

STATA 10 - SAMPLE SESSION

Cross-Sectional Analysis

**Short Course Training Materials
Designing Policy Relevant Research and
Data Processing and Analysis with STATA 10
1st Edition**

**Department of Agricultural Economics, Michigan State University
East Lansing, Michigan
January 2009**

Components of the Cross-Sectional Training Materials

Section 0 - Introduction to the Window structures for STATA 10 (Stata Results, Review, Variables and Stata Command Windows as well as the Do-File Editor). This section must be read before starting the sample session.

Section 1 - Basic functions

Section 2 - Table Lookup & Aggregation

Section 3 - Tables & Multiple Response Questions and Other Useful Commands

Section 4 - Graphs, tables, publications and presentations, how to bring them into word processor, and use of Survey commands.

Annexes

I - Frequently used Stata commands.

II - Several pages from the socio-economic survey of the smallholder survey in the Province of Nampula, Mozambique (NDAE Working Paper 3, 1992).

III - Computer analysis of survey data - File organization for multi-level data by Chris Wolf, MSU Department of Agricultural Economics. This document can be downloaded as a separate document in English or French at <http://www.aec.msu.edu/agecon/fs2/survey/index.htm>.

Acknowledgments

Funding for this research was provided by the Food Security III Cooperative Agreement between the Department of Agriculture Economics at Michigan State University and the United States Agency for International Development, Global Bureau, Office of Agriculture and Food Security.

SECTION 0 - File structure and Basic Operations for Stata 10	5
How Stata uses memory.....	7
The set memory command.....	7
How to set memory when STATA is started from an icon on the desktop.....	8
Increasing the amount of memory in the middle of a Stata session.....	9
The drop _all command.....	9
Types of files used by Stata and their extension names.....	9
Data files.....	9
Log files.....	9
The log using command.....	9
The cmdlog using command.....	10
The log close command.....	10
Do files.....	10
The doedit command.....	11
Discussion of the Windows used in STATA.....	12
The Do-file Editor.....	12
The Data Editor Window.....	13
The edit command.....	13
Saving the Stata Data File.....	16
The save, replace command.....	16
The Brower Window.....	16
The browse command.....	16
The Stata Results Window.....	16
The Command Window.....	17
The Viewer.....	17
Stata Graph window.....	17
Summary of the Basic File Types.....	17
SECTION 1 - Basic functions: Stata files, Descriptives and Data Transformations	19
Introduction.....	19
Data files and the working file.....	20
Working Directory.....	20
The cd command.....	20
Opening a data file.....	20
The use command.....	20
Describing the contents of a data file.....	22
The describe command.....	22
Data storage types.....	24
Display format.....	24
Labels.....	24
Documenting variables and labels.....	24
The labelbook command.....	24
–more–.....	25
The label list command.....	25
The codebook command.....	25
Generating descriptive statistics.....	26
Descriptive statistics - using one variable.....	27
Descriptives.....	28
The summarize command.....	28
Information returned by Stata commands.....	29
TABULATE - Frequencies.....	30
The tab1 command.....	31
The histogram command.....	31
Saving a graph to a file.....	31
The list command.....	32
Descriptive Statistics - using two or more variables.....	36
Two-way Tables with Categorical Variables (Cross-tabulation).....	36
The tabulate command.....	36
Summary statistics on a continuous variable for each value in a categorical variable.....	38
The by ... sort: summarize command.....	38
Data Transformations.....	40
Converting continuous variables to categorical variables.....	41

The generate command.....	41
The replace command.....	42
The label variable command.....	43
The label define command.....	44
The label values command.....	45
The recode function.....	46
SECTION 2 - Restructuring Data Files - Table Lookup & Aggregation	52
Restructuring Data Files.....	52
Step 1: Generate a household level file containing the number of calories produced per household.	56
Rename any key variables in both files to the same name.....	58
The joinby command.....	58
Compute total kilograms produced	60
The generate command.....	60
The drop command.....	61
Calculate the total calories produced	62
Select only staple food products	63
The keep if command.....	63
Create a new file which is a household level file rather than a household-product level file.....	64
The collapse command.....	64
Step 2: Generate a household level file containing the number of adult equivalents per household.	66
Create a variable with the adult equivalent for each person.....	67
The generate... if command.....	67
The replace... if command.....	67
Replace “missing values” with a mean value.....	69
Calculate the adult equivalents for the household.....	70
The collapse command.....	70
Step 3: Merge the two files created in steps 1 & 2 to compute calories produced per adult equivalent.	72
The merge command.....	72
Calculate the total calories produced per adult equivalent per household for the year.....	74
Computing quartiles	75
The xtile command using if	75
The foreach looping command.....	76
The levels command.....	76
Examples of the foreach looping command.....	79
SECTION 3 – Tables and Other Types of Analysis	87
Tables	87
The table command.....	89
Comparison of the commands summarize , tabulate and table	89
Print a table from the Viewer.....	92
Multiple Response Questions	93
1) Multiple dichotomy (yes/no questions)	93
The count command.....	93
The recode command.....	94
The egen command.....	94
The tabstat command.....	95
2) Multiple response	95
Other Types of Analyses	97
Weights.....	97
Indicator variables	98
Converting continuous variables to indicator variables.....	99
Converting categorical variables to indicator variables.....	100
SECTION 4 - Table and Graphs - how to bring them into a word processor, and	101
How to move Stata results into other applications.....	101
Tables.....	101
Copying tables from the Results window.....	102
Using Excel to create columns from the table.....	103
Graphs	103
Scatter plot using “ by ” subcommand.....	106
Overlaid graphs.....	106
Survey Estimation - Accounting for Design Effects.....	107
ANNEX I – Stata Commands	112
ANNEX II - Questionnaire	116

Stata 10 - SAMPLE SESSION

SECTION 0 - File structure and Basic Operations for Stata 10

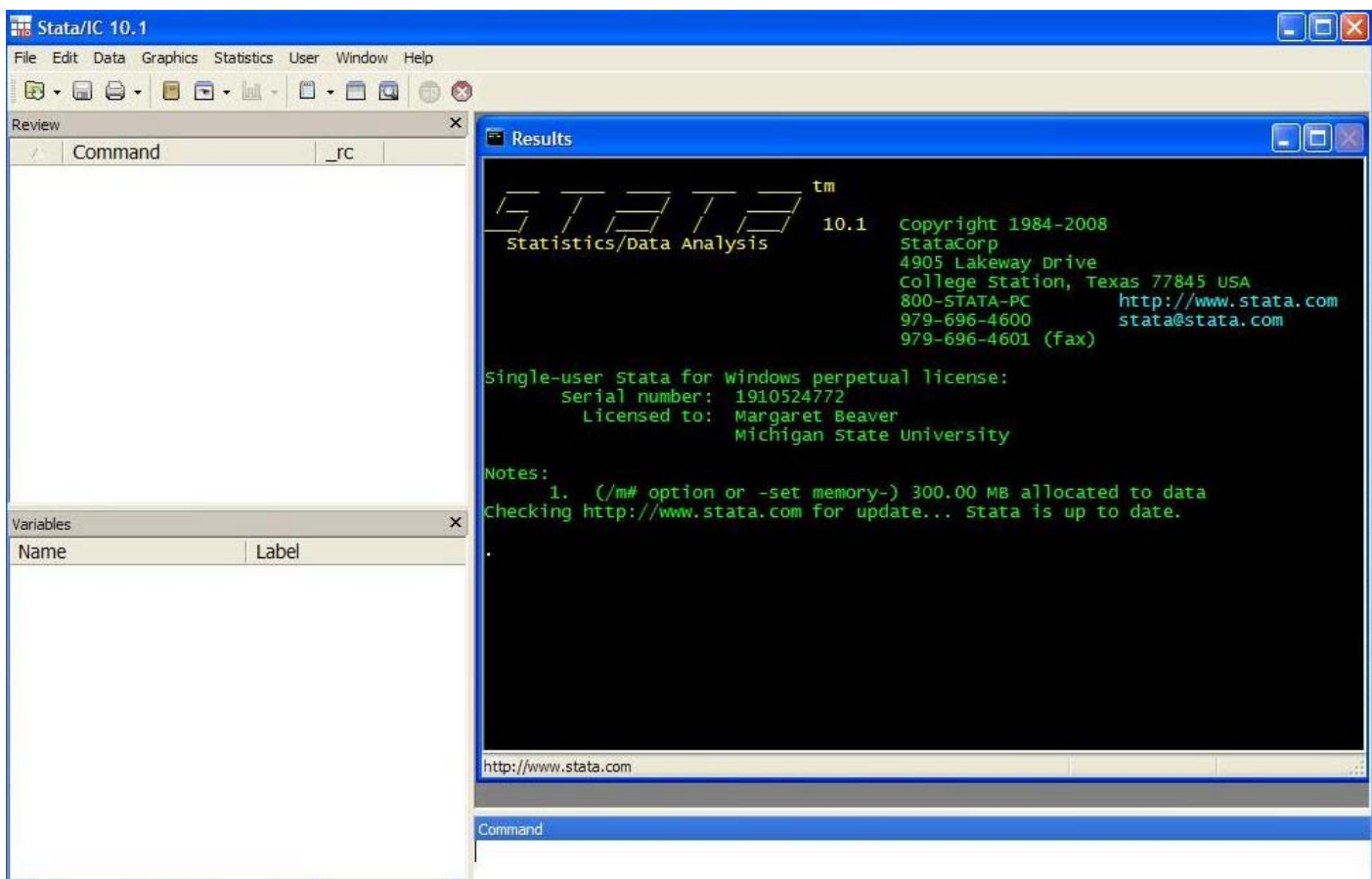
This section introduces the basic concepts of levels, the notion of cross-sectional analysis, and consequently, the methods of data organization. This section gives a brief description of the file structure of Stata , version 10. It is essential that you read through this section before starting the cross sectional session.

Overview

When you open Stata 10 for the first time, you will see four different windows within the program—

- the **Results** window (results of a command are displayed in this window),
- the **Review** window (commands submitted to the processor appear in this window),
- the **Variables** window (the list of variable names in the data set that has been opened) and
- the **Command** window (where commands can be typed, this is the “active” window at startup).

You can resize and reposition any of the windows in Stata. Below is an example of the default arrangement of the windows.



If you wish to rearrange the windows and keep your new arrangement, select

```

Edit
  Preferences
    Manage Preferences
    Save Preferences
    New Preferences Set

```

Type a name for the new preference set and click on **Ok**.

To return to the original arrangements, from the same menu choose

```

Edit
  Preferences
    Manage Preferences
    Load Preferences
    Factory Settings

```

Other windows are available, but are not opened at startup. These windows are:

- **Viewer** (used to view help files and log files, SMCL - markup and control language- files, and print log and other files. This window is not contained in the STATA 10 program window but stands alone and appears on the task bar as another icon.)
- **Data Editor** (where you can view the data you have loaded into the program's memory)
- **Do-file Editor** (text editor where you can build a "do" file, a file that contains commands that Stata can execute. This window is not contained in the STATA 10 window but stands alone and appears on the task bar as another icon.)

You can switch between the windows within Stata by using the **Window** choice from the **Menu**. Note that shortcuts are also listed, e.g. to switch to the **Command** window, you can press <Ctrl> 4, to switch to the **Variables** window, press <Ctrl> 6.

Version 10 of Stata provides menus to help the user. However, the user can also type all the commands in the **Command** window. Throughout this tutorial, if the action desired can be done using the menus, directions will be given on how to use the menus. The Stata command that will do the same action will also be given so that you become familiar with the commands. Stata provides a mechanism to paste commands into a do file that you can then execute. You can also copy the commands from the **Results** window and paste them into the **Do-file editor**. Another method is to copy commands from the **Command** window and paste them into the **Do-file editor**.

How Stata uses memory:

a) The set memory command

A data file must be loaded into memory before any analysis can be done. Stata/SE uses 10 megabytes of memory for data, Intercooled Stata uses 1 megabyte of memory and Small Stata uses 300 kilobytes of memory for data. You cannot change the amount of memory used for Small Stata. For the other versions the amount of memory can be temporarily changed or permanently changed. The command to change the memory is:

```
set memory [amount of memory]
```

example:

```
set memory
```

To check to see how much memory is being used and how much is remaining, use the following command:

```
memory
```

Before loading a file into memory, the result of this command in Intercooled Stata is:

```
-----
Details of set memory usage
  overhead (pointers)           0          0.00%
  data                          0          0.00%
-----
  data + overhead               0          0.00%
  free                          1,048,568  100.00%
-----
  Total allocated                1,048,568  100.00%
-----
Other memory usage
  system overhead               745,090
  set matsize usage              16,320
  programs, saved results, etc.  105
-----
  Total                          761,515
-----
Grand total                      1,810,083
```

| After loading a small file, the results are:

```

. use "c-qla.dta", clear
. memory

                                     bytes
-----
Details of set memory usage
  overhead (pointers)                6,096      0.58%
  data                               67,056      6.40%
-----
  data + overhead                    73,152      6.98%
  free                               975,416     93.02%
-----
  Total allocated                    1,048,568   100.00%
-----
Other memory usage
  system overhead                    745,090
  set matsize usage                   16,320
  programs, saved results, etc.       1,029
-----
  Total                              762,439
-----

```

One megabyte can be used up fairly quickly, so it is recommended that you set the memory at the beginning of the session to a larger size, e.g.

```
set memory 30m
```

```
:
```

b) How to set memory when Stata is started from an icon on the desktop

If you wish to have the memory already set when you start the program, you can edit the command that starts the program and add the parameter for memory.

1. Highlight the icon on your desktop, right click and select **Properties** from the choices.
2. In the **Comment:** box, add **/m30** (or whatever amount of memory you want to set it to) so that the command reads:

```
Stata/IC 10 /m30
```

3. When Stata is installed, the directory to look for data is specified as the directory where the program was installed. (See “Start in” box.) However, Stata remembers where you last opened a file and will use that reference when the program is started the next time.
4. If you have made any changes, click on **Ok** to save the changes.

The next time you start STATA from the icon, the memory will be set and the default directory will be set to whatever directory you have specified. If you start the program from the **Start, All Programs** menu, the memory parameter will not be set unless you modify that shortcut as well within the Stata 10 directory.

c) Increasing the amount of memory in the middle of a Stata session:

The `drop _all` command

`drop _all`

If you want to increase the amount of memory in the middle of your session, you will not be able to do so unless you close the data file using the command

Another option is to just close the Stata program and set the memory using the set memory command after you open the program and before you open a data file.

Types of files used by Stata and their extension names

1. Data files

- files containing data (Extension *.dta)

Data files have an extension of `.dta`. From the Stata 10 window, you can open a data file.

From the **Menu**:

Select **File**, then **Open**.

If you are not in the directory where your files are, change to the appropriate directory. Only files with an extension name of ".dta" will be listed.

From the **Command** window (if you are working in the correct directory), you can type:

use "name of file", clear

2. Log files

- commands and output (Extension *.SMCL)
Stata markup and control language
- commands and output (Extension *.log)
- ASCII text: commands only (Extension *.txt)

Stata can record a copy of the commands and the output from the commands in a "log" file. If you wish to record this information in a file, you must turn on the log. There are two types of logs:

1. **Log**: One records everything that you submit for execution and all the output resulting from the commands. You can specify one of two formats, either SMCL or ASCII text (log)

From the Menu: Select **File**, then **Log**, then **Begin**. You are prompted for a file name. The default extension is SMCL. The file is formatted in the Stata markup and control language. Type a name for the file and click on OK. If you prefer to record the information in ASCII text, then you would need to type the file extension of **.log**, e.g. `session1.log`.

The **log using** command

From the **Command** window, type:

```
log using session1, append
```

The above command opens a file to record the session and uses SMCL format. This file can only be opened in the Stata Viewer.

or type:

```
log using session1, append text
```

The above command opens a file to record the session and uses ASCII format. This file can be opened in any text editor or word processor.

The **cmdlog using** command

The other type of log file records only the commands and not the output from the commands. The command is

```
cmdlog
```

This command creates a file that records only the commands.

In the Stata Command window, type:

```
cmdlog using session1, append
```

A file is opened which is named "session1.txt", and information will be appended to anything that already exists in this file.

The **log close** command

To close the log, in the Command window, type

```
log close
```

Reminder: The log file that is written in SMCL format can only be opened in Stata. It is a specific format as mentioned earlier. If you want to share your commands and results from the log files with another person who might not have Stata, you should save your log files in the TEXT format with the extension of .log. Any editor or word processor can open this file. However, in the word processor, the font must be set to a fixed font, such as Courier New.

Otherwise, the output will be difficult to read.


-Stata commands (Extension *.do)

A ".do" file contains commands that Stata can execute. The "do" file is created in the Do-file Editor. The user can type commands or paste commands into the editor. Other ways to create a do file are:

a) You can create a log file that contains only the commands, using the "**cmdlog**" command, see above.

b) You can select the Review window, click the right mouse button and select "**Save Review contents**". The extension .do will be automatically added to the file name you enter into the "File name" box.

