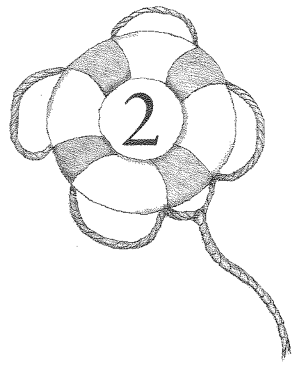


Data in and out of Stata

OPENING A STATA DATA FILE

To open an existing Stata data file you can either:

(1) use the **Open** icon on the toolbar and browse to the file location;
(2) use the pull-down menu **File** → **Open** and browse to the file location; or
(3) type in the Command window or a do file:
`use datafilename.dta,clear`

If you use method (1) or (2) when another data set is already open, Stata will warn you if those data have been changed and ask you either to clear the current data without saving the changes or to cancel the opening of the new data set. If you do want to save the changes then cancel and save the open data before opening the new data.

In method (3), the **`clear`** option tells Stata to clear any existing data before opening the data file specified in the command. Only use the **`clear`** option when you are absolutely sure you want to clear out any open data in its current state. The **`clear`** option is not necessary when opening the first data file after launching Stata when no other data are open. Stata will return an error message and stop if the **`clear`** is omitted and other data are open. If you haven't changed the default colours for fonts, error messages are returned in red, and the error message will say (in red): `no; data in memory would be lost.`

Version 10 uses **`clear`** and **`clear all`** commands in different circumstances, and as you become more familiar with Stata you will be able to use the **`clear`** options more efficiently.

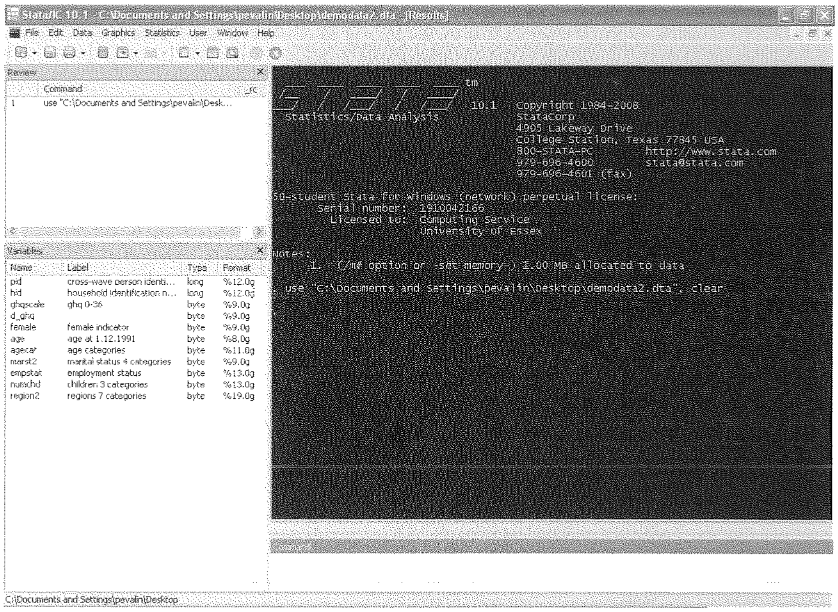
In (3) you do not have to specify the path to the data file if you have changed directories using the **`cd`** command to the location of the data file. In some cases it is preferable to read the master data from one location, such as when using a read-only data library, and then store smaller data files for analysis and output results in another location. For this kind of scenario we recommend using the **cd** command to point Stata to the location of the directory you wish to use to store your analysis data files and results, and then when opening the master data specify the whole path. An example of the series of commands where you wish to read master data from a data library but put all other files in a project folder on your local drive would be:

```
cd "C:\projects\project_a"
use "M:\datalibrary\masterdata.dta",clear
```

The first line instructs Stata to use the project_a folder on your C drive as the default directory, so unless otherwise specified all **use** and **save** commands retrieve or place files in that folder. The second line retrieves the master data file from the data library and opens it in Stata.

