***	0.166***	0.158***	0.161***
	(0.0156)	(0.0156)	(0.0156)	(0.0171)	(0.0155)
school	0.0840***	0.0685***	0.0951***	0.0785***	0.0812***
	(0.00521)	(0.00441)	(0.00606)	(0.00437)	(0.00546)
Observations	3296	3296	3296	3296	3296

Standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1

If you are concerned about potential misspecification, the heteroskedastic probit model allows you to specify the variance as a function of one or more of the independent variables. Suppose we suspect that the variance of our error term depends on your level of schooling, we could estimate the following `hetprob highwage exper male school, het(school)` and compute the marginal effects `mfx compute` and export them `outreg2 using test, excel mfx append ctitle(mfx hetprob)`. The LR test of lnsigma is a test of our assumption that the variance depends on schooling. We reject at the 5% significance level but fail to reject at the 10% level. Finally, note that if you are including interaction effects in these kinds of binary models you must be careful about interpreting them. The user written command "inteff" is recommended.

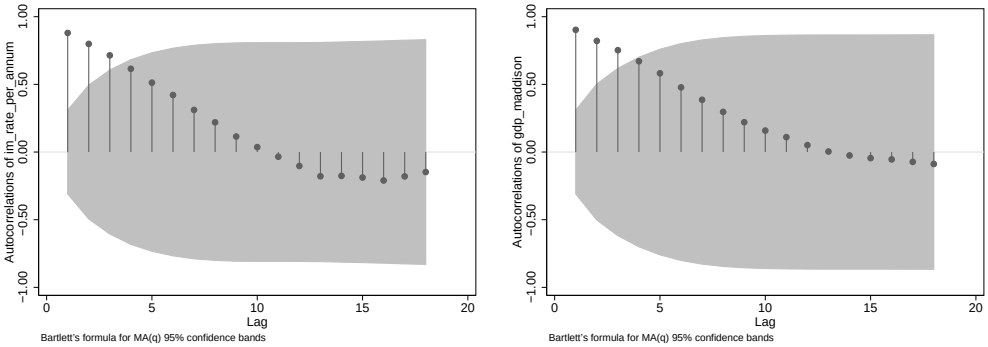


Figure 6: Autocorrelation Functions For Infant Mortality and GDP

5 Time Series

We will examine the basic fundamentals of time series analysis using data on Irish infant mortality. First we load the data `use im, clear` and apply some labels. `lab var netm ''Net Migration''` and `lab var at ''Average Annual Tempreature'' lab var ar ''Average Annual Rainfall''`. We also generate a time squared variable `gen year2=year^2`. We will first summarise the data `describe` and `sum`, and plot our variables. As we have time series data we need to tell Stata. `tsset year`. First we sort the data. `sort year`.

5.1 Initial Analysis

We could naively run a regression of infant mortality on gdp and other factors, but this is potentially a very serious mistake. `reg im gdp at ar netm` and export the results `outreg2 using timeseries, excel replace ctitle(Naïve Regression)`. These results suggest that GDP has an impact on infant mortality, however time series analysis involves dealing with the specifics of how variables behave over time. Ignoring the fact that variables are trending over time could return spurious results.[30] The high R^2 is a warning sign that this is a problem here. We could check the Durbin Watson test statistic `estat dwatson` and `estat bgodfrey`. Both of these indicate the potential for the residuals to be non-stationary, as we reject the assumption of no serial correlation. We should first examine our variables. `line im year` and `line gdp year, title(''Time Series Plot of Irish GDP'')`, remembering that Stata understands unambiguous abbreviations. GDP is clearly non-stationary, as it infant mortality, at least for the period after 1947. We can also check the autocorrelation and partial autocorrelation plots. `ac im pac im ac gdp pac gdp`. In each case we have linearly declining autocorrelation functions and a high value of the first partial autocorrelation function, indicating non-stationarity.

5.2 Testing For Unit Roots

We now confirm this with formal dickey fuller tests. Two issues arise when testing for stationarity, choice of lag length and whether constants and trends should be included in the test. For the second issue, it is recommended that the Dolado procedure[31] is followed, however here we will assume that it is correct to include both. We deal with the first issue by choosing lag lengths that reflect the best fit to the variables. `varsoc im`. In each case of infant mortality we choose two lags with one for GDP. `dfuller im, trend lags(2). dfuller gdp, trend lags(1)`.

[30] See any standard econometrics text for details.

[31] See Dolado, J., Jenkinson, T., & Sosvilla-Rivero, S. (1990). Cointegration and Unit Roots. Journal of Economic Surveys, 4(3), 249-27

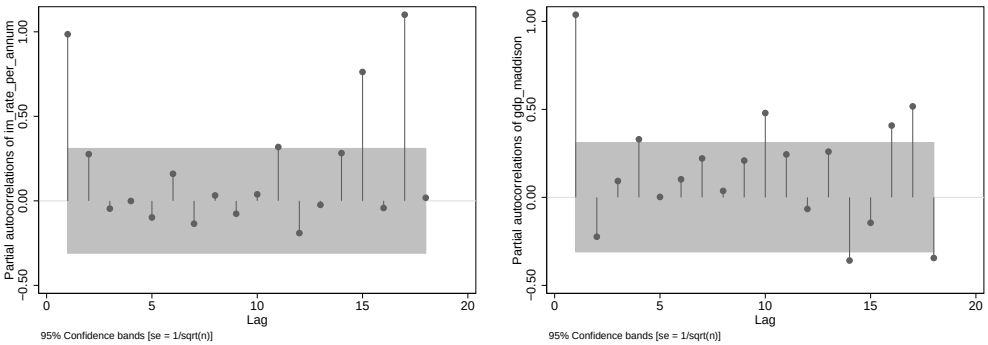


Figure 7: Partial Autocorrelation Functions For Infant Mortality and GDP

Table 11: Dickey Fuller Test Ouput

```
Augmented Dickey-Fuller test for unit root         Number of obs   =        38
                                 ---------- Interpolated Dickey-Fuller ---------
                    Test         1% Critical       5% Critical      10% Critical
                  Statistic           Value             Value             Value
------------------------------------------------------------------------------
 Z(t)              -1.624           -4.260            -3.548            -3.209
------------------------------------------------------------------------------
MacKinnon approximate p-value for Z(t) = 0.7830
```

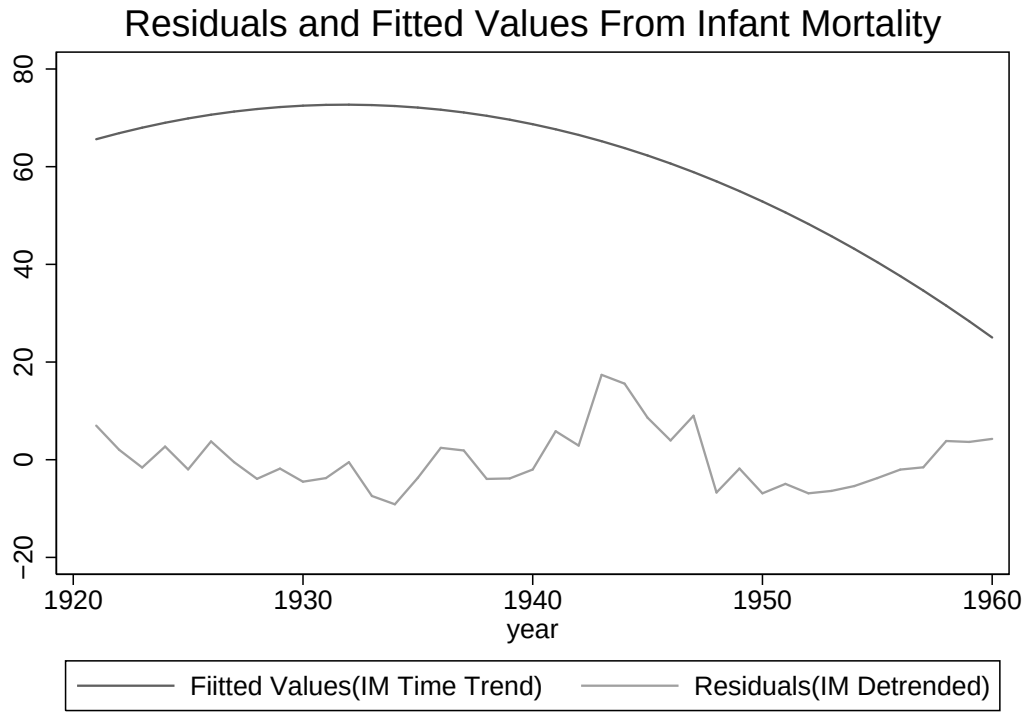
It is important to be able to interpret these tests properly. The null hypothesis is the presence of a unit root. If the test statistic does not exceed the 5% critical value in absolute terms (i.e. is not more negative then the 5% critical value), we cannot reject non-stationarity. In each case here we fail to reject the null hypothesis of a unit root, and conclude that both variables are integrated of order 1. There are several potential ways of addressing this. Remember that under normal conditions non-stationarity violates the regularity assumption on which OLS is based. To deal with this problem, we could detrend the variables, we could check for the presence of a cointegrating relationship between gdp and infant mortality, and finally we could run a model in first differences.

5.3 Dealing With Non-Stationarity

To detrend the variables we regress each variable on time and time squared to obtain the residuals. We include time squared as we suspect the trend to be quadratic. The residuals from this regression will be that particular variable with the time trend removed. `reg im year year2` and `predict imres, r`. Note that the predicted values will correspondingly give us the infant mortality time trend. `predict imtime`. We can graph this with `line imtime imres year, title(''Residuals and Fitted Values From Infant Mortality'') legend(lab(1 ''Fiitted Values(IM Time Trend)'') lab(2 ''Residuals(IM Detrended)'')) scheme(s1mono)`.