3. Do files

c) You can copy the command to the clipboard, using the option provided in the dialog box where commands are built.  and then switch to the **Do-file Editor** to paste the command.

d) You can copy commands from the **Results** windows into the **Do-file Editor** using **<Ctrl C>** to copy what you have blocked in the **Results** window and then switching to the **Do-file Editor** and pressing **<Ctrl V>** to paste the command that was copied from the **Results** window.

e) You can select the command from the **Review** window, which places it back into the **Command** window, where you can block the command, press **<Ctrl X>** (cuts the command from the **Command** window), switch to the **Do-file Editor** and press **<Ctrl V>** to paste the command.

Option c) may become your preferred method to build the do-file. Option e) is also useful.

Adding comments to document commands

Comments can be placed in the do-file as you copy and paste commands. Comments in a do file can start with an asterisk if the comment is one line. If the comment covers several lines use `/*` before the comment and end with `*/` so that STATA will not think the comments are commands. You can also use a double slash `//`. This option is useful if you want to add a comment after a command. Example of a comments are:


*** your name here and the date the file was created**

**/* do file to examine variables using the methods of
Tabulate and tab1 */**

describe // describing the variables in the file

Within the **Do-file Editor**, you can submit several commands at once. (In the **Command** window, only one command at a time can be submitted for execution.) You cannot send commands directly from the Stata **Command** window to the **Do-file Editor**. The command must be copied.

There are 2 ways to open the **Do-File** editor.

1. From the Button Bar, you can click on the “**Do-file Editor**” button . Another window opens which is the **Do-file** editor.

2. From the **Command** window you can type:

```
doedit
```

The `doedit` command




Discussion of the Windows used in STATA

A) The Do-file Editor


It is important to recognize the significance of the different types of files and to understand the various commands you use to create and access the files.

The **Do-file Editor** is the window where commands can be typed before they are submitted to the STATA processor. Commands can be **typed** directly into the **Do-file Editor** or you can copy the commands from the **Results** window and paste the commands into the **Editor**. There are four main uses of the **Do-file Editor**:

- To type commands directly into the **Do-file Editor** to be processed later by STATA,
- To send these commands to Stata10 for processing,
- To write or save these commands to a file for future use, and
- To retrieve files of commands that you have saved previously so that you can run them again without the need to rebuild the commands.

It is important to understand that the commands you put in the **Do-file Editor** will not be executed (no output will be produced) until you send the commands to the processor. The **Do-file Editor** is simply an area that helps you prepare the commands. To send the commands to the processor, you use the **Do** icon  in the **Do-file Editor** window toolbar. The **Do** command executes the commands in the current do-file. Another icon, called **Run**,  also executes the commands in the current do-file but does not show any output in the **Results** window. Choosing either one sends all the command(s) to the processor, which reads the commands written in the **Do-file Editor** and executes them. To send only specific commands, block the commands you want to send and select the **Do** icon .

When you have successfully completed each step in your analysis (or when you are ready to end a STATA 10 session, even if it was not completely successful) you should save the commands to a file for future use. To save the commands, make the **Do-file Editor** active and select **Save** from the **File** menu or click on the diskette symbol on the Tool Bar. A file created from the **Do-file Editor** is called the *command file*. It is a file containing only commands; it never contains any of the data you may be analyzing with the commands. You must save your data separately, as described in the following section. We suggest that you use the default *extension* of `.do` when naming command files. `Rep7.do`, `dem-all.do`, and `section1.do` are some examples.

By storing your commands to a `.do` file, you can retrieve, look at, or modify sets of commands and rerun them. To retrieve a do-file into the editor, open the **Do-File Editor** pull down the **File** menu and select **Open** or you can click on the “yellow” file folder  on the tool bar in the editor. Select the file you wish to open and click on **Open**. Once you have opened a specific file, you can use the commands from the file, without having to recreate or type them

again. If you make changes to the command file that you wish to keep, make sure you save them to disk again.

Caution: From **Windows Explorer**, if you double-click on a “.do” file, the Stata program will open and run all the commands in the do-file immediately. The do-file will not be opened. To open a “do” file from Windows Explorer, right click on the file name and choose “**edit**”. The application “STATA” will open and the “do” file will be opened in the **Do-File Editor**. When you have opened a do-file in this manner, STATA automatically executes the command

```
doedit ...\nameofdofile.do
```

which you can see in the **Results** window.

B) The Data Editor Window

Stata stores your data in a *data file*. In addition to the values themselves, a data file contains such things as variable labels and value labels, formatting information, missing-value specifications, notes, etc. Before you can do any data analysis in Stata 10, you must first tell Stata to open a Data file. Select **File** from the menu, select **Open**, highlight a data file (example: **c-hh.dta**) and click on


```
Open
```

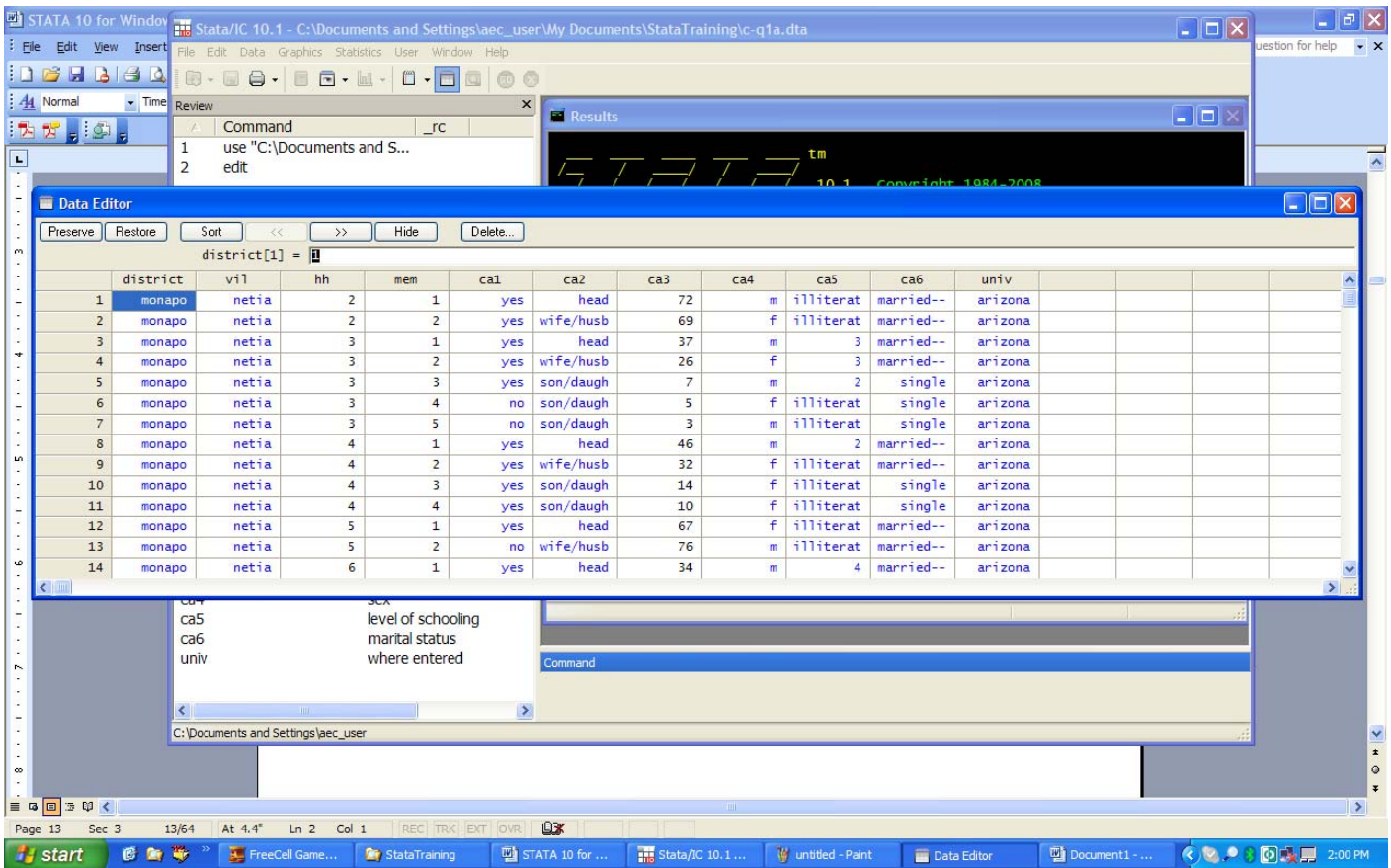
The command is immediately run. The data in the file are now available to be viewed in the **Data Editor** window. In the **Review** window you see the command that opened the data file. In the **Variables** window you see the list of variables that are available.

There are 2 methods that you can use to look at the data. The first opens the file in the **Data Editor** window. In this window you can manually change the data, so be careful when you use this method. The other method opens the data in a browser window where you cannot change any of the values, but you can sort and look at the data.

Open Data Editor window

The **edit** command

1. The first method to view the data is to open the **Data Editor** window. Click on the Data Editor button  or in the **Command** window, you can type **edit** and press <Enter>. If value labels have been assigned to the values in a variable, you will see the value label rather than the actual value. Below is an example of a data file with value labels displayed for some variables and values only for other variables.



Stata makes a backup copy of the data when you enter the Data Editor.

Preserve

The preserve button will update the backup copy of the data if you have made changes and want to continue to work in the data editor.

Restore

The restore button will replace the current data in the editor with the backup copy if you decide you don't want to keep any changes you've made.

Sort

The sort button will sort the variable that you have selected. It will not sort multiple variables, just the variable where you cursor is situated.

Exercise:

Exercise: Place your cursor in the variable column labeled **ca2** and click on the sort button.



This button will shift the current variable to be the first variable.



This button will shift the current variable to be the last variable.

The hide button will hide the current variable. It will still be in the data set, it just will not be visible.

Hide**Delete**

The delete button will give you 3 choices as to what you want to delete. You can delete the variable where your cursor is, you can delete the current observation or you can delete all observations that have the same value as the current observation.

Exercise:

Exercise: Change the value to 1 in the hh column where hh=3.

By default the value labels are displayed. To look at the values, <right-click> on the table and choose “**Hide all Value Labels**”.

Another option available with the <right-click> is to “Define/Modify Value labels” where a value labels can be defined, dropped or added.

Exiting the Data Editor

To exit the Data Editor, click on the “x” in the upper right hand corner of the Stata Editor.

When you exit the Stata Editor, a dialog box appears:

**Accept Changes**

saves the changes and exits the editor,

Discard Changes

restores the data to its original backup or “preserved” copy and exits the editor.

Don't Exit

allows you to continue working in the editor.

We do not want to keep the changes, so click on “**Discard Changes**”.

You will often get a data file, compute new variables, make transformations, and finally save the modified set of data to a new name to be used at another time. For example: you might retrieve a data file with land area per crop, add to it production per crop from another file, and then calculate yield. If you

want to use the new production and yield variables at a later time, you must make sure that the data file is saved with the new variables in it. Never save data that has been modified to the same file name.

Saving the Stata Data File

The `save, replace` command

To save a data file, close the **Data Editor** and then:

From the menus select
File, Save As... and enter the name.

From the **Command** window, you can also type

```
save "newfilename"
```

or, if you want to use the same name, type

```
save, replace
```


The same name as the file you opened will be used. In the **Results** window you will see documentation of the Stata commands that are equivalent to what you did in the **Data Editor**, e.g.

- preserve
- sort ca2
- replace hh = 1 in 2
- restore

The dash indicates the change was made in the **Data Editor**. The data were preserved first. We then sorted by `ca2`, then we changed a value in the variable called `hh` to 1 for the second record. The last line `"- restore"` indicates that the changes were not saved. The data were restored to its original state before we entered the **Data Editor**.

C) The Brower Window

The `browse` command

The second method to look at the data is to use the **"Browse"** mode. You cannot modify the dataset if you use this method. This method will prevent you from accidentally modifying the data. If you did not close the **Stata Editor**, you must close it first, before you go into the **Browse** mode. Click on the `browse`  button. In this window you can sort the data and also hide columns if you wish. To exit the **Browser**, click on the `"x"` in the upper right hand corner of the **Stata Browser**.

Note: *If the **Browser** is open or the **Data Editor** is open, the **Command** window is not available. You must close the editor or browser before you can type any commands or choose from the menus.*

D) The Stata Results Window

Stata 10 automatically writes all messages and output to the **Results** Window from the execution of your commands. For

example, if you run a tabulate command, then the frequency table will be written to the **Results** window. If you wish to save the information in the **Results** window you must remember to turn on a log file. See the explanation above on “Log files”.

E) The Command Window

The **Command** window is used to type commands directly. If you use the menus, the command is run immediately. The command is placed in the **Review** window. If you want to rerun a command that is in that window, click on the command. The command is placed in the **Command** window. To execute the command, press <Enter>.

Useful keystrokes within this window.

- <PageUp> recalls the last command run and places it in the **Command** window. If you continue to press <PgUp>, the next command above will be placed in the window.
- <PageDn> moves back down through the commands that appear in the **Review** window.
- <Esc> clears the contents of the **Command** window.

F) The Viewer

The **Viewer** in Stata is used to view help files and log files and to print these files. To enter the Viewer, click on **File, View....** The “**Choose File to View**” dialog window opens. You can type the name of the file or click on the

Browse

button. By default, the file type extension name is: SMCL Files (**.smcl**). Select the file you want and click on

Open

The file name is pasted into the dialog box where you can then click on

Ok

If you decide to use Help from the menus, the Help files are opened in the **Viewer**.

G) Stata Graph window

A graph is opened in its own window and is not stored in the **Results** window. If you wish to keep a graph, you can copy the graph to a word processing document or you can save the graph to a file. Right-click on the graph to see these options. A graph file has the extension **.gph**.

Summary of the Basic File Types

Do-file files

Do-file files (or command files) contain commands saved in the **Do-file Editor**. They do not contain output or data—only

commands. Do files are made accessible to Stata if you open the Do-file editor. Within the Do-file editor you can open a do-file.

Log files contain statistical output, data information and presentation generated by the Stata processor. They do not contain data. Log files are made accessible to Stata with a **File, View** command. The extension is *.smcl.

Data files contain data, including original survey variables plus any new variables created through various Stata commands such as the **generate** command. Data files are made accessible to Stata using a **File, Open** command from the menus or typing the command in the **Command** window.

Stata 10 SAMPLE SESSION

SECTION 1 - Basic functions: Stata files, Descriptives and Data Transformations

Introduction

This is a self-paced training aid designed to introduce the commands needed for some typical statistical survey analyses using **Stata 10**. This tutorial is intended to be a stand-alone training tool. To use it most effectively, you should ask a knowledgeable STATA user to help you get started and to answer questions as you work independently through the session. It can also be used as a guide for classroom training.

A copy of the questionnaire on which the data is based can be found in the Mozambique project 1992 **NDAE Working Paper 3: A Socio-economic survey of the smallholder survey in the province of Nampula: Research Methods**, copies of the three tables which were made available and can be found at the end of the manual in the annex section (for further information please contact Dr. Michael Weber at webermi@msu.edu). Four portions of the questionnaire are referenced, each of which has a corresponding Stata data file. Two other Stata data files are required for conversion of units of measure.

Questionnaire Section	Stata Data File
Main Household Section	c-hh.dta
Table IA: Household Member Characteristics	c-q1a.dta
Table IV: Characteristics of Production	c-q4.dta
Table V: Sales of Farm Products	c-q5.dta
Conversion factors for computing kilograms	conver.dta
Conversion factors for computing calories	calories.dta

This training consists of four sections, each of which should take approximately two hours. We recommend that you complete each section in a single sitting. These tutorial materials make the following assumptions:

- You know how to use Windows with a mouse
- The six data files listed above should be stored in a directory of your choosing on your hard disk.

Important: Always remember to *SAVE* the changes to the data after each exercise and section, using a **new** file name. Also, you may want to save Review window contents to a .do file if you have not been copying commands to a do file already. You may also want to save your log file created during each session.

Data files and the working file

Open your Stata software. If you have not read or completed **Section 0**, please do so now to clarify the concept of the **Command Window**, the **Review Window**, the **Results Window**, the **Do-file Editor** and the **Viewer**.

Working Directory

Data from questionnaires that has been entered into Stata are stored in what are called *data files*. If we want to work with a set of data, we must open the corresponding data file so that it is available to the program.

The `cd` command

The working directory is the directory where your data files are stored. You can use the `cd` command to change to the directory where you have placed the data files you want to use.

In the **Command Window** type:

```
cd "name of working directory"
```

Changing to the directory where the files are located eliminates the need to include the directory name in the **do-file** that we will be creating. If the directory you are changing to has spaces, the directory must be enclosed in quotes. Example of a directory name with spaces:

```
"C:\Documents and Settings\My Documents\data"
```

Opening a data file:
The `use` command

When a data file is opened, it is loaded from the disk into memory (the computer's "RAM"), making it the working file. This means that the data from this file is now available for you to use. Let's start with the data for Table IA: Household Member Characteristics. The data file that corresponds to this table is `c-q1a.dta`. To open this file, perform the following steps:


1. From the **File** menu, select **Open...**
*This will open the **Open File** dialog box.*
2. If you have run the "`cd`" command you should see a list of data files to be used with this tutorial. Select the file `c-q1a.dta`.
3. Click on the **Open**

button to open the file. The command appears in the **Review** window.

*In the **Review Window** you will see the text*

```
use "...c-q1a.dta", clear
```

“...” will be replaced with whatever the name of the directory is where you are working.