You will notice that Stata repeats the **cd "C:\projects\project_a"** command in the Results window in yellow. Unlike messages returned in red, which are error messages, yellow messages are result messages which give you feedback on what you have just done. You will also see that the **cd** and **use** commands are entered in white in the Results window, as this is the default colour for inputs (what you type). When you start producing results in the Results window, you will see that the majority of type here is green, as that is the default results colour.

After you have opened a data file you will see a list of variables in the Variables window. The variable names are listed on the left while the labels are on the right. Sometimes you have to expand the right-hand side of this window to see all the variable labels, particularly if they are quite long. Providing you have your windows set up in a similar way, the screen shot shows you how your screen may look after opening a data file. If your data are blank in the area for variable labels, this means that there were no labels assigned to the variables or that they were lost when you transferred your data into Stata. We show you how to add these later in the book. Also note that the command to open the data is repeated in the Results window and in the Review window. What you will see in the Review window depends on whether you are typing the command into the Command window or using a command in a do file. If it is the former, the command you typed will appear there. If it is the latter, you will see a line in the review window that starts with do but is then followed by a file directory, which is pointing to the location of the do file being used.


Stata/IC 10.1 - C:\Documents and Settings\pevalin\Desktop\demodata2.dta - [Results]
File Edit Data Graphics Statistics User Window Help
Review
Variables
C:\Documents and Settings\pevalin\Desktop

```
   Command                                          _rc
1  use "C:\Documents and Settings\pevalin\Desk...
```

Name	Label	Type	Format
pid	cross-wave person identi...	long	%12.0g
hid	household identification n...	long	%12.0g
ghqscale	ghq 0-36	byte	%9.0g
d_ghq		byte	%9.0g
female	female indicator	byte	%9.0g
age	age at 1.12.1991	byte	%8.0g
agecat	age categories	byte	%11.0g
marst2	marital status 4 categories	byte	%9.0g
empstat	employment status	byte	%13.0g
numchd	children 3 categories	byte	%13.0g
region2	regions 7 categories	byte	%19.0g

```
                                    tm
   STATA                  10.1   Copyright 1984-2008
  Statistics/Data Analysis       StataCorp
                                 4905 Lakeway Drive
                                 College Station, Texas 77845 USA
                                 800-STATA-PC        http://www.stata.com
                                 979-696-4600        stata@stata.com
                                 979-696-4601 (fax)

50-student Stata for Windows (network) perpetual license:
       Serial number:  1910042166
         Licensed to:  Computing Service
                       University of Essex

Notes:
      1.  (/m# option or -set memory-) 1.00 MB allocated to data

. use "C:\Documents and Settings\pevalin\Desktop\demodata2.dta", clear

.
```

You can see that the data file demodata2.dta has 11 variables. With a small number of variables it is possible to manage the whole file quite easily. However, if you use large data files, such as survey data, and you do not need to open the data with all of the variables then you can modify the **use** command to select which variables you choose. For example, if we only wanted to open the variables *female* and *empstat* from the demodata2.dta file then we would type in the Command window or do file:

```
use female empstat using demodata2.dta, clear
```

For different ways of exploring your data, see Box 2.1.

Keep and drop

There are two commands that allow you to specify the variables you wish to keep in the open data set once you have opened the

Box 2.1: Inspecting your data

If you type **codebook** into the Command window, you will be presented with information about variables' names, labels, and values. The default setting is to provide this information about all of the variables in your data set. If you wish to see the information from only a few then specify the variables after the command. For example:

```
. codebook sex age

-----------------------------------------------
sex
-----------------------------------------------

           type:  numeric (byte)
          label:  asex

          range:  [1,2]           units:  1
  unique values:  2             missing .:  0/10264

     tabulation:  Freq.  Numeric Label
                   4833        1 male
                   5431        2 female

-----------------------------------------------
age
-----------------------------------------------

           type:  numeric (byte)
          label:  aage12, but 81 nonmissing
                  values are not labeled

          range:  [16,97]         units:  1
  unique values:  81            missing .:  0/10264

       examples:  27
                  37
                  47
                  63
```