The detrended series looks like it could be stationary here, but this is could be due to the scaling on the graph. For gdp `reg gdp year year2` and `predict gdpres, r`. We could run the regression on our detrended variables `reg imres gdpres at ar netm` and export the results `outreg2 using timeseries, append excel ctitle(Detrended Regression)`. However we should make sure that this is correct. `varsoc imres` and `varsoc gdpres`. We can now use formal dickey fuller tests to check whether these detrended variables are stationary. `dfuller imres, trend lags(1)` and `dfuller gdpres, trend lags(2)`. In each

Figure 8: Using OLS To Detrend Variables



case, detrending the variables is not enough to induce stationarity. So we cannot simply include these detrended variables in our regression. Note that in fact this kind of detrending procedure is hardly ever necessary, as it is equivalent to simply including the time trend on the right hand side of our regression.[32]. Compare the coefficients on gdp from these two regression `reg im_ gdp_ year year2`[33] and `reg imres gdpres`. There is a second possibility for running a valid regression, it may be that despite the fact that infant mortality and gdp are non-stationary, they may be cointegrated, i.e. share a common trend. If we were running Stata version 10 or above we could run a Johansen test for cointegrating vectors between the two variables, however there are alternatives. The principle in testing for cointegration is that if such a relationship exists between two variables, the residuals from their regression should be stationary. So we run our basic model, this time including year and year squared. We know from the above analysis that we cannot rely on this detrending to induce stationarity, however we will examine the residuals to test for a cointegrating relationship.[34] `reg im_ gdp_ at ar netm year year2`. We first graph the residuals `line cointres year` and `ac coint` and `pac coint`. `varsoc cointres` and then `dfuller cointres, trend lags(1)`. We therefore cannot reject the null that the residuals are non-stationary, and therefore reject the presence of a cointegrating relationship. We finally return to the option of running a regression in first differences. Fortunately Stata provides a number of time series operators which simplify the process. L.var is the lag of that variable, i.e. l.im is im_{t-1}. D.im is the first difference ($im_t - im_{t-1}$), and f.im is the first lead (im_{t+1}). We can easily generate a new variable that is the first difference of gdp. `gen dgdp=d.gdp_` and for infant mortality `gen dim=d.im_`. Both now look roughly stationary, although gdp

[32]This is the Frisch-Waugh-Lovell theorem.
[33]Note that we need to add underscores in order not to confuse Stata as we created two new variables imres and gdpres.
[34]This form of unit root test is known as an Engle Granger cointegration test.

Table 12: A Comparison of Time Series Models

VARIABLES	(1) Naive Regression	(2) Detrended Regression	(3) First Differences
gdp_maddison	-0.0117***		
	(0.00115)		
at	1.330	-0.846	-3.418
	(3.139)	(2.881)	(2.704)
ar	-0.00902	-0.00642	-0.0134
	(0.00975)	(0.00839)	(0.00851)
netm	0.0531	-0.0256	0.00644
	(0.0889)	(0.0586)	(0.0612)
gdpres		-0.0130***	
		(0.00331)	
dgdp			-0.00154
			(0.00446)
Constant	167.4***	14.37	47.12
	(35.22)	(32.72)	(30.56)
Observations	35	35	35
R^2	0.860	0.379	0.119

Standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1

still causes some concerns. `line dim year line dgdp year`. We reject non-stationarity in both cases in formal tests. `varsoc dim varsoc dgdp`. `dfuller dim, lags(1) trend`. `dfuller dgdp, trend lags(0)`. We now proceed with a regression in first differences `reg dim dgdp at ar netm` and export the results `outreg2 using timeseries, excel append ctitle(First Differences)`.[35] We conclude that the only valid model is the first differences regression, and that gdp has no effect on infant mortality.

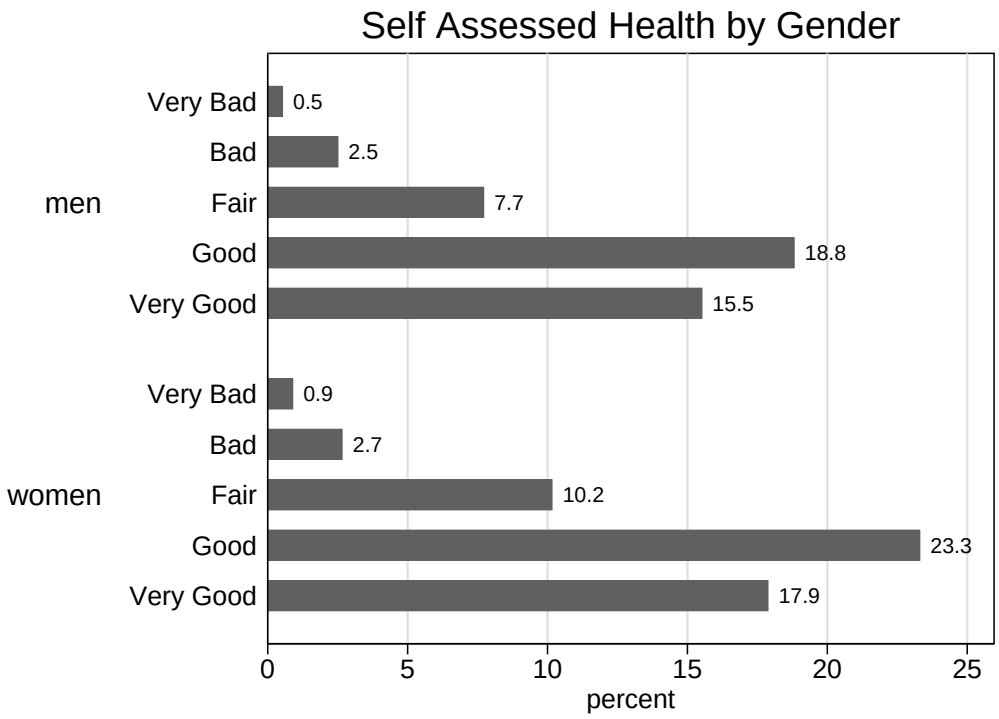
[35] We could include year as a variable here, however it makes little difference.

6 Ordinal and Multinomial Regression

6.1 Ordinal Data

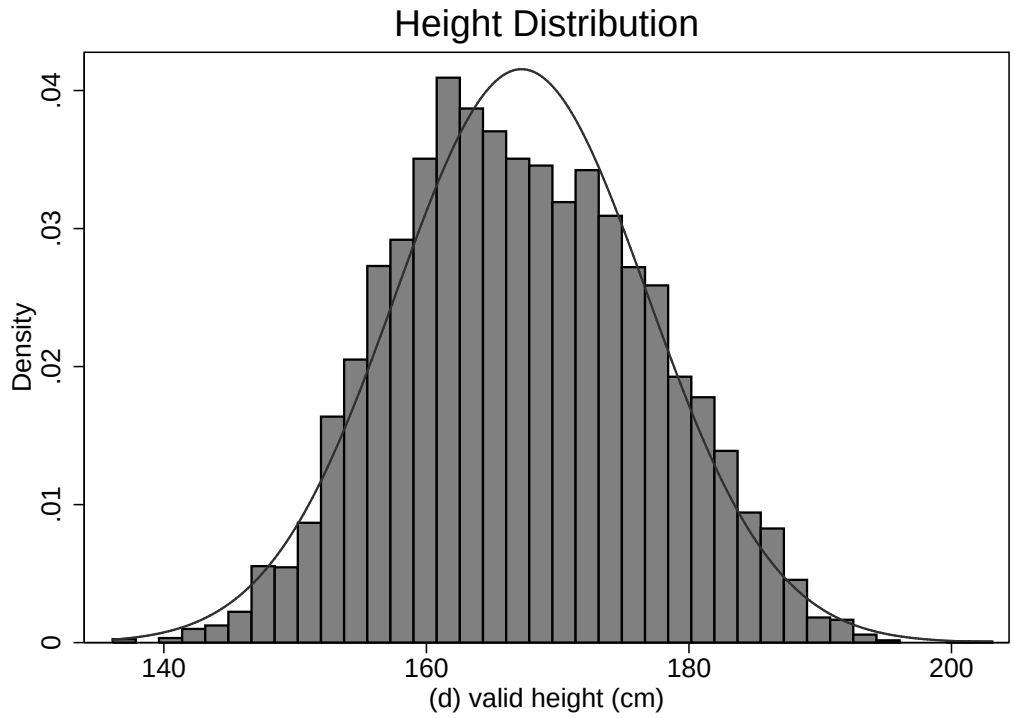
We have seen an example of how to analyse data where the outcome variable is binary. We will now examine the case where the Y variable is categorical. There are two situations; where the variable has a clear ranking (e.g. health or education) we will consider the ordered logit and ordered probit models. When the Y variable has no clear ranking the multinomial logit or multinomial probit is appropriate. We will load data taken from the Health Survey for England. `use ordinal, clear`. As always we examine our data. `describe` and `sum`. We will also examine our outcome variable in more detail. `rename genhelf health` renames the variable. `tab genhelf`. We see that this is the standard health question asked in surveys. Individuals are asked to assess their general health on a scale of 1 to 5, with 1 being very bad and 5 excellent. If we examine the labelling on this variable, we see that the coding is reversed. `codebook health` and `labelbook genhelf`. We will fix this, and also tidy the value labels. `recode health (1=5)(2=4)(4=2)(5=1)`. We can now change the labels `lab def genhelf 1''Very Bad'' 2''Bad'' 3''Fair'' 4''Good'' 5''Very Good'', modify`. Now this looks better `tab health`. Suppose we are interested in comparing health across gender. We could take the average health score, `tabstat health, by(gender)`, but this is not ideal and in fact there is little difference here. Instead we should compare across categories. We could graph this relationship `catplot health sex, percent scheme(s1mono) title(Self Assessed Health by Gender) blabel(bar, format(%8.1f))`.