4. We want to create a **do-file** to save our commands. The command that was just executed appears in the **Results** window. Copy this command (block the command press <Ctrl-C>). Click on the button in the Tool Bar to open the **Do-File Editor**  and paste the command into this file (<Ctrl-V>). Note also that the **use** command you just ran has been written to the **Review** window. You could press <PageUp> which places the command in the **Command** window where you can copy it. Switch to the **Do-file editor** and paste the command.
5. We want to copy the “**cd**” command and paste it just above the “**use**” command. We also want to add comments to define what the purpose of the **do-file**. Above the commands that you just pasted, insert some lines. You can type:

```
/* session 1 - basic functions, descriptives */
/* “your name here” - “the current date here” */
   (example: /* beaver – 5 Jan 2009 */)
/* member level file */
```

Other commands that are important and should be included are command to close any log file that may be open, clear the memory work space and drop all macro variables. An example of the commands that should be added to the do-file are:

```
capture log close
log using name_of_log_file, replace

/* Purpose of do-file */
/* Author and date */
/* Tasks to be done in this do file */

/*program setup */

version 10
clear all
macro drop _all

/* change to directory where files are stored*/
cd "C:\Documents and Settings\aec_user\My
Documents\StataTraining"
```

6. Save the **do-file**. From the **File** menu in **Do-File Editor** select **Save As...**
7. Enter the filename **session1**
The .do extension will be added to the name automatically.

8. Click on **Save**.

The do-file is now saved to disk.

We have opened the household-member data file which is now the current file in memory.

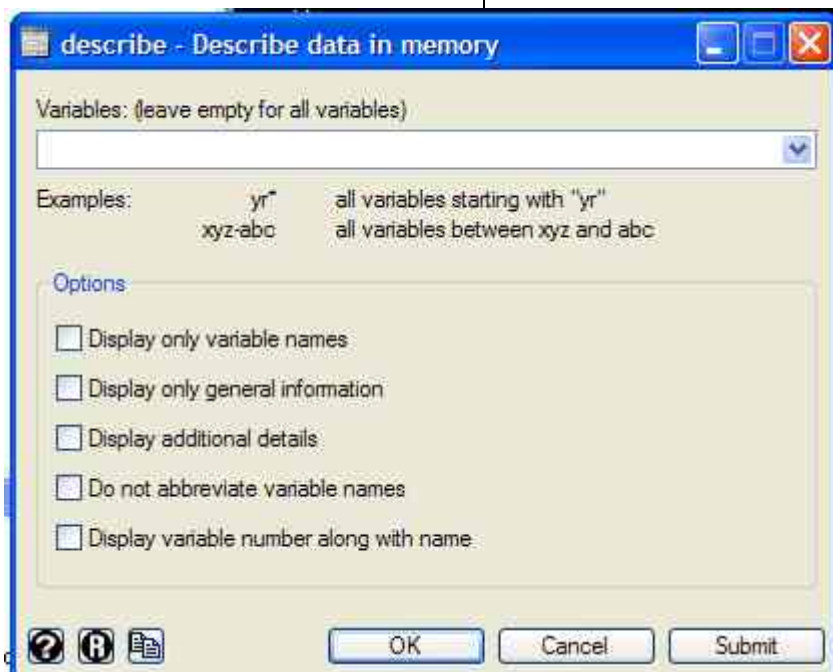
Describing the contents of a data file:

The describe command



A key piece of information we need to know about a data file is what variables it contains. We can find this out, along with other information, by using the **Describe data** command on the **Data** menu:

1. From the **Data** menu select **Describe data...**
2. There are several choices under this option: Select **Describe data in memory**. A dialog box opens:

There are several options in this dialog box.




At the bottom of the dialog box, there are three icons on the left. The first (a question mark) opens the “Help” screen to explain the options in the dialog box.

The second (an R)  resets the information in the dialog box so that nothing has been selected. The third icon  will copy the command to the clipboard. You can then switch to the do-file and paste the command into the do-file.

On the right hand side you have the choices to click on

Ok, Cancel or Submit. If you choose **Submit** the dialog box remains open so that you can select another option within the dialog box without having to

open the box again. If you choose **Ok** the dialog box closes. The command is automatically executed, whether you choose **Submit** or **Ok**.

We want a description of all variables; therefore, we can leave the list of variables blank. Before you click on **Ok**, click on . Switch to the do-file, and press <Ctrl V> to paste the command. Switch back to the dialog box and click on **Ok**.

In the **Results** window, you will see the description of the variables. To obtain the same results from the **Command** window type

describe

The output shows the file name, the number of observations, the number of variables, the size and then information about each of the variables, the storage type, the display format the value label and variable label.

```

Contains data from .....c-qla.dta
  obs:          1,524
  vars:           11
  size:         73,152 (93.0% of memory free)
-----
variable name    storage type  display format  value label  variable label
-----
district        float      %9.0g          district    district
vil             float      %9.0g          vil         village
hh             float      %9.0g          hh         household
mem           float      %9.0g          mem        member number
ca1           float      %9.0g          ca1        does this person work?
ca2           float      %9.0g          ca2        relation to head
ca3           float      %9.0g          ca3        age
ca4           float      %9.0g          ca4        sex
ca5           float      %9.0g          ca5        level of schooling
ca6           float      %9.0g          ca6        marital status
univ          float      %9.0g          univ       where entered
-----
Sorted by:

```

An explanation of each of the columns follows:

Data storage types

Storage type: Stata has 6 storage types:

- Float - real numbers, 8.5 digits of precision, width of 8 with 5 decimals, default unless another type is specified
- Double - real numbers, 16.5 digits of precision, width of 16 with 5 decimals
- byte - integer between -127 and 100
- int - integer between -32,767 and 32,740
- long - integer between -2,147,483,647 and 2,147,483,620
- strX - string indicating number of characters, Intercooled Stata maximum size is 244.

Display format

Display format: The display format is the third column which describes how the data are to be displayed. Stata will make an assumption with new variables so it is not always necessary to specify the format. Format information always begins with a percent sign “%”, to indicate the start of the format information. Refer to the User’s Guide, Chapter 15.5 for more details. In this example, the 9 describes the width of the variable. After the decimal the 0 indicates no fixed number of decimals will be displayed. If you wished to see only 2 decimals, the example would be %9.2g. The letter following indicates what type of format:

- e - scientific notation, e.g. 1.00e+03
- f - fixed format, e.g. 1000.03
- g - general format
- c - optional along with either f or g will display a comma, e.g. 1,000.03

Labels

Variable label: Label describing the variable.

Value label: If the variable has value labels the name of the label appears in this column. Stata assigns a name to the label which contains the values and labels. The label is then applied to the variable. More will be said about value labels later.

Documenting variables and labels:

There are several ways to view the labels and values for variables:


The labelbook command

If you wish to see what labels have been defined for specific values for the variables that have value labels as indicated above, you can run the command to create a codebook of the labels. From the menus:

1. From the **Data** menu select **Labels /Label Values**
2. Select **Produce codebook of value labels**
3. Click on the icon to copy the command to the clipboard, and then click on **Ok**

In the **Command** window you can also type



to `labelbook` obtain the same results. This command describes only those variables with value labels. It is a good command to document these variables. This output is quite long. You will see `-more-` at the bottom of the **Results** screen `-more-` indicates there is more information to be displayed, but the display has paused so that you can view the first part of the output. You will need to click on `-more-` several times to see the complete output. To continue to the next screen, click the `<spacebar>` or you can click on the `-more-` or you could also click on the green button on the tool bar - . This button is green only if there is more to be viewed in the **Results** window.


If you wish not have the output displayed a screen at a time, you can turn this feature off. The command is:

```
set more off
```

You can include it at the beginning of the `do-file` so that when you want to run the `do-file` another time, `-more-` will be turned off.

The label list command

You can select specific variables to only look at those labels. From the menus:

1. From the **Data** menu select **Labels / Label Values**
2. Select **List value labels**
3. Select **district** and **vil**, click on  and switch to the `do-file` editor to paste the command. Switch back to the dialog box and click on **Ok**.

The listing shows you what values are assigned to a label. *Note: A label name can be assigned to multiple variables. You can create a label name for 1=yes 2=no and assign that label name to several different variables.*


In the **Command** window you can also type

```
label list district vil
```

The codebook command

To document all the variables including those that do not have value labels, another command is available:

1. From the **Data** menu select **Describe data...**

2. Select **Describe data contents (codebook)** click on  and switch to the do-file to paste the command. Switch back to the dialog box and click on **Ok**.

In the Command window you can also type

```
codebook
```

In this output every variable is listed. The type of variable is given, the range of values in the variable, number of unique values, how many cases have a missing value, and it also includes descriptive statistics for variables. The output for the descriptives is based on whether Stata thinks the variables are continuous or categorical. Stata cannot always tell if the variable is categorical, so it does not always display a frequency table for a categorical variable.

Generating descriptive statistics:

After examining the variables we will begin to examine the data by running descriptive statistics (e.g. counts, averages, maximum, minimum, and standard deviations) for all variables. This type of analysis helps you to find data entry errors. It also gives you a "feel" for what kind of data are in the file, to see that missing values have been defined correctly, etc. It may be tempting to skip this step for some data sets or for some variables, but this is an important step that will almost always save time later and improve analysis. For example, finding out the average age of all respondents may not be something you are interested in knowing, but if the average age turns out to be 91.3 years, you would be alerted that that something is probably wrong with the data.

The summarize and tabulate commands

Basic descriptive statistics can be obtained from two commands—**Summarize** and **Tabulate**. **Summarize** is used for continuous variables, while **Tabulate** is used for categorical variables.

There are three types of variables.

Continuous variable

1. A *continuous variable* is a variable that does not have a fixed number of values. It measures something, e.g. age, weight, population. Variable **ca3** (age) is a continuous variable because age can take on many different values.

Categorical variable

2. A *categorical variable* is a variable that has a limited number of values that form categories or groups to which a thing belongs, e.g. geographic



Indicator variable

location, relation to head. For example, look at the Annex Table IA: House-hold Member questionnaire. Variable **ca2** (relation to head) is a categorical variable because its values are limited to 6 categories.

3. An *indicator variable* is a special type of categorical variable. This type of variable denotes whether something is true, e.g. yes/no questions, or whether a person is male or female. This type of variable contains only 2 categories, i.e., it divides the data into 2 groups.

Start by examining the data in the file. Use the **Data Editor** window to scroll through your data file. To do this, perform the following steps:

Descriptive statistics - using one variable

1. Click on the Data Editor button  on the Tool Bar or in the **Command Window**, type **edit** and press <Enter>. You could, instead, click on the Browse button  since we only want to look at the data.
2. Scroll through the data.

A period in a field indicates a missing value or system missing value. In Stata you can specify up to 27 different missing values, e.g. .a or .b and are called "extended" missing values. Extended missing values are used to identify specific reasons why there are no data, e.g. person refused to answer, or a question was not asked.


Scrolling through the data will give you a "feel" for what is in file. It might also help point out obvious errors, e.g. a variable whose values are missing for all listed cases.

Decide which of the variables in this file are continuous and which are categorical (normally you would refer to the questionnaire to make this decision). You need to know this in order to select the right procedure to use for each variable. If you mistakenly perform a **Tabulate** on a continuous variable, you will probably get more output than you really want, with possibly hundreds of different "categories", one for each different value found. If you perform a **Summarize** on a categorical variable, you will usually get meaningless results, since the average value of a variable that consists of categories has no real significance.


By examining the data, you should have found that variable **ca3** (age) is continuous and the remaining variables are categorical. To run descriptives on **ca3**, do the following:

Descriptives

The summarize command

1. From the **Statistics** menu select **Summaries, Tables & Tests** then **Summary and Descriptive Statistics** then **Summary Statistics**
This will open the Summarize - Summary Statistics dialog box. (This command is also available from “Data”, “Describe data” “Summary Statistics”.)
2. The cursor should be in the **variables** box. There is a dropdown arrow at the end of the variables box.  Click on the drop down arrow to select the variables you want. Highlight **ca3** and click . In the **options** section below the variable box, note that “Standard Display” is the default selection for output.

Don't forget to click on the  icon to copy the command to the clipboard, switch to the do-file editor, paste the command and switch back to the dialog box.


3. Click on the  button to run the command. The dialog box will remain open.

The output appears in the Stata Results window. You will see that the mean for age (**ca3**) is 21.33602 years. The Stata command is

```
summarize ca3
```

The Results window displays:

Variable	Obs	Mean	Std. Dev.	Min	Max
ca3	1524	21.33602	17.69252	.5	81



Return to the dialog box. (On the task bar you can see “summarize-Summary...”, click on this task.) Click on the  in the lower left corner to see more detail about the summarize command. In this help window, you can see that the first two letters of the command are underlined, e.g. summarize. In Stata, only the letters that are underlined are absolutely required for the command to be recognized. The following command also works.

```
su ca3
```

To close the Viewer, click on the x in the upper right

hand corner of the dialog box.

If we wanted more statistics, switch back to the **Summarize - Summary Statistics** dialog box (you can see the icon on the task bar.)

4. Click on the radio button next to “Display additional statistics”, Click on the  to copy the command to the clipboard.
5. Click on the  button to run the command. The dialog box will close.

The results are:

age					

	Percentiles	Smallest			
1%	1	.5			
5%	1	.6			
10%	3	1	Obs		1524
25%	7	1	Sum of Wgt.		1524
50%	16		Mean		21.33602
			Std. Dev.		17.69252
		Largest			
75%	32	75	Variance		313.0252
90%	48	76	Skewness		.9152221
95%	57	78	Kurtosis		3.00135
99%	69	81			

The median age is 16 (50% - Percentile).

The Stata command is

```
summarize ca3, detail
```

Switch to the **Do-File Editor** and paste the command. Insert comments to explain the commands you have pasted.

Information returned by Stata commands

When you run a command, Stata sends the information to the **Results** window as well as saves the information in memory. To see what has been saved, you can use the **return list** command,

In the **Command** window, type

```
return list
```

The information that is returned from the **summarize** command is displayed.

```

      r(N) = 1524
    r(sum_w) = 1524
    r(mean) = 21.33602362206289
    r(Var) = 313.0251689442948
    r(sd) = 17.69251731507687
  r(skewness) = .9152220664756392
  r(kurtosis) = 3.001349748747086
    r(sum) = 32516.10000002384
    r(min) = .5
    r(max) = 81
    r(p1) = 1
    r(p5) = 1
    r(p10) = 3
    r(p25) = 7
    r(p50) = 16
    r(p75) = 32
    r(p90) = 48
    r(p95) = 57
    r(p99) = 69

```

You can use these values stored in memory to perform calculations. For example, to subtract the mean of **ca3** from **ca3** :

```
generate ca3_mean = ca3-r(mean)
```

Using the information in memory eliminates the need to type specific numbers and will give you more accurate values.

Since the variables **ca1** (work on a farm or not), **ca2** (relation to head), **ca4** (sex), **ca5** (level of schooling) and **ca6** (marital status) are categorical, we will run a Tabulate on them. To run a tabulation, do the following:

1. From the menus click on **Statistics..** then **Summaries, Tables & Tests** then **Tables** then **Multiple one-way tables**
The Tab1 - One-way Tables dialog box opens.
2. Click on the drop down area to select the variables:
ca1 ca2 ca4 ca5 ca6
3. Click on the copy button, switch to the **Do-File Editor** and paste the command then switch back to the dialog box and click on the **Ok** button.
4. The command will be executed.

You will see in the Stata **Results** window that for **ca1** 70.67% of the household members work on a farm. The results for **ca6** show that 37.99% of those surveyed are in

TABULATE - Frequencies

The `tab1` command

monogamous marriages.

The Stata command is:

```
tab1 ca1 ca2 ca4 ca5 ca6
```

Note: *to produce a tabulation (frequency) of just one variable, you can use the **tabulate** command. However, if you want to list several variables in the frequency command, you must use the **tab1** command. Below, you will see that if you use the tabulate command and list 2 variables, a cross-tabulation is produced.*

Another useful way to examine a continuous variable is to Graph the variable to view the distribution of the values. From the menus select **Graphics, Histogram**

1. Click on the drop-down area for the Variables box and select `ca3`.
2. Check the box for **Width of bins** and type in **5** in the box next to this option. The ages will be grouped into 5 year ranges.
3. For the Y-axis click on the radio button next to “Frequency” so we will see the number of cases in the age groups.
4. Click the copy icon and then click on **Ok** to run the command.