In version 10 the **codebook** command has an option – **compact** or **c** – that produces a briefer description of the variables. The same variables – *sex* and *age* – are shown in compact form below.

```
. codebook sex age,c

Variable   Obs  Unique     Mean  Min Max  Label
-------------------------------------------------
sex       10264      2 1.529131    1   2  sex
age       10264     81 44.41553   15  96  age at date
                                          of interview
-------------------------------------------------
```

The command **inspect** is also useful for checking data accuracy. Again the default is to provide all variables, so specify those for which you require the information. For example:

```
. inspect sex age

sex: sex                        Number of Observations
--------                   ---------------------------
                            Total  Integers Nonintegers
|    #       Negative          -         -           -
|  # #       Zero              -         -           -
|  # #       Positive     10264     10264           -
|  # #                    -----     -----       -----
|  # #       Total        10264     10264           -
|  # #       Missing           -
+----------               -----
1         2               10264
  (2 unique values)

   sex is labeled and all values are documented
   in the label.

age: age                        Number of Observations
--------                   ---------------------------
                            Total  Integers Nonintegers
| #          Negative          -         -           -
| # #        Zero              -         -           -
| # #        Positive     10264     10264           -
| # # #                   -----     -----       -----
| # # # #    Total        10264     10264           -
| # # # # .  Missing           -
+----------               -----
16        97              10264
  (81 unique values)

   age is labeled but 10264 values are NOT
   documented in the label.
```

full data. The command **keep** is followed by a list of variables you wish to keep in the open data. For example:

```
keep sex age educ
```

The command **drop** is also followed by a list of variables but this time those that you wish to remove from the open data. For example:

```
drop pid hid region-ghq
```

If there is a group of variables you wish to either **keep** or **drop** then you can just put the first and last variable as listed in the Variables window with a dash between and Stata will read this as including all variables between the two named in the order they are in the variables window. This notation can be used in other commands as well.

OPENING OTHER TYPES OF DATA FILES

There are a number of commands to help you import data in other formats but here we concentrate on probably the two most common formats: Excel spreadsheets and SPSS data files. See Box 2.2 for a software package which converts many different forms of data.

Box 2.2: Stat/Transfer

Stat/Transfer is a software package that converts data files from one format to another. There are too many formats to list here but all commonly used spreadsheets (Excel, Access, dBase etc.) and statistical packages (Stata, SPSS, SAS, Epi Info, etc.) are covered. See www.stattransfer.com and www.stata.com/products/transfer.html.

Excel spreadsheets

If you have data in an Excel spreadsheet and want to transfer it into Stata, from where you can save it as a Stata data file (.dta), you will have to go through a few intermediate steps. You can then either use the pull-down menus or use the **insheet** command in the Command window or a do file. These instructions assume you have your data organized in Excel with the variable names in the first row and then one case or respondent's data per row.

With your data open in Excel you need to save it as a text (tab delimited) file with a .txt extension. You can do this by using **File → Save As** and then selecting the type of file to save.

If you choose to use the pull-down menus in Stata to open the data then use

File → Import → ASCII data created by spreadsheet

and then click **Browse** to go to the location of the .txt file. There are a few options available to you at this stage but if your data are organized as above then you do not need to change any of the default settings. When you browse to find the .txt file, you need to make sure that the you can see **Files of type: All(*.*)** set in the lower panel of the **Open** dialogue box.

Click on **OK** and the data should open with your list of variables in the Variables window. You can visually check your data by clicking the **Data Browser** button (see Chapter 1) on the toolbar and inspect the spreadsheet.

If you wish to use the Command window or put commands in a do file use:

```
insheet datafile.txt, clear
```

This assumes the .txt file is in the default Stata directory or you can enter the path to the file.