Figure 9: Health Distribution



There are more women in the very good and good categories, but also more in the bottom two categories. Note how we added labels onto the end of the bars with the "blabel" option. We also examine height. `rename htval height` and `codebook height` and `labelbook htval`. We replace missing values `mvdecode height, mv(-1)`.

Figure 10: Height Distribution



We see that height is roughly normally distributed `hist height, normal scheme(s1mono) title(Height Distribution)`. Suppose we wish to examine the determinants of self reported health status. We generate an age squared term `gen age2=age`^{2}. We could start with a simple OLS model `xi:reg health age age2 i.sex i.topqual3 height`. We save the results `outreg2 using ordinal, excel replace`.

However, the difficulty with OLS is potentially more important in this situation than in the previous case where we had a binary variable. The assumption behind these kinds of models is that the 5 point scale is really a manifestation of an underlying continuous health index. OLS assumes that movement from very bad to bad is equivalent to movement from good to very good. A simple examination of the distribution shows that it is highly skewed towards good health. The problem is that we do not know where these thresholds lie in this unobserved continuous index. The ordered logit and probit models take account of these thresholds. They are similar to the standard probit and logit models we encountered in the last section. `xi:oprobit health age age2 i.sex i.topqual3 height` and `xi:ologit health age age2 i.sex i.topqual3 height`. The output from the ordered probit model is shown in table 13. The "cut" values refer to estimates of the thresholds.

The same issue as in the binary case arises here. These are coefficients and not marginal effects, so we can only identify the significance of the variable and not the magnitude. We can obtain marginal effects, but in this case it will be the marginal effect of the variable for being in that particular category compared to all others. In the ordered logit case, if we were interested in the marginal effect for being in very good health `mfx compute, predict(outcome(5))`. Note that these are computationally intensive and can take a long time to estimate. In this case we see that a unit increase in height increases the probability of being in the very good category by .3%. Being in education category 8 reduces the probability of being in this category by nearly 20%. `mfx2` will estimate marginal effects for all categories. This raises a difficult question about interpretation. Here there are 5 categories, and with 11 variables, which means 55 coefficients.

Table 13: Ordered Probit Output

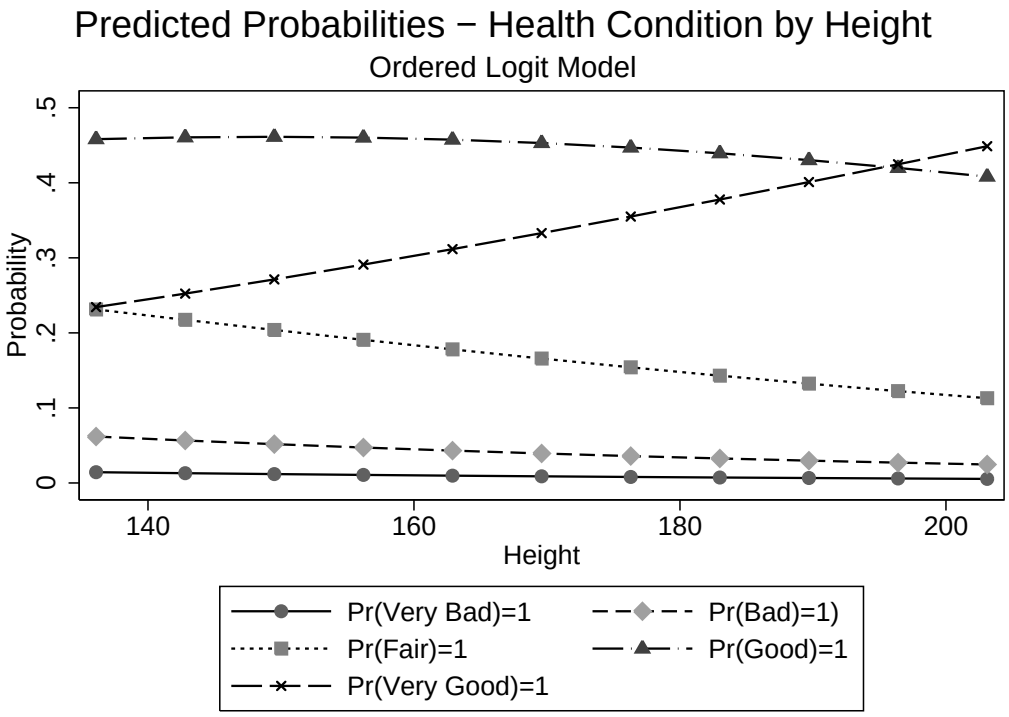
```
i.sex             _Isex_1-2           (naturally coded; _Isex_1 omitted)
i.topqual3        _Itopqual3_1-8      (_Itopqual3_1 for topqual3==-1 omitted)
Iteration 0:   log likelihood = -8403.7684
Iteration 1:   log likelihood = -8109.4856
Iteration 2:   log likelihood = -8109.3117
Iteration 3:   log likelihood = -8109.3117
Ordered probit regression                         Number of obs   =       6859
                                                  LR chi2(11)     =     588.91
                                                  Prob > chi2     =     0.0000
Log likelihood = -8109.3117                       Pseudo R2       =     0.0350

------------------------------------------------------------------------------
      health |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |  -.0056614    .003921    -1.44   0.149    -.0133464    .0020236
        age2 |  -.0000358   .0000398    -0.90   0.369    -.0001139    .0000423
     _Isex_2 |   .1133844   .0373295     3.04   0.002     .0402199    .1865488
_Itopqual3_2 |   .0691763   .0826651     0.84   0.403    -.0928444    .2311969
_Itopqual3_3 |  -.0504849     .08567    -0.59   0.556     -.218395    .1174252
_Itopqual3_4 |  -.1730846    .080075    -2.16   0.031    -.3300286   -.0161406
_Itopqual3_5 |  -.1893723   .0780889    -2.43   0.015    -.3424238   -.0363208
_Itopqual3_6 |  -.3217053   .0921638    -3.49   0.000    -.5023429   -.1410677
_Itopqual3_7 |  -.2502062    .097808    -2.56   0.011    -.4419063   -.0585061
_Itopqual3_8 |  -.5067183   .0812331    -6.24   0.000    -.6659322   -.3475045
      height |   .0085651   .0020132     4.25   0.000     .0046193     .012511
-------------+----------------------------------------------------------------
       /cut1 |   -1.47487   .3508896                     -2.162601   -.7871385
       /cut2 |  -.7184513   .3490263                      -1.40253   -.0343723
       /cut3 |   .1835116   .3488053                     -.5001342    .8671575
       /cut4 |   1.393712   .3490541                      .7095789    2.077846
------------------------------------------------------------------------------
```

Also, suppose you find that a variable increases the probability of being in the top category and the bottom category. It is difficult to know what to conclude from this kind of result. Often researchers will divide the variable into two categories (e.g. very goood and good health vs all other categories). OLS is a potential alternative if there are a large amount of categories.

A useful written command is "spost" which is aimed at the analysis of this type of categorical data. It provides a number of diagnostic tests, as well as tools for interpretation. We will make use of it to graph the predicted probabilities for our outcome. `ssc install spost`. We obtain the predicted probabilities for each of our outcome categories by height `prgen height, gen(height1)`, and graph these with the following code. This is useful as it is apparent from this that height increases the chances of being in the top category and reduces the chances of being in the "fair" category. Interestingly, it appears to have little effect on the probability of being in the bottom two groups.

```
twoway connected heighp1 heighp2 heighp3 heighp4 heighp5 heighx, scheme(s1mono) ///
title(''Predicted Probabilities - Health Condition by Height'', span) ///
subtitle(''Ordered Logit Model'', span) ytitle(Probability) ///
legend(lab(1 ''Pr(Very Bad)=1'') lab(2 ''Pr(Bad)=1)'') lab(3 ''Pr(Fair)=1'') ///
lab(4 ''Pr(Good)=1'') lab(5 ''Pr(Very Good)=1'')) xtitle(Height)
```

Figure 11: Ordered Logit Predicted Probabilities



6.2 Multinomial Regression

Now suppose we are interested in the determinants of housing conditions. We first examine this variable `tab tenureb`. We remove the missing values `mvdecode tenure, mv(-9 -8)`. It would not make sense to run an OLS model here as there is no clear way to rank these alternatives. For this we use the multinomial logit or multinomial probit model. We first tell Stata to use married as the base outcome for marital status `char marstatb[omit]2` and run our model `xi:mlogit tenureb hhsize i.sex age age2 i.marstatb` and for probit `xi:mprobit tenureb hhsize i.sex age age2 i.marstatb`. We will not run this command as it generally takes some time.[36] The interpretation of this output is a little different. Again remember that we can only interpret the significance and direction of these variables. The coefficients are like pairwise logit comparisons, they refer to the effect of a variable of being in that category relative to the base category. Note that we have 3 categories, but output for only 2. So for example, women are more likely to own and have a mortgage relative to the base (renting). Household size has a positive effect on renting relative to having a mortgage, but not for owning. As with the ordinal case, we can compute the marginal effects for any of the three categories. `mfx2, replace` and export the results `outreg2 using multinomial, excel replace`. As before, we will also examine the results graphically with the spost command.

[36]The MNP should be considered if you are concerned about the IIA assumptions with MNL.