The `histogram` command

The Stata command is:

```
histogram ca3, width(5) frequency
```

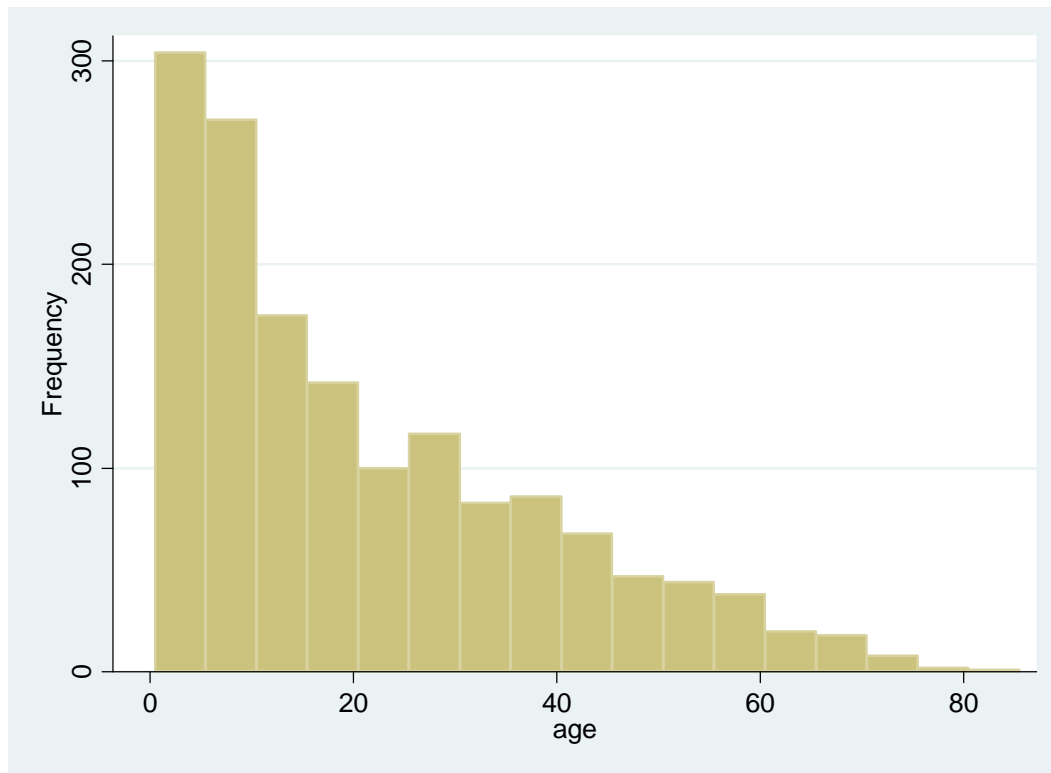
See a copy of the graph below.

Saving a graph to a file

If you want to save this graph to a word processing document, you can right click on the graph, select “copy graph”, then switch to your word processor and paste it into the document. If you want to save the graph to disk, right click and choose “save graph”.

Note: *Only one graph appears in the graph window at a time. If you run multiple graph commands at one time from a do-file, only the last graph will be visible. You must run one command, save or copy the graph, then run the next graph command, save or copy that graph.*

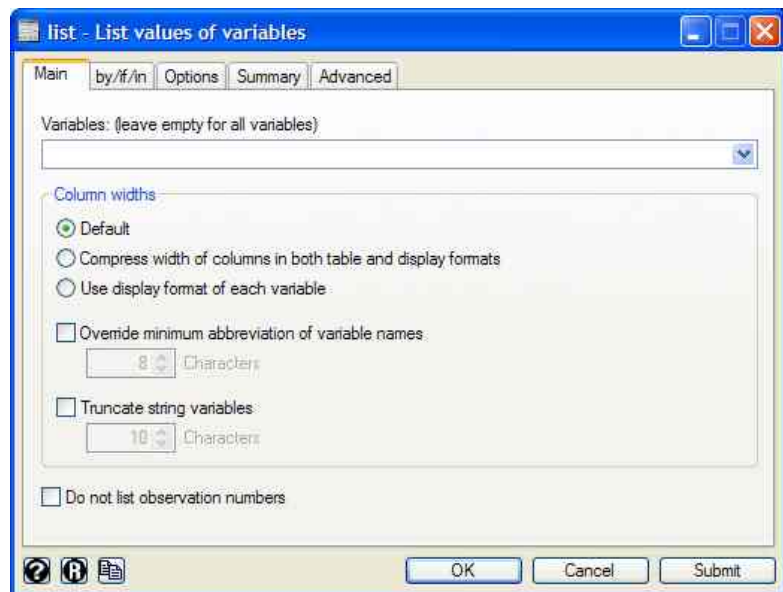
For a more detailed description of the sub-commands available for **Summarize** and **Tabulate** refer to the Guide for STATA References S-Z.



The `list` command

You may want to look at the data selecting only specific cases rather than scrolling down through the data set to find a specific case or cases. The `list` command gives you the option to select all or specific cases.

1. From the **Data** then **Describe Data** menu select **List Data**



The `list` dialog box `list - List values of variables` has 5 tabs where you can set specific parameters for the data that you

want to list.

On the **Main** tab you can specify the variables to be listed or leave it blank to list all variables. The default column width separates each variable by 5 spaces and shows the variables in “display” format. Below is an example:

1524.	district angoche	vil monari	hh 24	mem 6	ca1 no	ca2 son/daugh	ca3 5	ca4 m
	ca5 illiterat			ca6 single		univ arizona		

2. Select the variables using the drop-down arrow:
district vil hh mem ca1 ca2 ca3 ca4 ca5 ca6
Note: if you wished to include all variables, leave the box empty.
3. Click on the tab labeled “by/if/in”
In this tab we can limit the number of cases that are displayed.
4. Check the box next to “Use a range of observations”. Specify the range to be from 1 to 10.
5. Click on the “Options” tab. Under “Table options” check the box next to “**Force a clean table**”.
Note: value labels will be displayed. To see the numeric values, place a next to the box “Display numeric codes rather than labels values”.
6. Click on the “copy” button and then click on **Ok** to run the command.
7. In the **Results** window you see a list of the observations. If the information for each observation is wrapping to the next line, you can resize the **Results** windows so that it is wider. Place your mouse pointer on the right border of the window and when you see a double arrow, click the Left Mouse Button, hold it and drag the right side out to make the window wider.

If you see the **-More-** at the bottom of the Results window, there are several methods you can use to continue:


Press <Enter>

Press any key

Click on the **More** button on the tool bar

Click on the **-more-** at the bottom of the Results window

If you wish to interrupt a Stata command, you can:

click on the **Break**  button on the Tool bar or press <Ctrl-Break> or type q (the letter q for quit) in the **Command** window.

To rerun the command you just ran, click on the last command in the **Review** windows. You see the command is now in the **Command** window. Press <Enter> to run the command.

Copy the command to the **Do-file** editor and add comments to explain what you have done. The Stata command should look like

```
list district vil hh mem ca1 ca2 ca3 ca4 ca5 ca6 in
1/10, compress clean
```

If you wish to, you can type the list command in the **Command** window. If you are typing in the command window, you can pick the variables from the **Variable** window and the name will be pasted into the **Command** window.

Note that to list a subset of observations, Stata uses the key word “**in**”, e.g. in 1/10. The key word “**IN**” restricts the list to a range of observations. Examples are:

```
list in 1      lists first observation
list in -1     lists last observation
list in 2/4    lists observations 2 through 4
list in -3/-2  lists 2 observations starting with the 3rd
from the last observation.
```

To limit the listing to a specific criterion use the “**if**” key word. Examples are:

```
list district vil hh mem ca3 if ca3 > 70
list district vil hh mem ca2 ca3 if ca3 < 15 & ca2 < 3
```

If the variables you want to list are in the order in the file that you want to see, rather than list each of the names, you can type the first variable, then a dash (-), then the last variable in the list, e.g.

```
list district-ca3 if ca3 < 15 & ca2 < 3
```

If we want to see the observations with the five lowest values and five highest values, we would first sort by that variable and list the first five cases and the last five cases. For example, if the question is: What is the age of the 5

youngest head of households and what is the age of the 5 oldest head of households?

Stata commands:

```
sort ca2 ca3
list district vil hh mem ca1 ca2 ca3 in 1/5
gsort -ca2 +ca3
list district vil hh mem ca1 ca2 ca3 in -5/-1
```

Reminder: after any command built, we will copy the command into the Do-file Editor window and switch back to the menu box and run the command.

Apply what you've just learned about descriptive statistics by doing the following exercise.

Exercise 1.1

Run descriptive statistics on another sample file. Use the production questionnaire - Table IV, whose data are in file C-Q4.DTA.

Hints:

- a. make C-Q4.DTA your working data file.
- b. Use the **Summarize** command for continuous variables, and **Tabulate** or **tab1** for categorical variables.
- c. **Prod** is a categorical variable.
- d. Quantities (**p1b**, **p2b**, ...) are continuous variables.
- e. Units (**p1a**, **p2a**, ...) are categorical variables.
- f. **p4** (month in which stocks ran out last year) & **p6** (month in which stocks will run out this year) are categorical variables.

A small sampling of what you should find from running these frequencies and descriptive statistics follows:

Tabulate:					
product		Freq.	Percent	Cum.	
cotton		83	4.90	4.90	
peanuts		144	8.51	13.41	
rough rice		155	9.16	22.56	
bananas		50	2.95	25.52	
sweet potato		12	0.71	26.23	
cashew liquor		24	1.42	27.64	
sugar cane liquor		11	0.65	28.29	
dried cashew		2	0.12	28.41	
sugar cane		13	0.77	29.18	
cashew nut		130	7.68	36.86	
coconut		45	2.66	39.52	
beans		279	16.48	56.00	
manteiga beans		7	0.41	56.41	
sunflower		5	0.30	56.70	
oranges		13	0.77	57.47	
cashew fruit		44	2.60	60.07	
manioc		338	19.96	80.04	
sorghum		124	7.32	87.36	
maize		192	11.34	98.70	
"ossura"		5	0.30	99.00	
tobacco		4	0.24	99.23	
tomato		13	0.77	100.00	
Total		1,693	100.00		
Summarize:					
Variable	Obs	Mean	Std. Dev.	Min	Max
p1b	1670	26.35286	163.4359	0	5000
p2b	1598	22.81508	159.5101	.5	5000
p3b	173	2.523121	4.574581	0	30
p5b	1231	15.61243	86.10356	0	1460
p7b	869	4.938435	6.875536	0	100

Descriptive Statistics - using two or more variables

Two-way Tables with Categorical Variables (Cross- tabulation)

The `tabulate` command

We wish to produce a table that shows the distribution of cases according to their values using two or more categorical variables.

Look at the household member questionnaire in the annex section, Annex Table IA. One thing you might be interested to know is how the gender of the respondents varied by their relationship to the head of household. This would tell you, for example, how many females are heads of households. The **Tabulate** command will produce this type of summary. Make the household member file, `c-q1a.dta`, the working data file.

1. Click on the yellow open folder tool at the top left of the Toolbar
2. Select the file `c-q1a.dta`.

3. Click on **Open** to open the file.
4. Copy the command for opening the file which appears in the **Results** window, into the **Do-file Editor** window.

Reminder: You should add comments to your do-file so that you can remember what and why you were doing specific commands when you developed the do-file. Several days or weeks from now you may not remember. Comments in a do-file start with slash and then an asterisk and end with an asterisk and a slash:

```
/* this is a comment */
```

Stata will not run a line as a command if it begins with these symbols.

To create a two-way table do the following:

1. From the menus click on **Statistics..**
Summaries, Tables & Tests
Tables
Two-way tables with measures of association
The Tabulate2 - two-way tables dialog box opens.
2. In the **Row Variable** box choose **ca2** from the drop-down choices.
3. In the **Column Variable** box choose **ca4** from the drop-down choices.

We would like to see row percentages and column percentages.

4. Under **Cell Contents** click in the box next to **Within column relative frequencies** to put a ✓.
5. Click in the box ✓ next to **Within row relative frequencies**.
6. Click on copy button, switch to the **Do-File Editor** and page the command. Write a comment and then switch back to the dialog box to click on the **Ok** button. The command will be executed.

The Stata command is:

```
tabulate ca2 ca4, column row
```

The **Key** box in the **Review** window specifies which statistics appears on each row in the cells.

Key			
	frequency		
	row percentage		
	column percentage		
relation to head	sex		Total
	m	f	
head	321	21	342
	93.86	6.14	100.00
	41.42	2.88	22.74
wife/husband	2	306	308
	0.65	99.35	100.00
	0.26	41.98	20.48
son/daughter	374	336	710
	52.68	47.32	100.00
	48.26	46.09	47.21
mother/father	1	5	6
	16.67	83.33	100.00
	0.13	0.69	0.40
other relative	77	61	138
	55.80	44.20	100.00
	9.94	8.37	9.18
Total	775	729	1,504
	51.53	48.47	100.00
	100.00	100.00	100.00

In this case we wanted counts, row percentages, and column percentages. Row percentages sum to 100 across all the cells in a row, while column percentages sum to 100 down the cells in a column. The table produced by this command tells you that there are 21 female heads of households, and that 6.14% of the total heads of households are female (row percent). Of those who are female, 2.88% are head of household (column percent).

Summary statistics on a continuous variable for each value in a categorical variable

The `by ... sort: summarize` command

For this analysis the same command is used as for general summary statistics with a slight modification. This command will show how the mean and other statistics for a continuous variable differ by the values of one or more categorical variables.

Suppose we want to know how the age of the member varied by his/her relationship to the head of household. If we did this with **Tabulate** we would get a table with dozens of cells for the different ages represented, which

would be an unusable format. Instead we will use **Summarize** using the “by” key word.

1. From the **Statistics** menu select
Summaries, Tables & Tests
Summary Statistics
Summary Statistics
The Summarize - Summary Statistics dialog box opens.
2. Select **ca3** from the drop-down box for Variables
3. Under “Options” in this tab, select **Standard Display**.
4. Click on the “**by/if/in**” tab.
5. Click in the box “**Repeat command for groups defined by**”
6. In the box below this option, select **ca2**
7. Click on the copy button, switch to the do-file editor and past and switch back to click the **Ok** button. The command will be executed.

This command calculates the means of the variable **ca3** (age) separately for each different value of the variable **ca2** (relation to head) including the system missing value.

The Stata command is:

```
by ca2 sort: summarize ca3
```

Note that the command begins with “**by**”. This command is first sorting the data by **ca2** before it runs the summarize command. You could also sort the file by **ca2** first and then just use the “**by**” key word, e.g.

```
sort ca2
by ca2: summarize ca3
```

From this output you find that the average age of heads of households is 41.5277 years while the average age of their spouses is 33.1871 years. Four observations have no value for **ca2**.

```

. by ca2, sort : summarize ca3
-----
-> ca2 = head
  Variable |      Obs      Mean   Std. Dev.   Min   Max
-----+-----
      ca3 |      343   41.5277   14.12719    18   81
-----
-> ca2 = wife/husb
  Variable |      Obs      Mean   Std. Dev.   Min   Max
-----+-----
      ca3 |      310   33.1871   11.80466    13   76
-----
-> ca2 = son/daugh
  Variable |      Obs      Mean   Std. Dev.   Min   Max
-----+-----
      ca3 |      718    8.133844    5.797507     .5   48
-----
-> ca2 = mother/fa
  Variable |      Obs      Mean   Std. Dev.   Min   Max
-----+-----
      ca3 |         6   48.16667   22.09449    20   69
-----
-> ca2 = other rel
  Variable |      Obs      Mean   Std. Dev.   Min   Max
-----+-----
      ca3 |      143   12.55245   10.06785     1   75
-----
-> ca2 = .
  Variable |      Obs      Mean   Std. Dev.   Min   Max
-----+-----
      ca3 |         4     15    12.24745     6   33

```

Data Transformations

After examining the results of the descriptive statistics you will often want to do data transformations. A data transformation is an operation that takes an existing variable and either changes the values in a systematic way or uses the values to calculate a new variable. The following example shows a common data transformation: the conversion of a continuous variable to a categorical variable.

The information we received from the **summarize** command is interesting, but it might also be useful to see the actual distribution of the ages into groups or categories, so we can tell, for example, how many heads of household are older than 60. Since the age variable, **ca3**, is continuous, we cannot do this directly—first we have to transform it. Let's suppose we're interested in four categories: 0-10 years old, 11-19 years, 20-60 years, and over 60 years of age.

Converting continuous variables to categorical variables

The **generate** command
 The **replace** command
 The **label variable** command
 The **label define** command

First method:

The **generate** command

To categorize a variable, we can use the **generate** command. Categorizing a continuous variable makes detailed information more general. To keep the detailed information as well as the new general information, you must recode the variable into a new variable. If you recode into the same variable the original values will be lost.

There are several methods that can be used to recode a continuous variable.

First method: If you wish to see the category values of 1, 2, 3, and 4 where

1 = 0-10,
 2 = 11-19,
 3 = 20-60 and
 4 = over 60

you can do the following:

1. From the **Data** menu select **Create or change variables / Create new variable**
The Generate - Create a new variable dialog box opens.
2. Under the **Main** tab, type the name of the new variable in the **New Variable name** box:
age_gp
3. For the **Contents of new variable** box, type in
1
This is the value that you want the new variable to have.
4. In the drop down box for on the **New variable type** select **byte**.
5. Click on the **If/In** tab.
6. In the **If: (expression)** box, type in
ca3 >=0 & ca3 <=10
Note: you must use the ampersand symbol (&), not the word "and".
7. Click on the copy button, switch to the **do-file editor**, paste and switch back to the dialog box and click on **Ok**

The Stata command is:

```
generate byte age_gp= 1 if ca3 >=0 & ca3 <=10
```

Stata will indicate in the **Results** window how many missing cases were generated:

(949 missing values generated)

The replace command

That means that from the total of 1524 cases, 949 were not assigned a value. Now that the new variable has been created, another command is used to assign the codes for the cases with no values for this new variable. The command is the **Replace** command.