To save your data in Stata format (.dta) see the **Saving data** section below.

SPSS data files

Until recently SPSS data files had to be converted manually, in a similar way to Excel spreadsheets, but now there is the command `usespss` which allows you to open an SPSS for windows (.sav) datafile directly in Stata. Type:

```
findit usespss
```

Then follow the instructions to install the command.

The `usespss` command works in a similar way to the usual Stata commands for opening existing files (see above):

```
usespss using "pathandfilename", clear
```

After opening the data it can be saved in Stata format (.dta). See the **Saving data** section below.

If you are using version 9.0 then you will need to obtain the free update to version 9.2 for the command to work. Type **`update query`** in the Command window and follow the instructions.

The latest versions of SPSS can save data in Stata format (.dta).

ENTERING YOUR OWN DATA INTO STATA

It may be the case that you have raw data that you want to enter directly into Stata. As mentioned in Chapter 1, look for the following two icons: ▦ ▣ or ▦ ▣ on the menu bar. The one on the left is the **Data Editor** and the one on the right is the **Data Browser**. The **Data Editor** is the one you will want to open in order to enter your own data. Once you launch the **Data Editor**, you can proceed to add your own data. The variables go in columns, with each row representing a single observation. Here, you can enter text and numbers, depending on the nature of your variables. To save the inputted data, click **Preserve** in the top left-hand corner. You will see that new variables are listed for you in the variables window. You will need to label the variables and their categories (if applicable) using commands that are covered in Chapter 3.

SAVING DATA

To save your data at any time under a new file name you can either:

(1) use the pull-down menu **File** → **Save As** and browse to the location you wish to save the file and enter its new name; or
(2) type in the Stata Command window or do file if you wish to save the data in the default directory or the one you have previously specified with the **`cd`** command:
`save newdata.dta`

If you wish to overwrite an existing data file with a modified version then add the option **`replace`**:

```
save newdata.dta, replace
```

When you are using a do file (and remember that we strongly recommend you progress to using them as soon as possible) it is preferable to use the **replace** option all the time. This is because if the option is not used and a data file with that name already exists then the **save** command will cause an error (returned in red and indicating that the file already exists) and the do file commands will stop at that point. If the option is used the first time a data file with that name is saved, Stata will report a (green) message that states that the file you indicated could not be found to be replaced. This isn't a problem – Stata is just telling you that it couldn't literally replace a file because one by that particular name wasn't already in existence, but it will save it anyway and continue with the next command in the do file. But the next time you run the file and make changes to the original data file, you will see that Stata will overwrite previous versions with the **replace** option. See Box 2.3 for a discussion of the importance of careful data file management.

Box 2.3: Overwriting your data

If you regularly use master data drawn from a read-only data library then you need only be concerned with whether to or when to overwrite your data for analysis files. This is because the system will not allow you to overwrite the master data. If you have your own data master files on your local drive you run the risk of accidentally overwriting those files, especially when you are in the early stages of getting to know a new software package.

After going to all the trouble of collecting, coding and entering your own data you need to guard against ruining that work. We recommend that you keep a master copy in a place detached from your local drive (CD-ROM, USB drive, or a networked remote drive) and, to be doubly sure, designate the copy of the master data that you are using on your local drive as read-only.

LOG FILES

We mentioned log files briefly in the previous chapter. A log file keeps a record of your commands and results during a Stata session. At first you may wonder why this is necessary as the results are shown immediately in the Results window on your screen. The Results window has a limited capacity, and while you might initially find that this is sufficient for your use (or you may increase the buffer size to be much greater), you will quickly need the capacity to permanently record your sessions as your data manipulation and analysis becomes more complex.