Table 14: OLS and MFX for Ordinal Data

	(1)	(2)
VARIABLES	OLS	Logit MFX For Very Good Health
age	-0.00472	-0.000611
	(0.00307)	(0.00147)
age2	-2.92e-05	-2.89e-05*
	(3.13e-05)	(1.50e-05)
_Isex_2	0.0917***	0.0428***
	(0.0290)	(0.0139)
_Itopqual3_2	0.0670	0.0167
	(0.0628)	(0.0310)
_Itopqual3_3	-0.00878	-0.0284
	(0.0654)	(0.0304)
_Itopqual3_4	-0.114*	-0.0652**
	(0.0610)	(0.0268)
_Itopqual3_5	-0.117**	-0.0761***
	(0.0594)	(0.0265)
_Itopqual3_6	-0.223***	-0.115***
	(0.0709)	(0.0269)
_Itopqual3_7	-0.171**	-0.0873***
	(0.0755)	(0.0308)
_Itopqual3_8	-0.389***	-0.180***
	(0.0621)	(0.0236)
height	0.00679***	0.00320***
	(0.00156)	(0.000758)
Constant	3.273***	
	(0.271)	
Observations	6859	6859
R^2	0.087	

Standard errors in parentheses

*** p<0.01, ** p<0.05, * p<0.1

This time we will look at the predicted probabilities for men by age `prgen age, x(_Isex_2=0) f(20) t(80) gen(male)`. The probability of having a mortgage unsurprisingly decreases with age, and increases for the other two options.

```
twoway connected malep1 malep4 malep2 malex, scheme(s1mono) ///
title(''Predicted Probabilities - Home Ownership For Men'', span) ///
subtitle(''Multinomial Logit Model'', span) ytitle(Probability) ///
legend(lab(1 ''Pr(Own)=1'') lab(2 ''Pr(Mortgage)=1)'') lab(3 ''Pr(Rent)=1'')) ///
xtitle(Age)
```

Figure 12: Multinomial Logit Predicted Probabilities

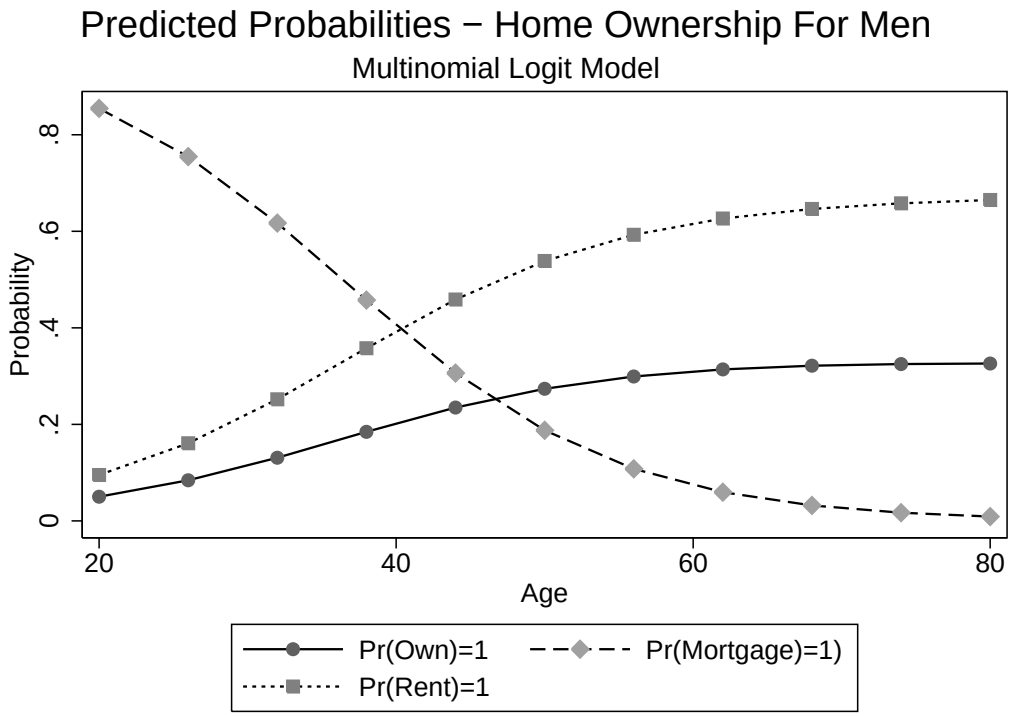


Table 15: Multinomial Logit Output

```
i.sex             _Isex_1-2           (naturally coded; _Isex_1 omitted)
i.marstatb        _Imarstatb_1-7      (_Imarstatb_3 for marstatb==2 omitted)
Iteration 0:   log likelihood = -7729.6211
Iteration 1:   log likelihood = -6337.5541
Iteration 2:   log likelihood = -6233.3063
Iteration 3:   log likelihood = -6227.7351
Iteration 4:   log likelihood = -6227.7068
Iteration 5:   log likelihood = -6227.7068
Multinomial logistic regression                 Number of obs   =       7271
                                                LR chi2(20)     =    3003.83
                                                Prob > chi2     =     0.0000
Log likelihood = -6227.7068                     Pseudo R2       =     0.1943
------------------------------------------------------------------------------
     tenureb |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
Own          |
      hhsize |  -.0236389    .032346    -0.73   0.465    -.0870359    .0397581
     _Isex_2 |    .199169   .0679491     2.93   0.003     .0659912    .3323468
         age |  -.0012009   .0148197    -0.08   0.935     -.030247    .0278451
        age2 |   .0010277   .0001468     7.00   0.000       .00074    .0013155
_Imarstatb_1 |   1.337345   .2719465     4.92   0.000     .8043398     1.87035
_Imarstatb_2 |   1.326905   .1286428    10.31   0.000      1.07477     1.57904
_Imarstatb_4 |  -.0765315    .284506    -0.27   0.788    -.6341531    .4810901
_Imarstatb_5 |  -.0434973   .1650001    -0.26   0.792    -.3668915     .279897
_Imarstatb_6 |   .3076225   .1870105     1.64   0.100    -.0589114    .6741564
_Imarstatb_7 |   -.146967   .1565473    -0.94   0.348     -.453794      .15986
       _cons |  -3.364784   .4225025    -7.96   0.000    -4.192874   -2.536694
-------------+----------------------------------------------------------------
rent it      |
      hhsize |  -.0507241   .0267455    -1.90   0.058    -.1031444    .0016962
     _Isex_2 |   .1605255   .0628003     2.56   0.011     .0374392    .2836117
         age |  -.1082971   .0123948    -8.74   0.000    -.1325904   -.0840037
        age2 |   .0015481   .0001318    11.74   0.000     .0012897    .0018065
_Imarstatb_1 |   .5218535   .1950537     2.68   0.007     .1395553    .9041517
_Imarstatb_2 |   1.565435   .1134457    13.80   0.000     1.343085    1.787784
_Imarstatb_4 |   1.862286    .190297     9.79   0.000     1.489311    2.235262
_Imarstatb_5 |    1.58454   .1345246    11.78   0.000     1.320877    1.848204
_Imarstatb_6 |     1.2927   .1937198     6.67   0.000     .9130164    1.672384
_Imarstatb_7 |   1.373128   .1063721    12.91   0.000     1.164643    1.581613
       _cons |   .0742578   .3289097     0.23   0.821    -.5703933     .718909
------------------------------------------------------------------------------
(tenureb==Mortgage is the base outcome)
```

Table 16: MFX for Multinomial Logit

VARIABLES	(1) Own	(2) Mortgage	(3) Rent
hhsize	-0.000749	0.00940	-0.00865
	(0.00596)	(0.00582)	(0.00535)
_Isex_2	0.0262**	-0.0443***	0.0181
	(0.0120)	(0.0134)	(0.0119)
age	0.00784***	0.0143***	-0.0221***
	(0.00239)	(0.00291)	(0.00220)
age2	8.15e-05***	-0.000323***	0.000241***
	(2.32e-05)	(3.07e-05)	(2.23e-05)
_Imarstatb_1	0.251***	-0.219***	-0.0317
	(0.0634)	(0.0394)	(0.0417)
_Imarstatb_2	0.108***	-0.320***	0.212***
	(0.0253)	(0.0185)	(0.0250)
_Imarstatb_4	-0.165***	-0.275***	0.440***
	(0.0235)	(0.0279)	(0.0350)
_Imarstatb_5	-0.141***	-0.236***	0.377***
	(0.0168)	(0.0228)	(0.0267)
_Imarstatb_6	-0.0647***	-0.209***	0.274***
	(0.0199)	(0.0355)	(0.0351)
_Imarstatb_7	-0.136***	-0.198***	0.334***
	(0.0188)	(0.0198)	(0.0245)
Observations	7271	7271	7271

Standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1

7 Panel Data

7.1 Panel Set Up

This tutorial presents the basics of panel data analysis, using a sample of employed workers in the British Household Panel dataset.[37] It is available for download from the UK Data Archive http://www.data-archive.ac.uk/. First we load the data `use bhps1, clear`. Note that this is quite a large dataset. If we `browse`, we see that the data is in the incorrect "long" format so we need to use the reshape command to fix this. `reshape long mlstat jbstat hlstat smoker jbsat fisit caruse age qfedhi hhsize hhtype ncars fiyr fihhyr jbhrs, i(pid) j(year)`. Now we have a unique identifier ("pid"), a time variable ("year"), and our outcomes.

As always, we first label and tidy our data.