8. From the **Data** menu select **Create or change variables / Change contents of variable**.
The Replace-Replace contents of variables dialog box opens.
9. In the **Variables** box select the name of the variable that was just created: **age_gp**
10. Type **2** in the **New Contents** box
11. Click on the **If/In** tab.
12. In the **If: (expression)** box, type in
ca3 >10 & ca3 <=19
13. Click on the copy button, switch and paste in the do-file editor, switch back and click on **Submit**. The dialog box remains open and the command is run.
The Results window indicates how many changes were made: (271 real changes made).
14. Now make the changes to assign values to the other categories: In the **If: (expression)** box, change the criteria to: **ca3 >19 & ca3 <=60**
15. Click on the **Main** tab and type **3** in the **New Contents** box
16. Click on the copy button, switch and paste in the do-file editor, switch back and click on **Submit**. The dialog box remains open and the command is run.
17. Type **4** in the **New Contents** box
18. Click on the **If/In** tab.
19. In the **If: (expression)** box, change the criteria to: **ca3 >60**
20. Click on the copy button, switch and paste in the do-file editor, switch back and click on **Ok**

The Stata commands created and run are:

```
generate byte age_gp= 1 if ca3 >=0 & ca3 <=10
(949 missing values generated)
```

```
replace age_gp = 2 if ca3>10 & ca3 <=19
(271 real changes made)
```

```
replace age_gp = 3 if ca3>19 & ca3 <=60
(629 real changes made)
```

```
replace age_gp = 4 if ca3>60
(49 real changes made)
```

Note that the **Results** window also shows how many observations were modified after each command was run.

The next step is to verify that the changes were made correctly. Run the **Tabulate** command on the new variable.

1. From the menus click on **Statistics..**
Summaries, Tables & Tests
Tables
One-way tables
The Tabulate1 - One-way Tables dialog box opens.
2. For the **Categorical Variables** box, select the variable **age_gp** from the drop-down box.
3. Click on the copy button, switch and paste in the do-file editor, switch back and click on the **Ok** button.

The Stata command is:

```
tabulate age_gp
```

There should be 4 codes in the frequency table—1, 2, 3, and 4. We can use the **Data Browser** to check to see what changes were made. Click on the **Data Browser** button. Close the window when you are finished.

The values do not have any labels to define what the values of 1, 2, 3, and 4 mean. We want to add both a variable label and give labels to the values in this variable.

To assign a variable label:

1. Click on **Data**, then **Labels**, then **Label variable**.

The label variable command

2. In the **Variable:** box, select the name of the variable: **age_gp**
3. In the **New variable label box**, type
Age Group
Note: Label may be up to 80 characters.
4. Click on the copy button, switch and paste in the do-file editor, switch back and click on the **Ok** button.

The Stata command is:

```
label variable age_gp "Age group"
```

To assign value labels to a variable we first have to define a label and assign value labels to the values in that label:

The label define command

1. Click on **Data**, then **Labels** then **Label values** then **Define or modify value labels**.
Remember, Stata assigns a name to a group of value labels.
2. In the **Define value labels** dialog box, click on the button **Define**
3. In the **Define new label** box, type **age_gp** in the **Label Name** box and click on the **Ok** button.
4. In the next dialog box, **Add value**, type 1 in the **Value** box and in the **Text** box type **0 to 10**. Click on the **Ok** button. The dialog box remains open.
5. Continue defining the labels for the values:
Type 2 in the **Value** box and in the **Text** box type 11 to 19, and click on the **Ok** button.
Type 3 in the **Value** box and in the **Text** box type 20 to 60, and click on the **Ok** button.
Type 4 in the **Value** box and in the **Text** box type 61 and older, and click on the **Ok** button.
6. All the values have been assigned a label. To close the dialog box, click on the **Cancel** button to close the **Add value** dialog box.
7. Click on the **Close** button to close the **Define value labels** dialog box.

As you can see in the **Results** window, the Stata commands are:

```
label define age_gp 1 "0 to 10"  
label define age_gp 2 "11 to 19", add  
label define age_gp 3 "20 to 60", add  
label define age_gp 4 "61 and older", add
```

The label values commands

Copy these commands to your do-file. The first command creates a label name and defines the label for the first value. The next 3 commands add to the label name and define the labels for the next 3 values.

Now that the label has been defined, we can assign this label to the variable we created with the 4 categories.

8. Click on **Data**, then
Labels then
Label values then
Assign value label to a variable.
The Label values – Assign value label to a variable dialog box opens. Note that age_gp already appears in the Value label box.
9. In the **Variable:** box select **age_gp**.
This is the variable that we want to attach a label to.
10. The **Value label** box, already has the label “**age_gp**” selected.
11. Click on the copy button, switch to the do-file editor, paste the command, switch back and click on the **Ok** button.

The Stata command is:

```
label values age_gp age_gp
```

Second method:

Another method we can use, which is much easier, is to generate the new variable, assign the new values and assign the labels for the values in one step:

1. Select **Create or change variables** from the **Data** menu
2. Select **Other variable transformation commands**
3. Select **Recode categorical variable**
4. In the “**Main**” tab, select **ca3** in the Variables box.
5. In the **Required** box, specify the range you want and the new value to be assigned as well as the label for that new value, e.g.
(0/10 = 1 “0 to 10”)
6. In the Optional boxes continue to specify the ranges and value to be assigned, e.g.
(10.001/19 = 2 “11 to 19”)
(19.001/60 = 3 “20 to 60”)
(60.001/max = 4 “61 and older”)

*Note: examples on how to specify the value can be see if you click on the “**Examples**” button.*

7. Click on the “**Options**” tab. Click on the radio button next to “**Generate new variables**”.
8. In the box, type the name of the new variable:
age_gp1
9. We can also specify a name for the value labels. Click in the box next to “**Specify a name for the value label defined by the transformation rules**”.
10. In the box, type “**age_label**”.
11. Click on **Ok**.

The Stata command is:

```

recode ca3 (0/10 = 1 "0 to 10")      ///
(10.001/19 = 2 "11 to 19")         ///
(19.001/60 = 3 "20 to 60")         ///
(60.001/max = 4 "61 and older"),   ///
generate(age_gp1) label(age_label)

```

Note, to continue a command on another line, the end of the line should have `///` to tell Stata that the command continues to the next line.

Let’s add a variable label to the new variable: The Stata command is:


```
label variable age_gp1 "Age group - second method"
```

Now, compare the **age_gp** variable with the **age_gp1** variable. Use a cross tabulation (`tabulate2` command). The counts should be identical.

Variation on the second method

The recode function

The same results can be achieved by using one command the `recode()` function in conjunction with the `Generate` command. The `recode()` function takes three or more arguments. The first argument is the variable name that you want to categorize. The rest of the arguments are used to determine how to code the new variable.

1. Select **Create or change variables** from the **Data** menu
2. Select **Create new variable**
3. Click on the reset button in the lower left hand corner of the dialog box -  if you need to remove any information that appears in the box.
4. Under the **Main** tab, type the name of the new variable in the **Generate Variable** box:
agecat

5. Click on the **Generate variable as type** drop down box and change to **byte**.
6. For the **Contents** box, click on the **Create** button.
7. In the **Expression builder** box, under the **Category** section, select **Programming**
8. A list of available functions is displayed. Scroll down to the **recode()** function and highlight that function. You will see a description of the function at the bottom of the dialog box.
9. Double click on this function. The function will be pasted in the window at the top of the dialog box so that you see:

```
recode(x,x1,x2,...,xn)
```

The first “x” is highlighted. Replace the first “x” with the variable name, **ca3**, so that the expression now looks like:

```
recode(ca3,x1,x2,...,xn)
```

Replace the “x1” with the value of the highest age that you want to recoded for the first group, e.g.

```
recode(ca3,10,x2,...,xn)
```

Continue replacing the values with the next group to be coded until all groups are defined, e.g..

```
recode(ca3,10,19,60,100)
```

Stata will use the value as the code assigned to all cases that fall within that group. The value of 10 will be assigned to all observations with ages between 0 and 10, the value of 19 will be assigned to all observations that fall between ca3 >10 and <=19, and so on.

10. Click on **Ok** to exit the expression builder dialog box.
11. Click on the copy button, switch to the **do-file editor**, paste the command and switch back. Click on **Ok** to run the command.

The Stata command is:

```
generate byte agecat: recode(ca3,10,19, 60,100)
```

Run a **tabulate** on the new variable - **agecat** - and compare the number of cases in each category between the new variable and the **age_gp** variable.

These new variables are not yet part of the data file stored

on disk. We must save the data file for these variables to be included permanently in the data file. It is a good practice to save the file under a different name in case we want to go back to a previous version of a file. For this reason we will use the **Save As** command from the **File** menu. The new file name will be **Q1A-AGE.DTA**.

1. From the **File** menu select **Save As...**

The cursor should be in the box under File name: above the Save as type: Stata data (.DTA) drop-down box. Since *.dta in the File name: area is blocked, you can immediately start typing the new file name.*

2. Type **q1a-age** (The .DTA extension will be added automatically.)
3. Click on **Save** to run the command.

The Stata command is:

```
save "q1a-age.dta"
```

Copy this command from the **Results** window to the do-file editor. We do not want to include the specific directory so delete the part of the command that references the specific directory. If we want to share the do-file with another colleague that person will only have to change the initial “cd” (change directory) command at the beginning of the do-file to be able to run the do-file.

Now each time the data file Q1A-AGE.DTA is opened, the **age_gp** variable as well as the other two **age_gp1** and **agecat**, will be included.

You might want to analyze this new categorical variable using the **tabulate** command to determine how many people in each age group are heads of households, spouses, or children.

1. From the menus click on

Statistics..

Summaries, Tables & Tests

Tables

Two-way tables with measures of association

The Tabulate2 - two-way tables dialog box opens.

2. Use **age_gp** for Row variable and **ca2** (relation to head) for Column variable.
3. Check the proper selections in the Cell content choices, for we want both Row and Column percentages.
4. Click on the copy button, switch to the do-file editor, paste the command and switch back. Click on **Ok** to run the command.

The Stata command is:

```
tabulate age_gp ca2, column row
```

From the table you can see that 11.95% of heads of households are 61 years of age or older. Also, of the people 61 years or older, 83.67% are heads of households.

Compare the information you get from this **Two-way table** analysis with the information from the **Generate** command performed on **ca3** (age) earlier.

Apply what you have learned about data transformations and descriptive statistics in the following exercise.

Exercise 1.2

Using the Household Data and Questionnaire (available in the annex), find out the number of households in each district that have 1-4, 5-7, and more than 7 persons per household. One way to find out this information is to create the following table.

Hints:

- a. Use the file **c-hh.dta**.
- b. Recode **h1** into **hsize** using the following groups: (1 thru 4) (5 thru 7) (8 thru Highest).
- c. Add a variable label and value labels.
- d. Run a **two-way table - Tabulate** on this variable by **district**

Looking at the results, you can see 34.76% of all 1 to 4 member households are found within Monapo and that 60.75% of all households in Monapo have 1 to 4 members in a household.

Key				
	frequency			
	row percentage			
	column percentage			
Household size	monapo	district ribaue	angoche	Total
1-4 members	65 34.76 60.75	48 25.67 40.34	74 39.57 64.35	187 100.00 54.84
5-7 members	39 29.77 36.45	56 42.75 47.06	36 27.48 31.30	131 100.00 38.42
8-12 members	3 13.04 2.80	15 65.22 12.61	5 21.74 4.35	23 100.00 6.74
Total	107 31.38 100.00	119 34.90 100.00	115 33.72 100.00	341 100.00 100.00

We have completed Section 1. Before we close down the session, we need to close the log file that has been recording the commands and output. The command to close the log file is

`log close`

We can type this command in the Command window and run it and then copy and paste the command in the do-file.

Before exiting Stata save the do-file. The file contains all of the commands. It is useful to keep this file so you can rerun the commands if you want review the commands and the output that is produced. If you have not yet saved the file follow these instructions. Otherwise, click on the Save icon on the tool bar.

1. Make the **Do-file Editor** the active window using its icon on the Windows taskbar.
2. From the **File** menu select **Save As...**
3. Enter the filename **session1**
The .do extension will be added to the name automatically.
4. Click on **Save**.

To exit Stata, switch back to the Stata window:

1. From the **File** menu select **Exit**

A dialog box will open to say that "Data have been changed without being saved. Do you really want to exit?"

2. Click on **Yes**

We do not need to save the newly created categorical variable. We will not be using it again.

If you want to look at the log file that we just created, open Stata.

1. From the **File** menu select **View**

A dialog box will open "Choose File to View"

2. Click on the browse button. You will see listed a file called "session1.smcl".

3. Select that file, click on **Open** , then click on **Ok**

The **Viewer** opens and displays the log file that has saved all the commands and output from Section 1 of the tutorial.

To close the **Viewer**, click on the **x** in the upper right hand corner of the **Viewer**.

STATA 10 - SAMPLE SESSION

SECTION 2 - Restructuring Data Files - Table Lookup & Aggregation

Restructuring Data Files

For some types of analysis the data files may need to be restructured to a different level. The data from the four sections of the questionnaires—household, member, production and sales—are in four separate data files because the data are at different levels. The household data is at the most general, or highest, level - one case per household. The other three files contain more detailed data, which is usually thought of as being at a lower level - there are multiple cases per household. If you are not familiar with the concept of levels of data, read "Computer Analysis of Survey Data -- File Organization for Multi-Level Data" by Chris Wolf, before continuing on with this section. This paper is available at <http://www.aec.msu.edu/fs2/survey/index.htm>

The analysis we did in Section 1 was done at each level separately, using just the variables in a single file at a time. However, other types of analysis require combining data from more than one file. Let's look at an example.

Suppose we want to create a table of calories per adult equivalent produced per day from the principal food crops. Furthermore, we want to see how this varies by district and calorie-production quartile.

TABLE:1 Food Production in calories per adult equivalent per day

Districts	Calorie Production Quartile			
	1	2	3	4
Monapo				
Ribaue				
Angoche				

The data in their current form cannot produce this table. Many transformations are required to restructure the data to be able to provide the results for this table. The above table is an example of the complications you will encounter in real-world data analysis. This entire section will be devoted toward the goal of creating this table.

To begin, let's look at the files we have and at the variables we need to use from each of these:

- c-q1a.dta: This file contains data on household roster characteristics. It is at the household-member level. We need to use the variables **ca3** (age) and **ca4** (sex) in this exercise to compute the number of adult equivalents per household.
- c-q4.dta: This file contains data on crops produced by the household. The variables we need to calculate the total production of the household are:
 - a. **prod** - contains the codes for the agricultural crop produced.
 - b. **p1a** - contains the codes for the unit in which the production was measured (100 kg sack, 50 kg sack, etc).
 - c. **p1b** - contains the number of units produced for the year.

Note that the unit of production is not a standard unit for each crop. For example, a "100 kg sack", as the term is used in Mozambique, weighs 100 kg only when the sack is filled with maize. When it is filled with manioc root, it weighs much less than 100 kg. Thus, we need *conversion factors* to be able to convert each of the units in which production was actually measured to our standard unit, which is the kilogram.

- conver.dta: This is a *table-lookup file*. This file was created specifically to handle the problem of converting non-standard units to a standard unit. For each product-unit combination there is a conversion factor to convert the measurement to equal the weight in kilograms. In other words, there is a different conversion factor for each product-unit combination. For example, the conversion factor for a 50 kg sack of rough rice is 39.44; for a 50 kg sack of cotton it is 17.5, while a 50 kg sack of peanuts is 41.67. The variables in this file are:
 - a. **prod** - product (crop) code
 - b. **unit** - unit of measure
 - c. **conver** - conversion factor (equal to the number of actual kilograms for the combination of **prod** and **unit**)

Below, a sample of data from CONVER.DTA shows that:

rice (**prod**=7) measured in a 20 liter can (**unit**=8) weighs 19 kg;
 rice (**prod**=7) measured in a 50 kg bag (**unit**=24) weighs 53 kg;
 beans (**prod**=30) measured in a 20 liter can weighs 17 kg;

beans (**prod**=30) measured in a 50 kg bag weighs 47 kg.

prod	unit	conver
(Product)	(unit)	(conversion factor)
...
7	8	19
7	24	53
...
30	8	17
30	24	47
...

- **calories.dta**: This also is a *table-lookup file*, created for convert kilograms of food into calories of food. It contains two variables:
 - a. **prod** - the product (crop)
 - b. **calories** - number of calories per kilogram of each of the crops

To create a data files that will produce the output table described above, we need to combine the data from different files. There are different methods that can be used to combine files, depending on what is desired. In Stata, we can

1. **Append** datasets. Appending data sets means that the data in different files have the same variables and the desire is to add one data set of observations to the end of another data set (or append one file to the end of another file). An example would be that you entered data for harvest in one file for one district and entered data for harvest for another district into another file. We want the data to be in the same file. To do that, we would use the append command.
2. **Merge** datasets. Merge combines datasets horizontally matching corresponding observations. An example is a survey asking questions about the household in Part 1 and another set of questions about the household in Part 3. Each part of the survey is entered into a different data file. To combine Part 1 and Part 3 (where both sets of data are at the household level), we would use the merge command.
3. **Joinby** datasets. This type of merge combines datasets horizontally matching all pairwise combinations possible. An example is a set of data on parents and a set of data on children. Joinby would match the parents to every

observation of the children within that family. The key word “unmatched” is used and within parentheses the type of join is specified). There are four types of joins:

none - all unmatched observations are ignored (this is the default), i.e. if there is not a matching observation in both files, the observation is dropped from the final dataset.

both - unmatched observations from the “master” (or file that is in memory) and “using” (file that is not in memory) data are included.

master - unmatched observations from the “master” data are included but not unmatched observations from the “using” file.

using - unmatched observations from the “using” data are included but not unmatched observations from the “master” file.