You need to explicitly tell Stata when to open a log file and to close it. This may seem odd to those familiar with SPSS, where an output window automatically opens when the first command is executed, but we believe it does give a greater degree of flexibility for complex series of analyses. In Stata you can open a log file at any time, close it, open it again to add further results, or open a new log file altogether. We find this particularly useful when we want to separate results from data manipulation or when preparing tables for a report where it is possible to produce a log file for the analysis for each table rather than one larger log file. Of course, you may prefer one large log file with clear annotations separating the different stages of the analysis, and Stata has the flexibility to manage either.

Log files come in two formats; both have their advantages and disadvantages. You specify the format you want by the extension to the log file name – this can be either **.log** or **.scml** – when you tell Stata to open a log file.

If you chose to use the log file with **.smcl** format (this is the default, so if you do not specify a file extension then you will get this format) then you can view this file by using the **View** option from the **File** pull-down menu in Stata and browsing to the log file. The **.smcl** log files have the same properties as the results that are displayed in the Results window. This file format also has the advantage that is can be copied and pasted easily into Excel and Word, which is a topic that we will return to in later in this chapter.

Log files with a **.log** extension are text files that can be viewed in any word processor. The downside is that it is not as easy to copy and paste tables to Excel, and to view them correctly you need to ensure that the font is one with equal spaces for each character such as **Courier**. To view your log file in Word, remember to select **Display all file types** when you are searching for your file, as its file extension is not .doc.

If you wish to only record what you type into the Command window, then you can open, close, turn on and off a command log file using **cmdlog** instead of **log**. You can have command logs open at the same time as normal log files if you wish.

Starting a log file

The command **`log using`** starts a log file, and you tell Stata the name you wish to call it (e.g. analysis) and which format you want. Assuming you wish to save the log file in the directory you have previously specified using the **`cd`** command, you would use:

```
log using analysis.log,replace
```

or

```
log using analysis.scml,replace
```

The **`replace`** option is used to overwrite the original file with a modified version. In a similar way to using the **`replace`** option with the **`save`** command, if **`replace`** is not specified and a file by that name exists, Stata will show an error and stop the do file. If **`replace`** is used and there is no file by that name Stata will show a warning the first time the file is created but then carry on with the next command in the do file.

If you are using the pull-down menus or the Command window and still want to keep a log file of your commands and results then you can use the log file icon on the toolbar – in version 9 and in version 10 – to open, suspend and close a log file.

When you open a log file, Stata automatically records the location of the log file, the type of log file and the time and date it was opened at the beginning of the file. For example:

```
. log using "C:\Documents and
              Settings\project_a\analysis.log"
       log:  C:\Documents and
              Settings\project_a\analysis.log
  log type:  text
 opened on:  19 Oct 2007, 11:32:41
```

Closing a log file

When you close a log file it is saved to the location specified when you opened it with the **`log using`** command. To close the log file, simply type:

```
log close
```

and Stata records the location of the log file, the type of log file and the time and date the log file was closed at the end of the file. For example:

```
. log close
      log:  C:\Documents and
              Settings\project_a\analysis.log
 log type:  text
closed on:  19 Oct 2007, 11:53:23
```

The **log close** command shuts the log file completely, and if you want to reopen it to add more information then you need to type **log using** with **append** (as below).

However, if you want to just turn off the log temporarily then you can use **log off** and then **log on** to turn it back on later in your analysis or in your do file. After the **log off** command is used Stata records when the log was paused:

```
 . log off
      log:  C:\Documents and Settings\
              project_a\analysis.log
 log type:  text
paused on:  19 Oct 2007, 12:03:35
```

After the **log on** command Stata records when the log was resumed:

```
   . log on
       log:  C:\Documents and Settings\
               project_a\analysis.log
  log type:  text
resumed on:  19 Oct 2007, 12:13:51
```