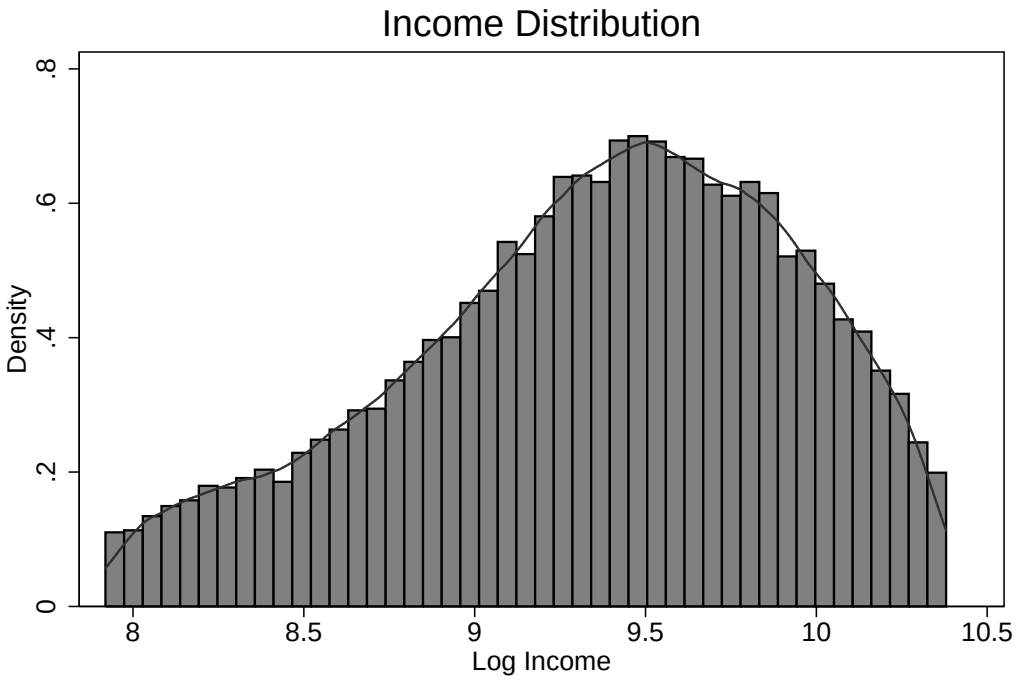
```
lab var mlstat ``Marital Status''
lab var jbstat ``Job Status''
rename  hlstat health
lab var health ``General Health''
lab var smoker ``Smoker''
lab var jbsat ``Job Satisfaction''
lab var fisit ``Financial Situation''
lab var caruse ``Use A Car''
lab var age ``Age''
lab var qfedhi ``Highest Educational Qualification''
lab var hhsize ``Household Size''
lab var hhtype ``Type of Household''
lab var ncars ``Number of Cars''
lab var fiyr ``Annual Income''
lab var fihhyr ``Household Annual Income''
lab var year ``Wave''
lab var jbhrs ``Hours Worked Per Week''
gen age2=age^2
lab var age2 ``Age Squared''
```

Suppose we are interested in the relationship between income and job satisfaction. We first examine our job satisfaction variable `tab jbsat`. There are over 40,000 observations on this variable in the data; remember that this will involve information on the same individuals over time. We will ignore missing values and individuals who didn't answer. First we check these values `codebook jbsat` and `labelbook kjbsat`. Alternatively, the user written command `lablist` is useful, which you can install with `ssc install lablist`. Then we can recode these to missing values `replace jbsat=.  if jbsat<1`. And check `tab jbsat`. We also do this for missing values in our health and education variables `mvdecode health, mv(-9 -1)` and `mvdecode qfedhi, mv(-9 -8)`. Let's examine our income variable `hist fiyr`. This suggests the presence of outliers which we may wish to exclude from our analysis. We first remove missing negative values `replace fiyr=.  if fiyr<0`. One way to do this is to exclude the bottom and top 5% of observations. First we find these values using the "centile" command. `centile fiyr, centile(5 95)` and recode these values as missing `replace fiyr=.  if fihhyr<2500 | fihhyr>32000`. Now we have a much more sensible distribution `hist fiyr`. We take the log of income `gen logincome=ln(fiyr)` and graph the distribution `hist logincome, kdensity scheme(s1mono) title(Income Distribution) xtitle(Log Income) caption(''Source:  BHPS'', span)`.

We can also examine our job satisfaction variable `tab jbsat` and graph it `catplot jbsat sex, percent blabel(bar, format(%8.1f)) title(Job Satisfaction) subtitle(By Gender)`. We can obtain the average income for each category `tabstat logincome, by(jbsat)`. We notice that those in the highest category for job satisfaction have the lowest income. We can use a scatterplot to confirm that the relationship is non-linear.

[37] This is a fantastic resource covering 18 waves, see http://www.iser.essex.ac.uk/survey/bhps for more details.

Figure 13: BHPS Income




```
graph twoway (scatter logincome jbsat if logincome>6, jitter(6)) ///
(qfit logincome jbsat if logincome>6), scheme(s1mono) legend(lab(2 ''Quadratic Fit'')) ///
ytitle(Log Income) title(Income and Job Satisfaction) caption(''Source: BHPS'', span)
```

If we look at the health variable, `tab health` and `codebook health`, we notice that the highest level of health corresponds to "1" and the lowest to "5". This may be confusing so we reverse this `recode health (1=5)(2=4)(4=2)(5=1)`. Don't forget to change the labels `labelbook khlstat` and `label define khlstat 1''Very Bad'' 2''Bad'' 3''Fair'' 4''Good'' 5''Very Good'', modify`.

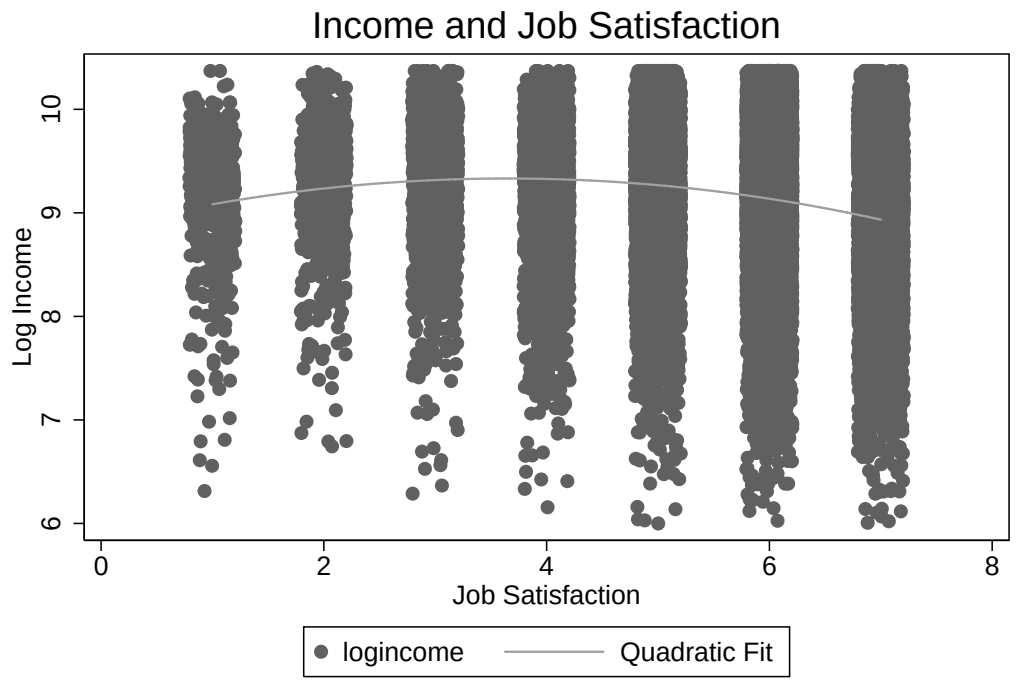
We will now start our regression analysis. We could estimate a simple OLS model `xi:reg logincome sex age age2 i.jbsat i.health i.year i.qfedhi` with income as a function of gender, age age squared, job satisfaction health and education, with controls for survey year. This involves ignoring the fact that we have information on the same individuals over time, which may not be ideal as you can imagine.

If we try to run a panel model `xi:xtreg logincome sex age age2 i.jbsat i.health i.year i.qfedhi, re` we get an error message. Like with the time series case, we must inform Stata that we have a panel, and identify our time and identifier. `xtset pid year`. However, before running the model, we should examine the panel aspects of the data. The `xtdescribe` command we obtain the characteristics of our data.

We have a strongly balanced panel (that is all individuals are present in all waves), with 6429 people tracked over 11 years. We can check that this gives us over 70,000 observations in total `bysort pid: gen no=_N` and `tab no`. "xtsum" is another useful command. Remember that there is variation within individuals as well as across individuals in a panel. We examine this for our job satisfaction variable `xtsum jbsat`.

Time invariant variables have no variation within individuals over time `xtsum sex`. This will be important when we examine the fixed effects model. Another command is "xttrans" which gives the transition probabilities for a variable. `xttrans jbsat`. See table 19. So if you are in category 7, you have a 50/50 chance of remaining there next period.