4. **Cross** datasets. In this type of merge, the first observation in the first file is joined horizontally with every observation in the second data set. The second observation in the first file is then joined with every observation in the second data set and so on. This type of file combination is rarely used.

In this tutorial we will use the “**merge**” and the “**joinby**” commands.

With this information in hand, we can now think about the specific steps we must take to create the file we need to produce the output we want. Logically, there are three steps:

1. We need to know how many calories each household produced for the year. We can generate a file with this information using data we have stored in three files—the production file, `c-q4.dta`, and two table-lookup files, `conver.dta` and `calories.dta`.
2. We need to know how many adult equivalents are in each household. We can generate a file with this information using data from the member file, `c-q1a.dta`.

Step 1: Generate a household level file containing the number of calories produced per household.

3. We need to combine the results from steps 1 and 2 into one file so we can compute calories produced per adult equivalent per day.

In executing this step, we must keep three things firmly in mind.

First, all production is currently measured in non-standard units. Each unit can have a different weight for each of the products. Thus, we must first convert all production into kilograms.

Second, we want to know many calories are produced by each household, not kilograms. Thus, after converting all production to kilograms, we must convert kilograms to calories.

Third, an examination of the file shows that we have data for each product produced by the household. But we want to know the total calories produced by the household for specific food products, not the total calories from each separate product. After we convert all production to calories, we must sum the calories within each household to arrive at the household total.

Let's begin by creating a new do-file. Open the **Do-File Editor**. Start by including comments about the purpose of the do-file, your name as the creator of the do-file and the date. Other items to include are the Stata version, the set memory command if you have not changed the startup memory, the “cd” command to switch to the directory where you want to work, the log command to record the session. Example:

```
version 10
set memory 30m
cd "C:\Documents and Settings\aec_user\My Documents\data"
log using session2, append
```

We are now ready to open c-q4.dta, the production file.

1. Select **File / Open...**
2. Select the file name **c-q4.dta**
3. Click on **Open** to run the command.
4. Copy the command to open this file from the **Results** window, switch to the **Do-File Editor** (<Ctrl+8> or click on the button on the task bar) and paste the command into the do-file. Delete the reference to the directory.

5. Save the do-file to the name `session2.do`

We must convert all production of the crops into kilograms. To find the conversion factor appropriate for each case in the production file (`c-q4.dta`), we need to look up the product and unit in the `conver.dta` file. We will merge the information from this file into the file in memory (the production file). The variable with the conversion factor will then be available to calculate the total kgs produced. In Stata we want to use the “joinby” command for this merge. It can be found through the menus with the following choice:

Data**Combine datasets****Form all pairwise combinations within groups.**

The input files for a merge must be sorted by the *key variable(s)* (*key variables are those variables you are using to match by between the two files*). Since there is a unique conversion factor for each product-unit combination, both our product variable and our unit variable are the key variables. The `CONVER.DTA` file is already sorted by **prod** and **unit**. We must sort the current working file that is in memory the same way, while taking account of the fact that the unit variable is named **p1a** and not **unit**. To sort the cases:

1. From the **Data** menu select
Sort
Ascending data
The Sort - Sort data dialog box will open.
2. In the **Variables:** box select **prod** and **p1a**
3. Click on the “copy” icon and then click on **Ok**.
4. Switch to the do-file editor and paste the command.

The Stata command is:

```
sort prod p1a
```

Let’s look at the two variables using the `tab1` command. We can type in the Command window

```
tab1 prod p1a
```

There are 1,693 cases. We have many products. For the tabulation of `p1a` we see 2 values that have no labels (0 and 1) and note that there are only 1670 cases that contain a value for `p1a`. There are possible data problems. We would expect to see a value in `p1a` for every crop that was harvested. How would you determine if there are missing data in the `p1a` variable? If it were possible, corrections

Rename any key variables in both files to the same name

should be made before proceeding further.

We cannot merge the two files unless the variables that we want to merge by have the same names. We will rename p1a to unit.

1. From the **Data** menu select **Variable utilities** then **Rename variable**
The rename - Rename variables dialog box will come up.
2. In the **Existing Variables** box select **p1a**. In the **New variable name** box type **unit**.
3. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on **Ok**.

The Stata command is:

```
rename p1a unit
```

The **joinby** command

The files are now ready to be merged. We are doing a “File - Table” merge where the second file is our “Lookup Table”. We want to keep all records in the “master” file (or the file in memory) and keep only those records in the “using” file that match.

1. From the **Data** menu select **Combine datasets**, then select **Form all pairwise combinations within groups**
The joinby - form all pairwise combinations within groups dialog box will open.
2. To fill in the box labeled **Filename of dataset on disk**: click on the **Browse** button, Select the filename **conver.dta** and click on **Open**.
3. In the box labeled **Join observations by groups formed from specific variables**, select **prod unit**
4. Click on the “Options” tab.
5. Under “Unmatched Observations”, select **Include from data in memory**
This option will keep cases in the “master” data set (in memory) that do not have a match in the lookup data set.
6. Click on the copy button, switch to the **do-file editor**, paste the command, delete the directory reference, switch back to the dialog box and click on **Ok**

The Stata command is

```
joinby prod unit using "conver.dta",  
unmatched(master) _merge(_merge)
```

The above command tells Stata to merge the working data file or “master” (the file in memory) with the `conver.dta` file or “using” data file, (using `conver.dta` as a table lookup) to add the **conver** variable to our working data file. We had renamed **p1a** to **unit**.

Key variables are required in any procedure to merge two files when one of the files is being used as a keyed table. Our key variables specify how to merge the lookup file using `product` and `unit` (the grouping variables), because we have a different conversion factor for each product-unit combination. If we had used only **prod**, Stata would expect each product to have only a single conversion factor, with the same value regardless of the unit of measurement used. For example, it would expect the same conversion factor for rice whether it was in a 100 kg bag or a 20 liter can. This would be incorrect.

Check the resulting data file

The new working file produced by the join contains the needed conversion factor variable, **conver**. For every product-unit combination, **conver** is equal to the number of kilograms in that unit. It is always important to verify if the join was successfully completed. Click on the **Data Browser** button to look at some cases to verify that the conversion factors match the products. We could also use the `list` command to see if a 20-liter can filled with maize grain has a conversion value of 18 kilograms (`prod = 47 unit = 8`).

The Stata command is:

```
list prod unit conver if prod==47 & unit ==8
```

Note: *Two equal signs (==) are required.*

The two equal signs distinguish relational equality from the `=exp` assignment phrase. For example, if you want to create a variable where you will be assigning values to that variable, you will use an expression (`exp`) and need only 1 equal sign (example: `gen newvar = oldvar*2.5`). In the above example, `prod` already has values and we want to see only records where **prod** has the value of 47. Therefore, it is a relational equality and we must use 2 equal signs (e.g. show me only records where `prod ==47` and `unit == 8`).

We should also run a `tabulate` on the `_merge` variable as well to look at how the merge was done:

```
tab1 _merge
```

From the output you should see there are the same number of records in the file as there was before the merge, i.e. 1,693. Note that there are 27 cases where there was not a match for the prod-unit combination in the look-up file. How would you specify the “list” command to look at these 27 cases?

You would want to investigate further to see if the records without a look-up value are crops that you want to have included in the analysis you are doing, and, if they are, correct the lookup file and/or the production file and run your procedure again.

Compute total kilograms produced

The generate command

We can now calculate total kilograms by multiplying the number of units (**p1b**) by the conversion factor.

1. Select **Create or change variables** from the **Data** menu
2. Select **Create new variable**
The Generate - Create a new variable dialog box opens.
3. Under the **Main** tab, type the name of the new variable in the **New Variable name** box:
qprod_tt
4. For the **Contents of new variable** box, type in **p1b * conver**
5. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on **Ok**
6. Add a comment in the do-file to explain what you have done.

The Stata command is:

```
generate float qprod_tt= p1b * conver
```

Note that there were 49 cases where no value was generated. We had 27 cases with no conversion value. Why do the rest of the cases (22 cases) not have values?

Now that the kilograms have been calculated we need to look up the value of a kilogram in calories for each product. This information is in the table lookup file called **calories.dta**. This file has two variables — product and number of calories per kilogram. The key variable is product (**prod**). In order to add the calorie-conversion variable to the working data file we need to do another merge with keyed table lookup (**joinby**). This

The **drop** command

time the key variable only needs to be the product variable. The data file has already been sorted by product (see the previous merge), so we don't need to sort it again. Stata will reuse the `_merge` variable again with the next join we do, so we should drop this variable first since we no longer need it. The command to delete a variable is called **drop**

The Stata command is:

```
drop _merge
```

Now we are ready for the next join:

1. From the **Data** menu select **Combine datasets**, then select **Form all pairwise combinations within groups**
The joinby - form all pairwise combinations within groups dialog box opens.
2. To fill in the box labeled **Filename of dataset on disk**: click on the **Browse** button, Select the filename `calories.dta` and click on **Open**.
3. In the box labeled **Join observations by groups formed from specific variables**, select **prod** only
4. Click on the “Options” tab.
5. Under “Unmatched Observations”, select **Include from data in memory**
This option will keep cases in the original data set that do not have a match in the lookup data set.
6. Click on the copy button, switch to the **do-file editor**, paste the command, delete the directory reference, switch back to the dialog box and click on **Ok**
7. Add comments to the do-file.

The Stata command is:

```
joinby prod using "calories.dta", unmatched ( master )
_merge(_merge)
```

The new working data file produced by the merge now contains the needed calorie variable, **calories**, but check to make sure. Maize grain (PROD=47) should have 3590 calories per kilogram in the **calories** variable. We can browse the data and/or we can use the list command again.

The Stata command is:

```
list prod calories if prod==47
```

Calculate the total calories produced

Also check the `_merge` variable to see how the merge was done:

```
tab1 _merge
```

Note that there are 87 cases with no value in the **calorie** variable. How would you check to see which products have no calorie value?

We can now compute total calories produced.

1. Select **Create or change variables** from the **Data** menu
2. Select **Create new variable**
The Generate – Create a new variable dialog box opens. We have used this dialog box earlier. To clear the contents, click on the "Reset" icon in the lower left corner of the dialog box.
3. Under the **Main** tab, type the name of the new variable in the **New Variable name** box:
cprod_tt
4. For the **Contents of new variable** box, type in **qprod_tt * calories**
5. Click on the copy button, switch to the do-file editor, paste the command, switch back to the dialog box and click on **Ok**
6. Add a comment in the do-file to explain what you have done.

The Stata command is:

```
generate float cprod_tt= qprod_tt * calories
```

Note that missing values were generated for 131 cases.

Assign variable labels

The two new variables do not yet have variable labels. To assign a variable label:

1. Click on **Data**, then **Labels**, then **Label variable**.
2. In the **Variable:** box, select the name of the first variable: **qprod_tt**
3. In the **New variable label** (may be up to 80 characters) box, type
Total production in kgs
4. Click on the copy button, switch to the do-file editor, paste the command, switch back to the dialog box and click on the **Submit** button.
Clicking on the "submit" button leaves the dialog box open so we can then define the

label for the `cprod_tt` variable without having to select it again from the menus.

5. In the **Variable:** box, select the name of the second variable: **cprod_tt**
6. In the **New variable label** (map be up to 80 characters) box, type
Total calories produced
7. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on the **Ok** button.
8. Add a comment to the do-file to explain what you have done.

The Stata commands are:

```
label variable qprod_tt "Total production in kgs"
label variable cprod_tt "Total calories produced"
```

This gives us a new working data file with total calories produced per product for each household.

Select only staple food products

The final output table asks only for information about the staple food crops. These are defined as:

peanuts	(prod=5))
rice	(prod=6)
nhemba bean	(prod=30)
manteiga bean	(prod=31)
manioc	(prod=41)
sorghum	(prod=44)
maize	(prod=47)

We can find the product code by looking at **prod** in the questionnaire. Since we are only interested in those products, we need to exclude the rest of the cases about other crops. Stata uses the “**keep**” command. Once you run this command you will no longer have the complete data set available. You must remember that you should never save a file to the same name after you have selected out a set of data. You will overwrite the original data and no longer have the complete set.

The keep if command

To select just a subset of cases:

1. Click on **Data**, then **Variable Utilities**, then **Keep or drop observations**.
You should see the drop - keep or drop observations dialog box.
2. Under the **Main** tab select the round button next to **Keep Observations**
3. In the **Observations to keep if:** box type

```
prod == 5 | prod == 6 | prod == 30 |
prod == 31 | prod == 41 | prod == 44 |
```

prod == 47

The “|” is a symbol for the word OR. We are telling Stata to select all cases with **prod** equal to 47 or **prod** equal 30 or **prod** equal 31 and so on ...

4. Click on the copy button, switch to the do-file editor, paste the command, switch back to the dialog box and click on **Ok**
5. Add a comment to explain what you have done.

The Stata command is:

```
keep if prod==5 | prod==6 | prod==30 |
prod==31 | prod==41 | prod==44 | prod==47
```

Only cases with these product codes will now be used for analysis. Note that 464 observations were dropped. You can use the **tabulate** command to verify that you now have only 7 crops in the file. In the Command window you can easily type

```
tab prod
```

There should be 1,239 cases remaining for the 7 crops that are considered staple foods.

Now, we need to know how many calories were produced **per household** for all these 7 staple food products combined. To do this, we need to sum, for each household, the values of **cprod_tt** for all of the food crops the household produced. In other words, we need to create a new household-level file from the current household-product level file where there is only one case per household. Stata uses the command “collapse” to aggregate the number of cases at one level to a new level. We will sum all the cases for each household to create just one case for household.

Create a new file which is a household level file rather than a household-product level file

The collapse command

To create the new household-level file, we use **collapse**. Stata always uses the working data file as the file to be collapsed.

1. From the 1) **Data** menu select 2) **Create or change variables** then select 3) **Other variable transformation commands** then select 4) **Make dataset of means, medians, etc.**

The Collapse - Make dataset of summary statistics dialog box will appear.

2. On the **Main** tab in the **Statistics:** box for 1: change “mean” to “sum” by clicking on the drop-down arrow. In the **Variables:** box select **cprod_tt**.

- Click on the **Options** tab and in the **Grouping variables** box, select **district vil hh** in that order because those variables represent the identification of an individual household.

*The Grouping variable(s) is used to specify the variables to be used for combining cases in the collapsed file. Any cases from the original file that have identical values for all 3 of the grouping variables will be combined into a single case in the collapsed file. We want the collapsed file to have one case per household, so we use the variables that identify a household in our survey—**district**, **vil**, and **hh**.*

- Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on **Ok**.
- Add a comment to explain what you have done.

The Stata command is

```
collapse (sum) cprod_tt, by(district vil hh)
```

In the **Variables Window** you should see only 4 variables. Look at the resulting file (click on the **Data Browser** tool). You should see only one case per household. The **collapse** command created a new variable **cprod_tt**, which we calculated by summing **cprod_tt**, total calories produced, across all cases (all the different food crops) for each household. The only variables which are contained in a collapsed file are the grouping variables and any new collapsed computed variables created (e.g. **cprod_tt**). Remember to close the browser before you continue. Stata automatically added a variable label which is the function and variable used to create the resulting new variable.

You can look at the variable definitions using the **describe** command. The computed variable **cprod_tt** does not have a very descriptive label any more so we need to change the label to reflect what the variable is.

- Click on **Data**, then **Labels**, then **Label variable**.
- In the **Variable:** box **cprod_tt** should be selected
- In the **New variable label** (may be up to 80 characters) box, type
Calories Produced in Staple Foods
- Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on the **Ok** button.

5. Run the **describe** command again.

The Stata commands are:

```
describe
label variable cprod_tt "Calories produced in staple
foods"
describe
```

The new working data file now contains what we need, total number of calories from staple foods produced per household. We can also look at this variable by doing a descriptives. Use the “**summarize**” command to run a mean on the new variable **cprod_tt**. You should find that the average number of calories produced per household per year is 4,483,965.

Save this data file using the **Save As...** command.

1. Use **Save As...** from the **File** menu
2. Name the file **hh-file1**
3. Click on **Save**.
4. Copy the command from the **Results** window and paste it into the **do-file** editor, delete the reference to the directory, and add a comment to explain what you have done.

Remember to save your do-file regularly. You must be in the **Do-file** editor to save the do-file.

Step 2: Generate a household level file containing the number of adult equivalents per household.

The data needed to calculate adult equivalents per household is in the member file, C-Q1A.DTA.

1. Click on the “**Open Folder**” button on the Stata Taskbar
2. Select the file name **c-q1a.dta** and open the file.
3. Copy the command and paste it into the **do-file** editor, delete the directory reference, and add a comment to explain what you have done.