Adding to your log file

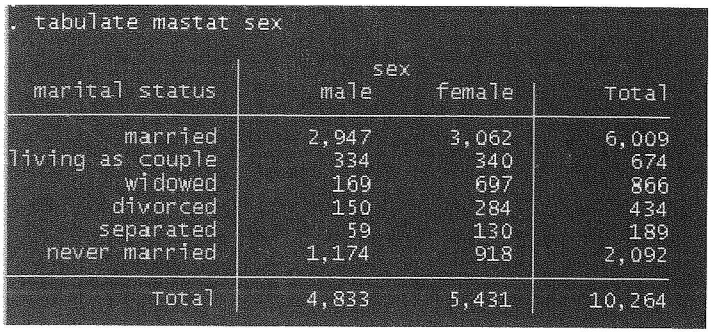
If you want to reopen an existing log file and add further results to it, rather than overwrite it as with the **replace** subcommand, you open the log file in the usual way but use an **append** option:

```
log using analysis.log,append
```

and Stata records the location, type and time and date the new results were added to the log file in the same way as when the log file was first opened.

COPYING RESULTS TO EXCEL AND WORD

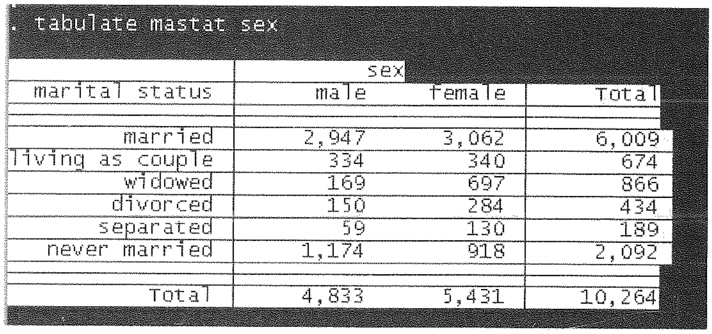
Tables and other forms of analysis results can be copied from the Results window straight into Excel and Word to create tables in your documents and spreadsheets. In this example, two variables, marital status (*mastat*) and gender (*sex*) from our example data are crosstabulated using the **tabulate** command (see Chapter 6 for a further discussion of this command):



```
. tabulate mastat sex

                   |          sex
  marital status   |      male      female |     Total
-------------------+-----------------------+----------
           married |     2,947       3,062 |     6,009
 living as couple  |       334         340 |       674
           widowed |       169         697 |       866
          divorced |       150         284 |       434
         separated |        59         130 |       189
     never married |     1,174         918 |     2,092
-------------------+-----------------------+----------
             Total |     4,833       5,431 |    10,264
```

Use the cursor and mouse to highlight the table:

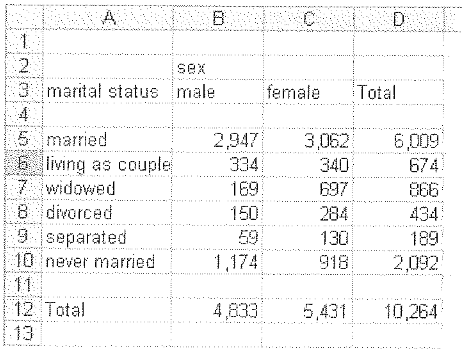


```
. tabulate mastat sex

                   |          sex
  marital status   |      male      female |     Total
-------------------+-----------------------+----------
           married |     2,947       3,062 |     6,009
 living as couple  |       334         340 |       674
           widowed |       169         697 |       866
          divorced |       150         284 |       434
         separated |        59         130 |       189
     never married |     1,174         918 |     2,092
-------------------+-----------------------+----------
             Total |     4,833       5,431 |    10,264
```

Either right-click and choose **Copy table** or use the **Edit** pull down menu to do the same.

In Excel, choose one cell and then right-click and choose **Paste** or use the **Edit** pull-down menu to do the same. The data will be automatically entered in their own cells in the Excel spreadsheet:



	A	B	C	D
1				
2		sex		
3	marital status	male	female	Total
4				
5	married	2,947	3,062	6,009
6	living as couple	334	340	674
7	widowed	169	697	866
8	divorced	150	284	434
9	separated	59	130	189
10	never married	1,174	918	2,092
11				
12	Total	4,833	5,431	10,264
13				

In Word, **Paste** the data into the document. It will appear spaced by tabs, and to convert it to a table you need to select the rows and go to **Table** → **Convert** → **Text to table**. The default settings will be OK to change the pasted data into a table in Word.