Figure 14: BHPS Income by Job Satisfaction



Source: BHPS

Table 17: xtdescribe Output

```
     pid:  10007857, 10014608, ..., 19140509                 n =       6429
    year:  1990, 1991, ..., 2000                             T =         11
           Delta(year) = 1 unit
           Span(year)  = 11 periods
           (pid*year uniquely identifies each observation)

Distribution of T_i:   min      5%     25%       50%       75%     95%     max
                        11      11      11        11        11      11      11
     Freq.  Percent    Cum. |  Pattern
 ---------------------------+-------------
     6429    100.00  100.00 |  11111111111
 ---------------------------+-------------
     6429    100.00         |  XXXXXXXXXXX
```

Table 18: xtsum Output

```
Variable         |      Mean   Std. Dev.       Min        Max |    Observations
-----------------+--------------------------------------------+----------------
jbsat    overall |  5.413611   1.348332          1          7 |     N =   40011
         between |              1.05996          1          7 |     n =    6274
         within  |             .9841251   .0136113   10.14088 | T-bar = 6.37727
```

Table 19: xttrans Output

Job Satisfacti on	Job Satisfaction 1	2	3	4	5	Total
1	20.69	15.71	12.45	11.88	12.26	100.00
2	8.15	15.54	20.76	9.94	18.73	100.00
3	3.57	8.31	24.72	14.58	24.92	100.00
4	2.73	4.29	14.00	23.48	28.49	100.00
5	1.10	2.59	8.26	10.64	35.05	100.00
6	0.57	1.25	3.67	3.87	19.40	100.00
7	0.74	0.56	1.25	1.62	6.97	100.00
Total	1.60	2.71	6.99	7.45	21.52	100.00

Job Satisfacti on	Job Satisfaction 6	7	Total
1	19.92	7.09	100.00
2	22.93	3.95	100.00
3	20.91	2.99	100.00
4	22.43	4.57	100.00
5	37.85	4.51	100.00
6	60.99	10.25	100.00
7	40.04	48.82	100.00
Total	45.24	14.48	100.00

It is also likely that the Y variable is correlated over time. We can confirm this by looking at the correlation within income over time `pwcorr logincome l.logincome l2.logincome l3.logincome, sig`. Note that we are using a time series operator, all of which also work in a panel context. As expected, income is serially correlated. Not that the `pwcorr` command uses parwise deletion of missing values, whereas `corr` uses listwise.

7.2 Panel Data Is Special

It is important to realise that panel data differs from cross sectional and time series data for a number of reasons. In our standard cross sectional model we have our outcome as a function of other variables across individuals (or firms, countries etc). $y_i = \alpha + \beta X_i + \epsilon_i$. In the context of panel data we have our outcome as a function of other variables across individuals and time $y_{it} = \alpha_i + \beta X_{it} + \epsilon_{it}$. There are others, but the most important advantage of panel data is that it can be used to deal with the problem of unobserved heterogeneity. Individuals differ due to omitted variables which can often not be measured. This can bias estimation in other contexts. The error term for an individual i in time t now consists of two parts $u_{it} = \alpha_i + \epsilon_{it}$. The first is something that makes that individual different, but does not change over time (α_i). The second is the usual random error term in any particular period that we are familiar with from our cross sectional models (ϵ_{it}). The main econometric choices in panel analysis relate to how you model the α_i component. These can be thought of in terms of intercepts. For an illustration, see Kennedy (2003). Consider the problems with OLS in this context. Firstly, the standard errors in this model are incorrect as they are almost certainly correlated within individuals over time. If you had a large positive error in the first period, you are likely to have a large positive error in the second. We can correct this problem by clustering the standard errors at the individual level. `xi:reg logincome sex age age2 i.jbsat i.health i.year i.qfedhi, vce(cluster pid)` and save the results for the coefficients we're interested in `outreg2 age age2 sex _Ijbsat_2 _Ijbsat_3 _Ijbsat_4 _Ijbsat_5 _Ijbsat_6 _Ijbsat_7 using panel, excel replace ctitle(OLS(Clustered))`. We include a dummy variable for each year to capture anything unique to that time period. We can test whether the cofficients are jointly 0 with the `testparm` command. We see that being in the highest category for job satisfaction appears to be associated with a reduction in income. However, in this case (referred to as pooled OLS), you are imposing everyone to have the same intercept $\alpha_i = \alpha$. As we have stated, individuals are likely to differ on some unobserved characteristic(s), so this assumption is unlikely to hold. We can easily test the assumption of a common slope.

An alternative is to assume that the unobserved characteristics are random, and are not correlated with any of our X variables and consequently the period specific error term. I.e. α_i is not correlated with ϵ_{it}.

7.3 Random and Fixed Effects

We will now implement our random effects estimator with the "xtreg" command. `xi:xtreg logincome sex age age2 i.jbsat i.health i.year i.qfedhi, re` and export the results `outreg2 age age2 sex _Ijbsat_2 _Ijbsat_3 _Ijbsat_4 _Ijbsat_5 _Ijbsat_6 _Ijbsat_7 using panel, excel append ctitle(RE)`. The output is shown in the table below (the coefficients have been omitted). After an RE regression we can test whether POLS is an appropriate model with the command `xttest0`. The null hypothesis is that there is a common intercept, so in this case we strongly reject and proceed with the RE model.

Note that the model explains 34% of the variation in income overall, but 27% of the variation within individuals over time, and 36% of the variation across individuals. σ_u is the standard deviation of the individual time invariant error term (α_i), while σ_e is the standard deviation of ϵ_{it}. Rho essentially refers to the correlation of the error term within individuals over time. Notice that Stata kindly reminds you about the assumption of X and u_i (or α_i) being uncorrelated.

If we believe that the individual error term is correlated with the time varying term, then the random effects model is not appropriate and will produce biased estimates. This can be thought of as a missing variable problem. For example, in this model we may believe that personality determines both income and job satisfaction. A way to deal with this is to use changes within individuals over time.

Table 20: Test For A Common Intercept

```
Breusch and Pagan Lagrangian multiplier test for random effects

        logincome[pid,t] = Xb + u[pid] + e[pid,t]

        Estimated results:
                         |       Var     sd = sqrt(Var)
                 ---------+-----------------------------
                logincome |   .6037161       .7769917
                        e |   .1510079       .3885974
                        u |   .2929869       .5412827

        Test:   Var(u) = 0
                                 chi2(1) = 33273.67
                              Prob > chi2 =     0.0000
```

Table 21: Random Effects Output

```
Random-effects GLS regression                   Number of obs      =     33021
Group variable: pid                             Number of groups   =      5990

R-sq:  within  = 0.2797                         Obs per group: min =         1
       between = 0.3613                                        avg =       5.5
       overall = 0.3478                                        max =        10

Random effects u_i ~ Gaussian                   Wald chi2(33)      =  13946.18
corr(u_i, X)       = 0 (assumed)                Prob > chi2        =    0.0000

     sigma_u |  .38981447
     sigma_e |  .25416102
         rho |  .70169989   (fraction of variance due to u_i)
```

Table 22: Fixed Effects Output

```
Fixed-effects (within) regression               Number of obs      =     33021
Group variable: pid                             Number of groups   =      5990

R-sq:  within  = 0.2867                         Obs per group: min =         1
       between = 0.0432                                        avg =       5.5
       overall = 0.0968                                        max =        10

                                                F(32,26999)        =    339.13
corr(u_i, Xb)  = -0.0551                        Prob > F           =    0.0000
     sigma_u |  .54079082
     sigma_e |  .25416102
         rho |  .81908033   (fraction of variance due to u_i)
```

This eliminates the influence of all time invariant characteristics of the individual, including unobserved heterogeneity. The intuition is as follows. As personality doesn't change over time (let's just assume it doesn't for now!), it cannot explain the change in the outcome over time. Of course it may explain the level, but this doesn't matter when we're looking at first differences or deviations from a mean. This is commonly referred to at the fixed effects estimator, as it is equivalent to adding a fixed effect (dummy variable) for each individual. `xi:xtreg logincome sex age age2 i.jbsat i.health i.year i.qfedhi, fe` and export the results `outreg2 age age2 sex _Ijbsat_2 _Ijbsat_3 _Ijbsat_4 _Ijbsat_5 _Ijbsat_6 _Ijbsat_7 using panel, excel append ctitle(FE)`. Notice that the overall explained variance is much lower.

Table 22 compares these estimates across the different approaches. We see that the coefficient on job satisfaction category 7 (completely satisfied) is -.04 (i.e. being in this category reduces income by 4%, remember it is measured in logs) for POLS, -.03 for RE, and -.025 for FE, and is also only marginally significant. So taking account of unobserved heterogeneity reduces the effect substantially. It is important to understand how the FE coefficient is generated. Only movement from one category of job satisfaction to another (changes over time within individuals), is used to estimate the effect on income. People who move categories are generally referred to as "switchers".

7.4 The Hausman Test

The drawback of the FE estimator is that it is less efficient than the RE estimator (i.e. higher variance), and also the fact that you cannot recover the coefficients on time invariant characteristics. So if the assumption of no correlation between the individual error and explanatory variables holds, then you should use Random Effects. Fortunately there is a way of testing which estimator is more appropriate in any given situation. This is based on the fact that under the null hypothesis of random individual effects the estimators should give similar coefficients. The Hausman test can be implemented by comparing the estimates from the two models. Suppose we run a simple case `xi:xtreg logincome i.jbsat, re`. We need to save these estimates so we can use them for comparison `estimates save RE`. Now we run our FE model `xi:xtreg logincome i.jbsat, fe` and save `estimates save FE`. Now we test `hausman FE RE, sigmamore`.

```
                ---- Coefficients ----
             |      (b)          (B)            (b-B)     sqrt(diag(V_b-V_B))
             |       FE           RE         Difference          S.E.
-------------+------------------------------------------------------------
   _Ijbsat_2 |    .0603609     .0606981       -.0003372        .0027453
   _Ijbsat_3 |    .0712668     .0786652       -.0073983        .0027158
   _Ijbsat_4 |    .0326557     .0330731       -.0004174        .0028292
   _Ijbsat_5 |    .0671614     .0700455        -.002884        .0027512
   _Ijbsat_6 |    .0518018     .0504758         .001326        .0028042
   _Ijbsat_7 |   -.0425275    -.0607399        .0182125        .0031464
------------------------------------------------------------------------------
                           b = consistent under Ho and Ha; obtained from xtreg
            B = inconsistent under Ha, efficient under Ho; obtained from xtreg

    Test:  Ho:  difference in coefficients not systematic

                  chi2(6) = (b-B)'[(V_b-V_B)^(-1)](b-B)
                          =      193.25
                Prob>chi2 =      0.0000
```

In this case, we can reject the null hypothesis that the coefficients are equal, therefore we conclude that the FE model is correct.