The adult equivalent value says that, on average, a female 10 to 19 years old needs only 84% as many calories as a male 10 years or older, and that children under 10 need only 60% as many calories as the typical male 10 years and older. Thus, for example, a child (male or female) under age 10 is counted as .60 adult equivalents. For each person (observation) in the member file we need to look at the variables sex, **ca4**, and age, **ca3**, to calculate adult equivalents.

The rules we will use for calculating adult equivalents for

this survey are:

Males, 10 years and older = 1.0
 Females, 10 to 19 years old = 0.84
 Females, 20 years and older = 0.72
 Children, under 10 years old = 0.60

Create a variable with the adult equivalent for each person

The generate.... if command

We will use the **Generate.../if...** command to compute the adult equivalents for each member. We will name the adult equivalent variable that we create as **ae**.

1. Select **Create or change variables** from the **Data** menu
2. Select **Create new variable**
The generate - Generate a new variable dialog box opens.
3. Under the **Main** tab, type the name of the new variable in the **Generate Variable** box:
ae
4. For the **Contents** box, type the value of **1**
5. Click on the **if/in** tab.
6. Type the statement **ca4 == 1 & ca3 >= 10**
7. Click on **OK**

The replace.... if command

Now that the new variable has been created, another command is used to assign the codes for the other adult equivalent groups that have not yet received a value. We use the **replace** command.

8. Select **Create or change variables** from **Data**
9. Select **Change contents of variable**.
The replace-Replace contents of variables dialog box opens.
10. In the **Variables** box select the name of the variable that was just created: **ae**
11. Type **.84** in the **New Contents** box
12. Click on the **If/In** tab.
13. In the **Restrict to observations if** box, type in **ca4==2 & (ca3 >=10 & ca3 <=19)**
14. Click on **Submit** . The dialog box remains open and the command is run.
15. In the **Restrict to observations if** box, change the criteria to: **ca4 == 2 & ca3 >=20**
16. Click on the **Main** tab.
17. Type **.72** in the **Contents** box .
18. Click on **Submit**. The dialog box remains open and the command is run.
19. Type **.6** in the **Contents** box
20. Click on the **If/In** tab.
21. In the **Restrict to observations if** box, change the criteria to: **ca3 <10**

22. Click on **Ok**.

The statements we need are detailed in the table below.

Numeric value	If statement
1	ca4 == 1 & ca3 >=10
.84	ca4 == 2 & (ca3 >=10 & ca3 <=19)
.72	ca4 == 2 & ca3 >=20
.6	ca3 <10

23. Copy the 4 commands from the **Results** window and paste them into the **do-file editor** and add a comment to explain what you have done.

The new variable does not yet have a variable label. To assign a variable label:

1. Click on **Data**, then **Labels**, then **Label variable**.
2. In the **Variables:** box, select the name of the variable name: **ae**
3. In the **Attach label to variable** (up to 80 characters) box, type
Adult equivalents
4. Click on the **Ok** button.
5. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on the **Ok** button. Add a comment to explain what you have done in the do-file.

The Stata commands are:

```
generate byte ae= 1 if ca4 == 1 & ca3 >=10
replace ae = .84 if ca4==2 & ca3 >=10 & ca3 <=19
replace ae = .72 if ca4==2 & ca3 >=20
replace ae = .6 if ca3 < 10
label variable ae "Adult equivalents"
```

To verify that the new adult equivalent variable, **ae**, has been calculated, display a frequency table for it.

1. From the menus click on **Statistics**, then **Summaries, tables & tests** then **Tables** then **One-way tables**
The tabulate1 - One-way Tables dialog box opens.
2. Select the variable name **ae** in the **Categorical Variables** box which is found under the tab labeled **Main**.
3. Check the box next to **Treat missing values like other values**

4. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on the **Ok** button.

The Stata command is:

```
tabulate ae, missing
```

You should see there are 1524 total cases. Ideally there should be four values represented in the table —1, .72, .84, and .60— and no missing cases. You can see we have nine missing cases. This tells us that our data file is missing either the age or the sex for nine people. This problem should have been identified during the cleaning process. At this point it would be ideal for the researcher to go back to the original questionnaires to determine the reason why these data are missing. Since we can't do this, we will use an alternative method.

Replace “missing values” with a mean value

If we leave these values missing, the total adult equivalents of those households will appear to be slightly smaller, which may distort the results. We could avoid this problem by eliminating the households with missing information from our analysis, but then we can't use the information about the food production from those households. Instead, we will try to make a reasonable assumption about those nine missing members. We know that the adult-equivalent values range from a low of .6 for children to a high of 1.0 for adult males, which is not a very wide range. We can determine the mean adult-equivalent value for the whole sample and use that value to fill in the missing data. To find out the average adult-equivalent value for our sample...

1. **Statistics** then **Summaries, tables and tests** then **Summary and Descriptive Statistics** then **Summary Statistics**
2. Select the variable: **ae**
3. Don't forget to copy the command into the **do-file editor**, then click on the **Ok** button

The Stata command is:

```
summarize ae
```

We can see that the mean value of **ae** for all individuals is .79, with a standard deviation of only .17. We will assume that the nine individuals with missing age or sex codes are all "average" individuals, and assign them the

adult-equivalent value of .79. (*Warning: be very cautious about "filling in" missing data this way. Careless use of this technique can give you misleading results. We are using this example to illustrate the use of Stata commands and not recommending that you do this routinely to compensate for missing data.*)

We will use the **Replace** command to change the system missing values (.) in the **ae** variable to .79.

1. **Data then Create or change variables then Change contents of variable**
The replace - Replace into same variable dialog box will appear.
2. Under the **Main** tab, select **ae** in the Variable: box
3. In the **New Contents** box type .79
4. Under the **if/in** tab in the **Restrict to observations if:** box type
ae==.
The "period" represents system missing.
5. Don't forget to copy the command into the do-file editor, then click on the **Ok** button.
6. Check the results of your **replace** command by rerunning the **tabulate** command.

You should see 9 cases in the frequency with a value of .79.

The Stata command is:

```
replace ae = .79 if ae==.
tabulate ae, missing
```

Calculate the adult equivalents for the household

The **collapse** command

Now we need to calculate the number of adult equivalents for each household. The current file is at the member level, but we need values at the household level. Again we use **Collapse** to go from the member level to the household level. The new variable **ae** will be calculated by summing **ae** across all members of a household.

Reminder: The **Grouping variable(s)** specify the variables to be used for combining cases in the collapsed file. Any cases from the original file that have identical values for all of the grouping variables will be combined into a single case in the collapsed file. We want the collapsed file to have one case per household, so we use the variables that identify a household in our survey—**district**, **vil**, and **hh**.

1. From the **Data** menu select **Create or change variables** then **Other variable transformation commands** then **Make dataset of means, medians, etc.**
The collapse - Make dataset of means, medians, etc. dialog box will appear.
2. On the **Main** tab in the **Statistics:** box for 1: change “mean” to “sum” by clicking on the drop-down arrow. In the **Variables:** box select **ae**.
3. Click on the **Options** tab and in the **Grouping variables** box, select **district vil hh** in that order because those variables represent the identification of an individual household.
The Grouping variable(s) is used to specify the variables to be used for combining cases in the collapsed file. Any cases from the original file that have identical values for all 3 of the grouping variables will be combined into a single case in the collapsed file. We want the collapsed file to have one case per household, so we use the variables that identify a household in our survey—district, vil, and hh.
4. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on **Ok**
5. Add a comment in the do-file to explain what you have done.

The Stata command is

collapse (sum) ae, by(district vil hh)

Collapse creates a new working file. The new working data file is at the household level, with one case per household. The variable **ae** is the total adult equivalents for that household. Look at the resulting file (click on the Data Browser tool). You should see four variables with only one case per household. You can also look at the variable definitions using the **describe** command. The computed variable **ae** does not have a very descriptive label any more so we need to change the label to reflect what the variable is.

1. Click on **Data**, then **Labels**, then **Label variable**.
2. In the **Variables:** box, select the name of the first variable: **ae**
3. In the **Attach label to variable** (up to 80 characters) box, type
Adult equivalents per household
4. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on **Ok**
5. Run the **describe** command again.

To verify that this variable was created, **summarize** the variable **ae**.

1. **Statistics** then **Summaries, tables and tests** then **Summary and Descriptive Statistics** then **Summary Statistics**
2. Variable is **ae**
3. Don't forget to copy the command into the do-file editor, then click on the **Ok** button.

You should find that the average adult equivalent over all households is 3.49.

The Stata commands are:

```
label variable ae "Adult equivalents per household"
summarize ae
```

This completes step 2. Save this file as HH-FILE2.DTA.

1. Click on **File/Save As...**
2. Filename is **hh-file2**
3. Click on the **Save** button.
4. Copy the command from the **Results** window and paste it into the **do-file editor**, delete the reference to the directory, and add a comment to explain what you have done.

The Stata command is:

```
save "hh-file2.dta"
```

If you run the syntax again and try to save the "hh-file2.dta", you will get an error message. To save to a file that already exists on the hard disk, an additional subcommand must be added, ", replace"

```
save "hh-file2.dta", replace
```

Step 3: Merge the two files created in steps 1 & 2 to compute calories produced per adult equivalent.

The merge command

We have created two files: **hh-file1.dta**, which contains the calorie-production data for all households, and **hh-file2.dta**, which contains the adult-equivalent data for all households. We need to combine these files case-by-case matching by district, village and household, to get both sets of data into one file. To do this, we use **Combine datasets / Merge datasets** under the **Data** menu choice.

We noted earlier that key variables are required for any merge. When you're joining two files which are at the

same data level, as we're about to do, it may not seem important to include key variables, but it is. The key variables determine which observations are to be combined.

Note: *You should never use **Combine datasets** without **Key Variables** because without them you have no guarantee that the program will combine the cases in the manner that you wish.*

The command will execute without any warnings or error messages, but the results may be incorrect.

At this point, if you have not closed Stata, hh-file2.dta is still the working file.

A very important point: Stata cannot merge two datasets unless they are both sorted in the order of the key variables. One way to check to see if Stata knows the file is sorted is to use the **Describe** command. In the Results window you can see at the end of the list of variables, the words “sorted by” and the list of variables that the file is sorted by. Because we created hh-file1.dta by collapsing the file, it is already sorted by district, vil and hh. hh-file2.dta was also created by collapsing the file so it is also sorted by district, vil and hh. We are ready to merge the two files.

1. Select **Data** then **Combine datasets** then **Merge Two datasets**
The Merge - Merge dataset in memory with dataset on disk dialog box will appear. The default type of merge is one-to-one on key variables – this is the merge we want to do.
2. For the Filename of dataset on disk box, click on the **Browse** button. Select the file hh-file1.dta and click on **Open**
3. In the **Key variables (match variables)** box, select

district vil hh

These are the Key Variables
4. Click on the **Options** tab. Under this tab, you see the box labeled **Specify new name of variable to mark result of merge** The default name is _merge. This variable received a code of 1 or 2 or 3 to describe what type of merge occurred. The code definition is:
 - 1 = observation is from file in memory**
 - 2 = observation is from file on disk**
 - 3 = observations are from both files**
 It is very important to look at the values in this

- variable after you have run the merge.
- Click on the copy button, switch to the **do-file editor**, paste the command, delete the directory reference, switch back to the dialog box and click on **Ok**.
 - In the do-file insert comments to remind you what you've done.

The Stata command is:

```
merge district vil hh using "hh-file1.dta", unique
```

Now that you have run the merge, run a tabulate on the `_merge` variable. You can abbreviate the name to “`_m`”, e.g. **tabulate `_m`**. You should see only the value of “3” for 343 observations. That means that there was an observation for each “district - vil - hh” combination in each of the two files.

Calculate the total calories produced per adult equivalent per household for the year

Merge Files created a new working data file. The two variables you need to compute calories produced per adult equivalent are now in the working file. **Total calories produced (`cprod_tt`)** per household for the year divided by total adult equivalents per household (**`ae`**) divided by 365 days per year gives us calories produced per adult equivalent per day (**`cprod_ae`**).

- Select **Data** then **Create or change variables** then from **Create new variable**
The generate - Create a new variable dialog box opens.
- If you see information in the dialog box, click on the **Reset** icon to clear the contents.
- Under the **Main** tab, type the name of the new variable in the **Generate Variable** box:
`cprod_ae`
- For the **Contents of new variable** box, type in **`cprod_tt/ae/365`**
- Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on **Ok**.
- Add a comment in the do-file to explain what you have done.

The new variable does not yet have a variable label. To assign a variable label:

- Click on **Data**, then **Labels**, then **Label variable**.
- In the **Variables:** box, select **`cprod_ae`**
- In the **Attach label to variable** (up to 80

- characters) box, type
Calories produced per adult equivalent per day
- Click on the copy button, switch to the do-file editor, paste the command, switch back to the dialog box and click on **Ok**.
 - Add a comment to explain what you have done.

The Stata commands are:

```
generate float cprod_ae= cprod_tt/ae/365
label variable cprod_ae "Calories produced per adult
equivalent per day"
```

Computing quartiles

The `xtile` command using `if`

Before we can produce the table we want, we have to create one more variable, denoting which calorie-production quartile each household falls into within each district. The Stata command to use is called **xtile**. This command is not available through the menus. To look at the structure of the command, we can use the Help menu.

- Click on **Help / Stata command**.
- In the **Command:** box, type **xtile** and click on **Ok**.

Under the Description heading, the definition of `xtile` is that it is a command that categorizes a variable into the specified quantiles and places the information into a new variable. Examples can be found under the Examples: heading. Since we want to divide the data into quartiles within each district, we can use the “if” subcommand, e.g.,

```
xtile quart = cprod_ae if district == 1, nq(4)
```

where

`quart` is the new variable that is created
`cprod_ae` is the variable used to rank the data
`district` is the controlling variable
`nq(4)` is short for `nquantiles(number)` which specifies the number of quantiles to use.

You must type this command in the Command window and press <Enter>.

Using the “if” subcommand works where you have only a few codes within the variable. We have 3 districts so it would not be a problem to use the `if` subcommand. What if we had 20 districts? This method would be a bit cumbersome.

Another method that could be used is a counter.

```
for z in num 1/3: xtile quartz = cprod_ae if
district==z, nq(4)
```

The `foreach` looping command

Stata provides another looping command that we can use to compute the new ranking variable. It is not available through the menus. The looping command can be found in the Programming manual and is called **foreach**. Stata added a new command in April, 2003, called **levels**. The values are stored in temporary variables called `r(levels)`. That information can be stored in a local variable and the variable used to cycle through the values.

1. Type the following command in the Command window:

```
levels district
```

The results should display the values of the districts, e.g. 1 2 3

2. Now let's store that information in a local variable. To make a temporary local level, we include the word "local" which means the variable only exists with the do-file. We need this command to be placed in the do-file. Switch to the do-file editor and type

```
levels district, local(levels)
```

3. We can now create variables containing the rank of the household within each district. We must type these commands into the do-file because the command is multiple lines. You are already in the do-file editor. Type:

```
foreach z of local levels {
  xtile quartile`z' = cprod_ae if (district == `z'), nq(4)
}
```

'z' is a local macro name which is set to each value in the variable "levels". The values we know are 1, 2, and 3. In the first loop of this programming command z is equal to 1, in the second loop z is equal to 2, etc.

quartile`z' refers to a variable name where the contents of z is appended to the name quart, e.g. quart1, quart2, quart3, etc.

district = `z', means that for the first loop district is equal to 1, for the second loop district is equal to 2, etc.

Very important note: The macro name `z' must be surrounded by a “left” single quote (found in the upper left hand corner or the keyboard to the left of the key with the number 1) and a “right” single quote (found on the key to the left of the <Enter> key). If you do not use the left single quote, you will see an error message that says in red:

```
' invalid name
```

Be sure that you end the first line with a left curly brace, e.g. { and that you place on a line by itself after all commands that you want to be included in the loop, a right curly brace, e.g. }

Since we have 3 districts, 3 new variables will be created with names of quartile1, quartile2, quartile3.

4. We want all the information in just one variable so we will create another variable and fill it with the information from the variables created above. If you created a variable called “quart” following the instructions above, you will need to drop it before proceeding (**drop quart**). The next step you are familiar with. We create a new variable and fill it with system missing.

```
gen quart = .
```

5. We now replace the data in **quart** with the data in the temporary variables. Remember, we must rerun the levels command as well since the data are temporarily stored in memory. Type the following lines, block and run them.

```
/*replace values with information from temporary
variable */
levels district, local(levels)
foreach z of local levels {
  replace quartile=quart`z' if district==`z'
}
```

This commands cycle through the values for z and replaces the contents of quart with the contents of quart1 if district is equal to 1 in the first loop, then replaces the

contents of quart with the contents of quart2 if district is equal to 2 in the second loop, then replaces the contents of quart with the contents of quart3 if district is equal to 3 for the final loop.

- The next step is to delete the temporary variables. Type the following, block and run the commands:

```
/*delete temporary variables */
  levels district, local(levels)
  foreach z of local levels {
    drop quartile`z'
```

Always check the new variables that are created to see if the values are what you expect to see. We can use the **tabulate** command with 2 variables - district and quart - to check the variables

- From the menus click on **Statistics..** then **Summaries, tables & tests** then **Tables** then **Two-way tables with measures of association**
The tabulate2 - Two-way tables dialog box opens.
- In the **Row Variable** box select **quart**
- In the **Column Variable** box select **district**
- Click on the copy button, switch to the do-file editor, paste the command, switch back to the dialog box and click on **Ok**.
- Write a comment in the do-file to explain what the commands are doing.