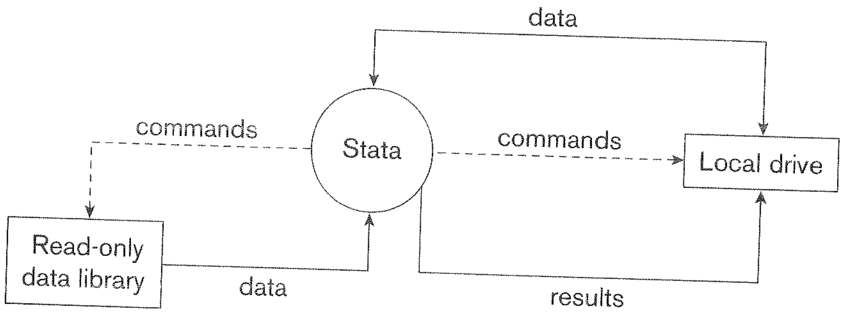
FILE MANAGEMENT

In Figure 2.1 we illustrate the main routes for your Stata commands (whether these are generated by a pull-down menu, Command window or do file), data and results when you are working with master data in a read-only library. This could easily be adapted in the case where your master data are held in a read-only folder on your local drive (see our recommendations in Box 2.3).

There are probably as many ways of organizing analysis files as there are researchers, but we would like to suggest two main ways as a starting point for your own file management system. We would like to stress the importance of adopting a systematic approach to file management and archiving whatever system you finally decide to use. This can pay dividends, as one of us (DP) found out (see Box 2.4).

The first approach is to group files by project. In this way all data files, do files and log files are in the same project folder. This allows you to specify the project folder in the **cd** command knowing that all files for that project are there to access, overwrite or save. This system works well provided the number of files does not become unwieldy. A way of dealing with large numbers of files is to adopt a system of prefixing your file names in order to group types of files. As the default way of showing files in a folder using Windows is alphabetical on the file name, if you use *do_*, *data_*, and *log_* to start names for do files, data files and log files respectively, then they will be shown in those groups. We also recommend that you try to be systematic in the naming of files,

Figure 2.1
Information flow schematic



Box 2.4: The importance of good file management

In 2003 I (DP) analysed data from an American panel study (the National Longitudinal Study of Adolescent Health[1] – AddHealth) for a paper which was eventually published in 2005.[2] In the summer of 2006 I was contacted by a researcher who was doing a meta-review of studies that had looked at cannabis use and depression.[3] As this was one of the aspects of our paper, the researcher had questions about the measures and methods we had used, as well as asking for some additional results that were not published, such as bivariate odds ratios where we had only published multivariate results.

When I think back to when I started data analysis and my rather haphazard file management, such a request then would have caused me hours of work trawling back through badly named and annotated files to find the particular statistics needed for this meta review. But this time, because I had adopted a file management system and had copiously annotated my do files, I was able to find the information and provide the unpublished results in a matter of minutes.

Most of the time the annotations on do files and their systematic use and storage will not be called on as mine were, but they are an investment which you may well be required to cash in sometime in the future. The time between submission to a journal and receiving reviews that may ask for additional analysis can be quite substantial, and you may have gone on to other projects in that time. Good annotations will allow you to get back into the analysis much faster than trying to work through each command line to try and remember what you had done all those many months ago.

[1] Udry, J.R. (2003) The National Longitudinal Study of Adolescent Health (Add Health), Waves I & II, 1994–1996; Wave III, 2001–2002 [machine-readable data file and documentation]. Chapel Hill, NC: Carolina Population Center, University of North Carolina at Chapel Hill.