7.5 Dynamics

We have already seen that there is a high degree of correlation in our income variable, and we may wish to account for this with a more dynamic model, for example by including income in the previous period (y_{t-1}) as a regressor. Unfortunately the FE estimator is biased (RE is worse here as α_i is incorporated into ϵ_{it}) in the case where T is small (T>30 is recommended). We run this specification for comparison only `xi:xtreg logincome l.logincome sex age age2 i.jbsat i.health i.year i.qfedhi, fe`. Note again the use of the lag operator (`l.logincome`). A potential solution is to use an instrumental variables approach. The details are not straightforward, and require further assumptions about the structure of the error term. We implement the Arellano and Bond estimator for illustration `xi:xtabond logincome age age2 i.jbsat i.health i.year i.qfedhi`, which uses the second lag as an instrument for the first difference.

Table 23: Comparison of Panel Data Estimators

VARIABLES	(1) OLS(Clustered)	(2) RE	(3) FE	(7) FE Lag	(8) FE ABond
L.logincome				0.365***	0.346***
				(0.00576)	(0.0158)
sex	-0.464***	-0.511***			
	(0.0113)	(0.0111)			
age	0.0569***	0.0642***	0.0604***	0.0414***	0.0350***
	(0.00259)	(0.00179)	(0.00767)	(0.00688)	(0.00867)
age2	-0.000618***	-0.000694***	-0.000724***	-0.000373***	-0.000253***
	(3.09e-05)	(2.06e-05)	(2.53e-05)	(2.51e-05)	(5.76e-05)
_Ijbsat_2	0.0247	-0.00271	-0.00615	0.00882	0.00542
	(0.0238)	(0.0152)	(0.0153)	(0.0140)	(0.0155)
_Ijbsat_3	0.0772***	0.00947	0.000456	0.0163	-0.00175
	(0.0220)	(0.0135)	(0.0136)	(0.0126)	(0.0142)
_Ijbsat_4	0.0359*	0.00368	0.00129	0.0166	-0.00257
	(0.0217)	(0.0133)	(0.0135)	(0.0127)	(0.0142)
_Ijbsat_5	0.0792***	0.0113	0.00234	0.0210*	-0.00471
	(0.0210)	(0.0127)	(0.0129)	(0.0121)	(0.0137)
_Ijbsat_6	0.0479**	-0.00313	-0.0101	0.0139	-0.0108
	(0.0208)	(0.0126)	(0.0127)	(0.0120)	(0.0136)
_Ijbsat_7	-0.0434*	-0.0301**	-0.0258*	0.00844	-0.0201
	(0.0225)	(0.0131)	(0.0133)	(0.0127)	(0.0145)
Constant	8.470***	8.541***	8.129***	4.896***	5.152***
	(0.0915)	(0.0577)	(0.273)	(0.321)	(0.366)
Observations	33021	33021	33021	25314	17458
R-squared	0.366		0.287	0.413	
Number of pid		5990	5990	4916	4145

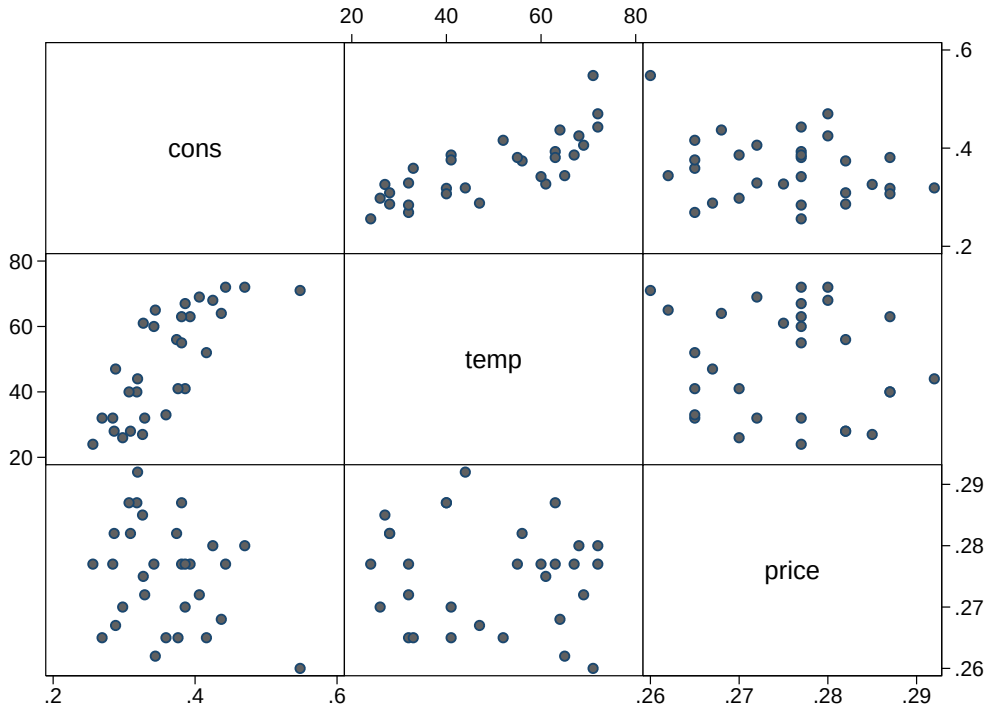
Robust standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1

8 Instrumental Variables

8.1 Endogeneity

This tutorial will provide an introduction to Instrumental Variables regression. One of the basic assumptions required for the unbiasedness of OLS is exogeneity, that there is no correlation between the X variables and the error term in the model. This can arise under a number of scenarios, including omitted variables, simultaneity and measurement error. Common examples include the effect of schooling on wages and the effect of aid on growth. In these circumstances IV can be used to produce consistent estimates of the effect of the endogenous variable on the outcome. The idea is to find some variable (Z) that predicts the X variable, but not the Y. This isolates the exogenous part of X, which we can use to examine the true causal effect of X on Y. In the case of the schooling example, this could be a change in compulsory schooling leaving age which exogenously (i.e. is outside the decision making capacity or characteristics of the individual) increases the amount of schooling received. Here we will illustrate the technique using a simple example of how openness affects growth. First we load the data `use openness, clear` and have a preliminary look at it `sum` and `describe`. `sum pcinc open land`. If we graph scatterplots of these three variables we see that there are a number of outliers, so we remove these from our analysis `graph matrix pcinc open land, scheme(s1mono) title(''Income, Openness and Area'')`.

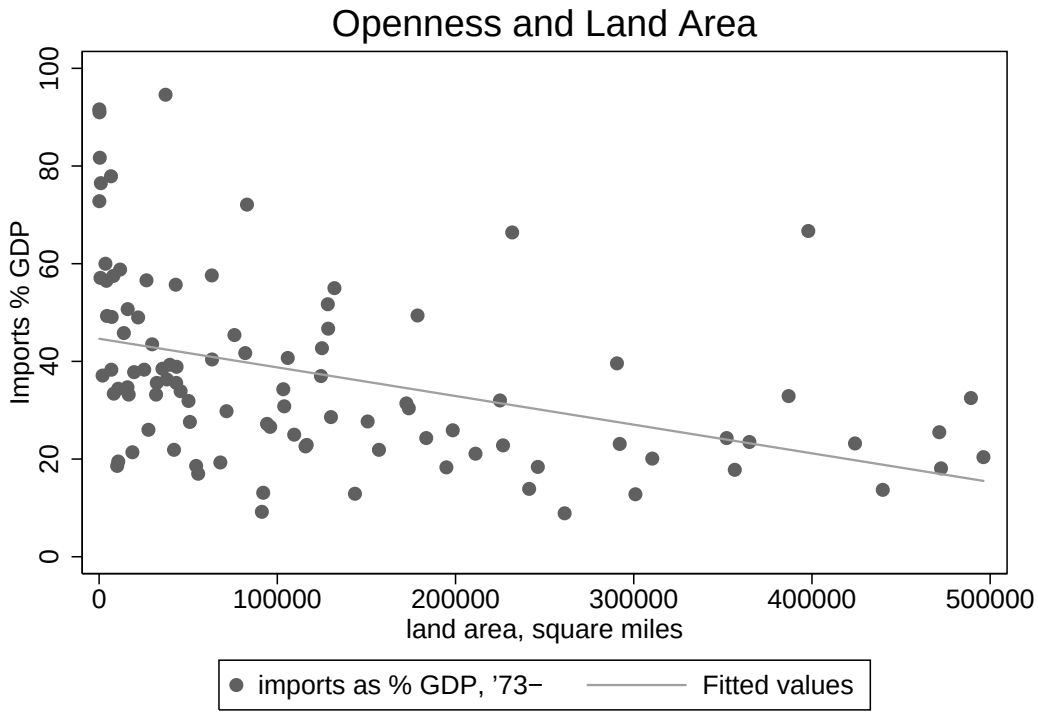
Figure 15: Graph Matrix for Openess, Area and Income Per Capita



`replace pcinc=. if pcinc>15000` `replace open=. if open>100` `replace land=. if land>1000000`. We are primarily interested in the relationship between income per head and openness, so we graph that relationship `twoway(lfit pcinc open)(scatter pcinc open)`. We see that there may be some suggestion of a week relationship in the descriptive data. We could run an OLS model with robust standard errors `reg pcinc open inf oil good, r` and save the results `outreg2 using iv, excel replace ctitle(OLS)`. On the basis of this model, 1% increase in the proportion of imports as a % of GDP raises income per head by $20.

Oil producers have higher income, as do countries with "good" data. Higher inflation has a negative effect. However it is important to recognise that the relationship between GDP and openness may be determined by another unobserved factor, or that higher income may cause higher levels of openness. In other words, openness is potentially endogenous, and therefore the model estimated above has no causal interpretation. To deal with this issue we need to instrument for this variable. The two criteria for an instrument are that the variable must be correlated with the X (openness) but not correlated with the Y (income). One possibility here is land area. If you believe that smaller countries are more likely to engage in international trade (because of economies of scale, specialisation etc.) but that area does not necessarily affect GDP per capita otherwise, then area is a valid instrument. The first assumption is testable, if we graph the relationship we can clearly see a significant negative correlation `twoway (scatter open land if land<600000) (lfit open land if land<600000), ytitle(Imports % GDP) title(Openness and Land Area) scheme(s1mono)`.

Figure 16: Openness and Area



If you do not believe the second assumption in this case, that is probably wise! We will just use this as an example of how to implement the procedure. IV papers ultimately depend on how much you believe this assumption, and you will encounter many you do not. Unfortunately there is no econometric test which will settle the matter entirely. We could look at the correlation between these three variables `pwcorr pcinc open land, sig`. First, openness is significantly negatively correlated with area. Secondly income is positively correlated with openness, although not significantly. The assumption of no correlation between Z and U refers to the relationship in the population, here we only have a sample, and a relatively small sample at that. However, if we have more instruments than exogenous variables, it is possible to test how close the corresponding sample moment is to zero. We will illustrate this later.

Table 24: Correlation Between Income, Openness and Area

```
             |    pcinc     open     land
-------------+------------------------------
       pcinc |   1.0000
        open |   0.0669   1.0000
             |   0.4796
        land |   0.1507  -0.3083   1.0000
             |   0.1095   0.0008
```

Table 25: OLS and IV Comparison

VARIABLES	(1) OLS	(2) IV
open	23.76*	49.29*
	(13.45)	(29.28)
inf	-13.82**	-9.959
	(6.684)	(7.702)
oil	5022***	4883***
	(1390)	(1358)
good	3679***	3482***
	(489.6)	(493.9)
Constant	563.0	-466.3
	(545.7)	(1174)
Observations	110	104
R^2	0.330	0.339

Robust standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1

8.2 Two Stage Least Squares

IV is often referred to as two stage least squares (2SLS). This is because the procedure can be conducted with OLS by regressing the endogenous variable on the instrument and obtaining the predicted values (the first stage). The second stage consists of regressing the Y variable on these predicted values and other exogenous variables. To illustrate the procedure, consider the simple case where we instrument for openness with land area. `reg open land` and `predict openhat` and `reg pcinc openhat`. It is recommended that you instead use the "ivregress" command or the user written "ivreg2". For one thing, the standard errors from the manual procedure are incorrect. `ivregress 2sls pcinc (open = land)`. You can see that the coefficients are almost the same here. We will implement the procedure for our full model with robust standard errors `ivregress 2sls pcinc inf oil good (open = land), vce(robust)` and save the results `outreg2 using iv, excel append ctitle(IV)`. So compared to our OLS model, the effect of openness has doubled, although it is still only significant at the 10% level. This is not surprising, we have a small sample, and the price we pay for using IV is less precision in our estimation. The lower correlation between the X and the Z, the higher these standard errors will be.

Table 26: Testing for Weak Instruments

```
First-stage regression summary statistics
  ------------------------------------------------------------------------------
               |            Adjusted      Partial       Robust
      Variable |   R-sq.       R-sq.        R-sq.       F(1,99)   Prob > F
  -------------+----------------------------------------------------------------
          open |  0.1801      0.1470       0.1415       17.6524    0.0001
  ------------------------------------------------------------------------------
  Minimum eigenvalue statistic = 16.3221

  Critical Values                       # of endogenous regressors:    1
  Ho: Instruments are weak              # of excluded instruments:     1
  ---------------------------------------------------------------------
                                    |    5%     10%     20%     30%
  2SLS relative bias                |         (not available)
  ----------------------------------+----------------------------------
                                    |   10%     15%     20%     25%
  2SLS Size of nominal 5% Wald test |  16.38    8.96    6.66    5.53
  LIML Size of nominal 5% Wald test |  16.38    8.96    6.66    5.53
  ---------------------------------------------------------------------
```

8.3 Weak Instruments, Endogeneity and Overidentification

It is also important to recognise that the validity of IV rests on asymptotic properties. IV estimates can be badly biased in situations where the instrument is weakly correlated with the X so it is important to test for weak instruments. This is generally done by examining the F statistic from the first stage regression. Low values are an indication of potentially weak instruments. We can examine this by using the "firststage" test `estat firststage, forcenonrobust` . We use the "forcenonrobust" option as we used robust standard errors and the theory for this test relies on the usual standard errors (so again caution is advised in this case). There is a lot of information in this output, but the important things to focus on are the F test for land "17.65" (a general rule of thumb is that if this is less than 10 there is cause for concern) and the minimum eigenvalue statistic "16.32", which is the Stock and Yogo test for weak instruments. We compare both of these to the critical value (at the 5% level) of "16.38". The test passes on the first statistic, but fails for the second. As we are using robust errors we should only reject the presence of weak instruments if our statistics greatly exceeded the critical values. As this is not the case, we should be sceptical about our results.

If you are confident about the validity of your instrument, it is possible to test whether the variable is in fact endogenous. This is based on the fact that if the variable is indeed exogenous there should be little difference between the OLS and IV estimates. This test is referred to as the Durbin-Wu-Hausman test, and is implemented with the following command `estat endogenous`. In this case we fail to reject the hypothesis that OLS and IV are equivalent, and therefore fail to reject the hypothesis that openness is exogenous. This may seem strange, as the IV estimate is twice that of OLS. The reason for this is again our small sample size and large IV standard errors. If we had more data, this could reduce the standard errors. Remember that this test is only valid with a valid instrument.

```
  Tests of endogeneity
  Ho: variables are exogenous
  Robust score chi2(1)           =  .467088  (p = 0.4943)
  Robust regression F(1,98)      =  .423783  (p = 0.5166)
```

Earlier we stated that it was possible to examine the validity of an instrument if the model was over-identified (i.e. more instruments than endogenous variables). This is based on the fact that the sample correlation between the instrument and error term should be close to zero if the instruments are valid. We will illustrate this using a second instrument, whether the country is a major oil producer. Clearly this will affect GDP as well as openness, so this is not a good instrument. Again we just using this example for illustration of how to implement the test. First we run our model with our two instruments, and the gmm option. The generalised method of moments is similar to the 2sls estimator, but is suitable for use in the overidentified case. `ivregress 2sls pcinc inf good (open = land oil), wmatrix(robust)`. We can now implement the test, which is often referred to as the "overidentifying restrictions test" or the "Hansen-Sargan" test. `estat overid`.

```
Test of overidentifying restriction:
Hansen's J chi2(1) = 4.05487 (p = 0.0440)
```

In this case we reject the null hypothesis that all instruments are valid, which is hardly surprising. Finally, there are a number of other important commands relating to IV that you may require in your analysis. The user written command "ivreg2" provides extensions to the Stata "ivregress" command. When the instrument is binary, the "treatreg" command is applicable, and when your Y variable is binary, the "ivprobit" command. With weak instruments, the LIML (limited information maximum likelihood) and JIVE (Jackknife IV) estimators can be used to provide better small sample estimates. The command for IV in a panel context is "xtivreg". There are also a number of different estimators for dealing with dynamic panel models, such as "xtabond" which implements the Arellano and Bond estimator.

9 Recommended Reading and References

Dublin Microeconomics Blog http://dublinmicroblog.blogspot.com/

UCLA Stata Website http://www.ats.ucla.edu/stat/stata/sk/default.htm

Princeton Stata Website:http://dss.princeton.edu/online_help/stats_packages/stata/stata.htm

Graphs 1 http://www.survey-design.com.au/Usergraphs.html

Graphs 2 http://www.ats.ucla.edu/stat/stata/library/GraphExamples/default.htm

A Visual Guide to Stata Graphics, Michael N. Mitchell http://www.stata.com/bookstore/vgsg.html

UK Data Archive http://www.data-archive.ac.uk/

Irish Social Science Data Archive http://issda.ucd.ie/

Econometric Analysis 6th Edition, William H. Greene http://pages.stern.nyu.edu/ wgreene/Text/econometricanalysis.htm

Econometric Analysis of Cross Section and Panel Data, Jeffrey Wooldridge
https://www.msu.edu/ ec/faculty/wooldridge/book2.htm

A Guide To Econometrics, Peter Kennedy http://mitpress.mit.edu/catalog/item/default.asp?ttype=2tid=9576

Microeconometrics Using Stata, A. Colin Cameron and Pravin K. Trivedi http://www.stata.com/bookstore/mus.html

Regression Models for Categorical Dependent Variables Using Stata, J. Scott Long and Jeremy Freese
http://www.stata.com/bookstore/regmodcdvs.html

Mostly harmless econometrics: an empiricist's companion, JD Angrist and JS Pischke
http://www.mostlyharmlesseconometrics.com/

University of Essex. Institute for Social and Economic Research. ESDS Longitudinal and University of Essex. UK Data Archive. ESDS Longitudinal, British Household Panel Survey: Waves 1-11, 1991-2002: Teaching Dataset (Work, Family and Health) [computer file]. 2nd Edition. University of Essex. Institute for Social and Economic Research, [original data producer(s)]. Colchester, Essex: UK Data Archive [distributor], November 2011. SN: 4901, http://dx.doi.org/10.5255/UKDA-SN-4901-2