[2] Wade, T.J. and Pevalin, D.J. (2005) Adolescent delinquency and health. *Canadian Journal of Criminology and Criminal Justice*, 47: 619–654.

[3] Moore, T.H.M., Zammit, S., Lingford-Hughes, A. et al. (2007) Cannabis use and risk of psychotic or affective mental health outcomes: a systematic review. *Lancet*, 370 (9584): 319–328.

especially do files and log files, so that it is obvious what stage of the project they relate to. For example, we might have one do file for extracting data from master files, one do file for manipulating those data to construct variables for analysis, and then three do files for analysis. In this case we would name the do files as follows:

do_extraction.do
do_construction.do
do_analysis_1.do
do_analysis_2.do
do_analysis_3.do

If you do not like the prefixing of the file names then you can click on the **Type** column top button in Windows, when in **List** or **Detail** view, and the files in that folder will be grouped by their extensions.

The second approach which we use for larger projects is to set up three subfolders in the project folder: one for do files, one for data, and one for log files. This way does not allow you to easily use the **cd** command to specify a single folder, but long paths can easily be copied and pasted in the do file editor. When using this system it is important to be able to identify which log files come from which do files. We use numbers to link them so that *analysis_1.do* produces *results_1.log*, and if there is more than one log file from a single do file we would use letters to distinguish them such as *results_1a.log*, *results_1b.log*.

USING THESE COMMANDS IN A DO FILE

Below we provide a starting template for a standard beginning and end of a do file. This starting template is for a single do file producing a single log file. Do files can be much more complex than this basic example, and we would expect you to develop and tailor your do files as you gain more experience with Stata. For example, you may have a number of log files open at the same time, in different formats, and then pause, resume and close each of them at separate times. In this case the **name(tempname)** option is very useful, but for now remembering that this is possible is enough to be going on with.

Explanations of the commands are annotated in the do file in the ways that can be actually used to annotate and note do files so they are more meaningful. You can add comments in do files which give an explanation of the commands you have written. These can be handy when you go back to a do file months later and can't remember why you did certain things. Indeed, you probably will not appreciate the importance of commenting until you have experienced this type of frustration. It is certainly better to have more comments than not enough! Any line that begins with an asterisk will be ignored by Stata as a command. If your comment extends for numerous lines, you will have to make sure that every line begins with *, or you can use a different technique for extensive commenting, writing /* at the beginning of the comment and */ at the end. Anything between these signposts is ignored. It is important, if you decide to use this latter method for extensive commenting, that you remember to put in your end of comment signifier */ or everything after the opening /* will be regarded as a comment – possibly the rest of your do file! Both types of commenting techniques are shown below. For additional commenting techniques, see **help comment**.

```
set mem 10m
version 10.1          /* not required but a good
                      habit to get into in
                      case you have any version-
                      specific commands */
capture log close     /* closes any log files
                      still open */
set more off          /* turns off the need to hit
                      space bar when the Results
                      window is full */
cd M:\projects\project_a\analysis

log using analysis_1.scml,replace
* the .scml is not formally required as it is
* the default format

use id sex age mastat educ using ///
   "K:\datalibrary\data_file",clear

* manipulation and analysis commands here

save analysis_file.dta,replace
* the .dta is not formally required
log close
```

Note that this is the first time we have used the /// notation to indicate that a command line goes over to the next line. This is only used in do files and not the Command window – see interacting with Stata in Chapter 1.

Another way of organising your do file would be to only open the log file when you have analysis results you wish to keep and exclude the data manipulation. In this case your do file might be structured as follows:

```
set mem 10m
version 10.1
capture log close
set more off
cd M:\projects\project_a\analysis

use id sex age mastat educ using ///
   "K:\datalibrary\data_file", clear

*first set of manipulation commands here

log using analysis_1.scml, replace

*first set of analysis commands here

log off

*second set of manipulation commands here

log on

*second set of analysis commands here and
*be sure to close log

log close

save analysis_file.dta, replace
```