The number of cases in each cell should be almost the same counts plus or minus a case or two, e.g.

quart	district			Total
	monapo	ribaue	angoche	
1	28	30	29	87
2	27	30	29	86
3	27	30	29	86
4	27	29	28	84
Total	109	119	115	343

The new variable requires a label:

1. Click on **Data**, then **Labels**, then **Label variable**.
2. In the **Variables:** box, select the name of the first variable: **quart**
3. In the **Attach label to variable** (up to 80 characters) box, type
Calorie production quartile
4. Click on the copy button, switch to the do-file editor, paste the command, switch back to the dialog box and click on **Ok**.

The Stata command is:

```
label variable quart "Calorie production quartile"
```

Examples of the **foreach** looping command

Examples of the use of the **foreach** command are:

Computing new variables:

```
foreach var of varlist inc1-inc12 {
  generate tax`var' = `var' * .10
}
```

Collapsing across variables:

```
foreach qtr of numlist 1/4 {
  local m3 = `qtr'*3
  local m2 = (`qtr'*3)-1
  local m1 = (`qtr'*3)-2
  generate incqtr`qtr' = inc`m1' + inc`m2' + inc`m3'
}
```

This command computes the quarterly income variables **incqtr1-incqtr4** using the **foreach** command.

Display the final output table

We can now display a table showing the average caloric production in quartiles for each of the districts.

1. From the menus click on **Statistics** then **Summaries, tables & tests** then **Summary and Descriptive Statistics** then **Summary statistics**
The summarize - Summary statistics dialog box opens.
2. In the Variable(s): box select **cprod_ae**.
3. Click on the “**by/if/in**” tab.
4. Click in the box “**Repeat command for groups defined by**”

5. In the box below this option, select **district quart**
6. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on **Ok**.
7. Add a comment to explain what you have done.

The Stata command is:

by district quart sort: summarize cprod_ae

You should note that the mean for the 2nd quartile in Monapo is 2,539.364. The output from the **summarize** command gives you the numbers necessary for the table. However the output is difficult to read. There is another command, **table**, which can also be used to produce the final table. We will discuss this command in Section 3.

Before you save the file, you should sort the file by the key variables and then save this file as `hh-file3.dta`.

1. Sort the file by the key variables. Type in the Command window:
Sort district vil hh
2. We no longer need the variable `_merge` so it should be dropped. Type in the Command window:
drop _merge
3. Click on **File/Save As...**
4. Filename is `hh-file3`
5. Click on **Save**.
6. Copy the three commands and paste them into the do-file.
7. Close the log file. Type in the Command window

log close

8. Copy this command into the do-file.

Remember to save the contents of the **Do-file Editor** to a permanent file so you can use it another time.

1. The **Do-File Editor** should be the active window
2. Click on **File/Save As...**
3. Use the filename `session2`
The .do extension will be added automatically.

This file now contains all the commands that you pasted either from the **Command window** or from the **Review**

Document the do-file with comments

window or from dialog boxes.

Note: *Whenever you do any substantial amount of work, you should always copy the commands to a do-file and save the file so that you have documentation on what analysis you have done and so you can repeat the analysis without building all the commands again.*

Documenting the do-file with comments can save you much time trying to remember what analysis you did and why.

Let's see how you would retrieve the do-file you just created. To exit Stata, in the **Stata Editor**:



1. Click on **File** then **Exit**

Stata will prompt you if you have not saved the data file and will give you an opportunity to return to the program to save the data file. If you do not want to save your data file, click on "yes" to exit.

Start Stata again. To open our do-file:

1. Click on **Window** then **Do-file editor** then **New Do-file** or press **<CTRL 8>** or you can click on the **Do-file editor** icon on the tool bar.
The Do-file editor window will open.
2. Click on the yellow file folder tool and select the file **session2.do**
3. Click on **Open**

You can then re-execute these same commands or edit them as you wish.

There are 2 icons on the tool bar:  and . You can block lines in the do-file and click on either of these icons. The first icon ("**Run Selected Lines**") runs the commands you've blocked but in the **Results** window you do not see the commands or any output from analysis. The second icon ("**Do Selected Lines**") runs the commands you've blocked and in the **Results** window you can see which commands were run as well as the output from any analysis.

Your **SESSION2.DO** should look similar to lines below; your documentation comments may not match exactly what has been included in this listing. Comments start with `/*` at the beginning of each comment and ending each comment with a `*/`. You can also just use an `*` if the command is one line.

```

capture log close
log using session2, append

/*STATA do file – section 2 – Cross-sectional Stata Tutorial
Purpose: Calculate food production in calories per adult equivalent per day
M Beaver - January 2009 */

/* Tasks: 1) Compute total kgs produced, compute value of production in calories for specific food crops and
aggregate to the household level to obtain total food calories produced
2) Compute adult equivalents and aggregate to the household level
3) Merge the two files and calculate food production in calories per adult equivalent per day
4) Produce a table showing average food production in calories per adult
equivalent in quartiles for each district */

/* Stata recommends you include the version that the do file was written in */

version 10
clear all
macro drop _all

*modify next command to match the directory you are using
cd "C:\Documents and Settings\aec_user\My Documents\StataTraining"

*open log file
capture log close
log using "session2.smcl", replace

* turn off "more" so the whole file will run
set more off

*****
*Step 1
*****

/* open production data file */

use "C:\docs\sample\c-q4.dta", clear

/* sort variables to match by
to merge in the conversion value to convert to kgs */

sort prod p1a
tab1 prod p1a

/* rename the p1a variable to unit to match the conver data file */
rename p1a unit

joinby prod unit using "conver.dta", unmatched( master ) _merge(_merge)

*check to be sure merge done correctly
tab1 _merge

/* check to see if got what was expected using list command */
list prod unit conver if prod==47 & unit ==8

* calculate kgs produced
generate float qprod_tt= p1b * conver

/* merge in the lookup conversion value for calories and calculate total calories */

```

```

drop _merge
joinby prod using "calories.dta", unmatched( master ) _merge(_merge)

*check to be sure merge done correctly
tab1 _merge

*compute total calories produced
generate float cprod_tt= qprod_tt * calories

/* add variable labels */

label variable qprod_tt "Total production in kgs"
label variable cprod_tt "Total calories produced"

/* select only staple crops */

keep if prod == 5 | prod == 6 | prod == 30 | prod == 31 | prod == 41 | prod ==44 | prod == 47

/* check to see that there are only 7 crops listed */
tabulate prod

/* need to sum all calories produced by the household
   Using the collapse command*/

collapse (sum) cprod_tt, by(district vil hh)
label variable cprod_tt "Calories produced in staple foods"
describe

/* verify you have the right average calories produced over whole sample */

summarize cprod_tt

/* save the file */

save "hh-file1.dta", replace

*****
*Step 2
*****

/* calculating adult equivalents based on age and gender */

use "c-q1a.dta", clear

generate byte ae= 1 if ca4 == 1 & ca3 >=10
replace ae = .84 if ca4==2 & ca3 >=10 & ca3 <=19
replace ae = .72 if ca4==2 & ca3 >=20
replace ae = .6 if ca3 < 10

label variable ae "Adult equivalents"

/* check the variable */

tabulate ae, missing

/* calculate mean to determine average ae across the whole population
   To fill in the missing values*/

summarize ae

/* replace all system missing with the value of .79 */
replace ae = .79 if ae==.

```

```

tabulate ae, missing

/* need to sum the adult equivalents for each household */

collapse (sum) ae, by(district vil hh)
label variable ae "Adult equivalents per household"
summarize ae

* save file for later use */
save "hh-file2.dta", replace

*****
*Step 3
*****

use "hh-file2.dta", clear

/* now combine both the hh-file1 with hh-file2 , both files are already sorted by key variables
match files by district vil hh */

merge district vil hh using "hh-file1.dta"

*check to see which file the variables are coming from
tab _merge
*3 - variables came from both files
drop _merge

/* calculate the calories per adult equivalent per day */

generate float cprod_ae= cprod_tt/ae/365
label variable cprod_ae "Calories per adult equivalent per day"
sum cprod_ae

/* rank the new variable by district into quartiles
check for number of districts */

tabulate district
/* there are 3 districts so we want to loop 3 times */

for z in num 1/3: xtile quartz=cprod_ae if district == z, nq(4)

/*initialize variable */
gen quart=.

/*replace values with information from temporary variable */
for z in num 1/3: replace quart=quartz if district==z
for z in num 1/3: drop quartz

/* check results - should see equal number of cases in each category */
tabulate quart district
label variable quart "Calorie production quartile"

/* produce the table */
by district quart sort: summarize cprod_ae

/* sort file by key variables */
sort district vil hh
save "hh-file3.dta", replace

*close log file
log close

```

Exercise 2.1

Produce similar output using calories retained (production minus sales) instead of calories produced. It will show calories retained per adult equivalent per day from the total of the same six food crops. The output should be broken down by district and calorie production quartile.

Hints:

- a. The procedure is very similar to the work that we just completed. Open a new do-file to save your commands for this exercise.
- b. Sales come from `c-q5.dta`.
- c. Check the file for the appropriate variable for the quantity of sold production. Note that the product codes are the same as for `c-q4.dta`. Also check for the variables by which to sort.
- d. You can start from a blank file and build all the commands necessary to produce the calories retained, or you can copy the commands used to generate the table from section 2 and adjust the commands as necessary to calculate the calories retained. Changes must be made for file names and variables.
- e. Computing the calories sold involves the same basic steps as computing the calories produced. (Step 1)
- f. Merge this newly created file, (the file containing calories sold), with the file containing calories produced, `hh-file3.dta`. Check the `_merge` variable (tab `_merge`) and explain why you see more than one value.
- g. Keep in mind that only 256 households sold products, but all 343 households produced and retained calories. If the “calories sold” variable is missing, it means the household did not sell food, so it should be recoded to zero.
- h. Compute `calories retained = calories produced - calories sold`. The average calories retained per adult equivalent for the whole population should be 3044.233
- i. Rank into quartiles.
- j. Use the **Tabulate** command to show calories retained by **district** and **quartile**.
- k. Save the data file to the name, `hh-file4.dta`.
- l. Save the contents of the do-file editor to a new name reflecting the name of the exercise.

Below is an example of the output you should produce:

```

-> district = monapo, quarts = 1
Variable |      Obs      Mean   Std. Dev.      Min      Max
-----+-----
cret_ae |        28  1171.574   420.7985   224.4898  1806.867

```

```

-> district = monapo, quarts = 2
Variable |      Obs      Mean   Std. Dev.      Min      Max
-----+-----
cret_ae |        27  2239.088   199.4202   1888.33   2554.892

```

```

-> district = monapo, quarts = 3
Variable |      Obs      Mean   Std. Dev.      Min      Max
-----+-----
cret_ae |        27  3343.003   461.9159   2685.971  4303.122

```

```

-> district = monapo, quarts = 4
Variable |      Obs      Mean   Std. Dev.      Min      Max
-----+-----
cret_ae |        27  7619.101   3557.135   4359.737  20873.97

```

```

-> district = ribaue, quarts = 1
Variable |      Obs      Mean   Std. Dev.      Min      Max
-----+-----
cret_ae |        30  1251.391   358.8783   429.2929  1790.432

```

```

-> district = ribaue, quarts = 2
Variable |      Obs      Mean   Std. Dev.      Min      Max
-----+-----
cret_ae |        30  2171.697   205.3644   1835.298  2566.006

```

```

-> district = ribaue, quarts = 3
Variable |      Obs      Mean   Std. Dev.      Min      Max
-----+-----
cret_ae |        30  3165.192   330.2283   2578.604  3731.045

```

```

-> district = ribaue, quarts = 4
Variable |      Obs      Mean   Std. Dev.      Min      Max
-----+-----
cret_ae |        29   5828.97   1632.9     3825.879  9464.901

```

```

-> district = angoche, quarts = 1
Variable |      Obs      Mean   Std. Dev.      Min      Max
-----+-----
cret_ae |        29   929.4182   388.3228   207.9077  1395.962

```

```

-> district = angoche, quarts = 2
Variable |      Obs      Mean   Std. Dev.      Min      Max
-----+-----
cret_ae |        29  1718.789   166.1601   1447.059  1984.059

```

```

-> district = angoche, quarts = 3
Variable |      Obs      Mean   Std. Dev.      Min      Max
-----+-----
cret_ae |        29  2442.247   347.8035   1997.711  3063.996

```

```

-> district = angoche, quarts = 4
Variable |      Obs      Mean   Std. Dev.      Min      Max
-----+-----
cret_ae |        28   5022.29   2443.454   3134.742  12674.86

```

STATA 10 SAMPLE SESSION

SECTION 3 – Tables and Other Types of Analysis

Tables

Using the **Table** command you can calculate various statistics and present them in a variety of ways that are completely under your control. **Table** allows you to choose how you want to assemble variables and statistics for display in rows, columns, and super-columns or super-rows. A super-column or super-row has a variable nested below it. Variables can be stacked or nested. Nested means that all of the values for one variable are displayed below the individual values of another variable. You can manipulate table structure, content, and presentation format.

With this command there are a few limitations:

- a) up to 4 variables can be specified in the by()
- b) up to 5 statistics can be displayed in each cell
- c) the sum of the number of rows, columns, super-columns, and super-rows is called the number of margins. A table may contain up to 3000 margins, e.g. a one-way table may contain 3000 rows, a two-way table may contain 2998 rows and 2 columns, or 2997 rows and 3 columns and so forth

Commands that produce similar results are:

- tabstat - displays summary statistics for a series of numeric variables in a single table
- tabsum - produces one- and two-way tables of means and standard deviations - this command is faster, but the table command is more flexible
- tabulate - one- and two-way tables of frequencies
- tab1 - produces one-way tabulation for each variable
- tab2 - produces two-way tabulations of all combinations of the variables

Let's compare the **tabulate** command with the **table** command to create two-way tables.

Open the member file we created from Section 1 that contains the age variable, q1a-age.dta.

1. **File/Open...**
2. Select q1a-age.dta
3. Click on **Open**
4. Copy the command, paste it into a new do-file and add comments. Add all the initial commands that

should be incorporated with every new do-file. Refer to the do files you have already created. You can copy several of the commands that you need and comments. Remember to start the log file for this session.

First, do a simple two-way table using the **tabulate**.

1. From the menus click on **Statistics** then **Summaries, tables & tests** then **Tables** then **Two-way tables with measures of association**
The tabulate2 - Two-way tables dialog box opens.
2. In the **Row Variable** box select **ca2**
3. In the **Column Variable** box select **age_gp**
4. Under Cell Contents click in the box next to **Within column relative frequencies** to put a .
5. Click in the box next to **Within row relative frequencies**.
6. Click on the copy button, switch to the do-file editor, paste the command, switch back to the dialog box and click on **Ok**.

The Stata command is:

```
tabulate ca2 age_gp, column row
```

Below is the output.

```
+-----+
| Key   |
+-----+
|       |
|  frequency |
| row percentage |
| column percentage |
+-----+
```

relation to head	Age group				Total
	0 to 10	11 to 19	20 to 60	61 and ol	
head	0 0.00 0.00	6 1.75 2.22	296 86.30 47.13	41 11.95 83.67	343 100.00 22.57
wife/husband	0 0.00 0.00	25 8.06 9.26	280 90.32 44.59	5 1.61 10.20	310 100.00 20.39
son/daughter	503 70.06 87.78	184 25.63 68.15	31 4.32 4.94	0 0.00 0.00	718 100.00 47.24
mother/father	0 0.00 0.00	0 0.00 0.00	5 83.33 0.80	1 16.67 2.04	6 100.00 0.39
other relative	70 48.95 12.22	55 38.46 20.37	16 11.19 2.55	2 1.40 4.08	143 100.00 9.41
Total	573 37.70 100.00	270 17.76 100.00	628 41.32 100.00	49 3.22 100.00	1,520 100.00 100.00

The **table** command

Let’s use **table** to produce a similar table. However with the **table** command we cannot ask for row or column percentages – the **table** command is generally used for summary statistics. Frequency and Totals are possible to select from this command.

1. From the menus click on **Statistics..** then **Summaries, tables & tests** then **Tables** then **Table of summary statistics (table)**.
2. Under the **Main** tab select **ca2** in the Row variable: box
3. Click in the box next to **Column variable** and select **age_gp** in the box below.
4. In the **Statistics section, #1**, select **Frequencies** from the drop down box.
5. Under the **Options** tab check **Add row totals** and also check **Add column totals**
6. Click on the copy button, switch to the do-file editor, paste the command, switch back to the dialog box and click on **Ok**.
7. Write a comment in the do-file to explain what the command does.

The Stata command is:

```
table ca2 age_gp, contents( freq ) row col
```

Note: the word “contents” can be abbreviated to “c”, e.g. c(freq).

The results are:

relation to head	Age group				Total
	0 to 10	11 to 19	20 to 60	61 and older	
head		6	296	41	343
wife/husband		25	280	5	310
son/daughter	503	184	31		718
mother/father			5	1	6
other relative	70	55	16	2	143
Total	573	270	628	49	1,520

Comparison of the commands **summarize**, **tabulate** and **table**

The following is a comparison of computing averages using **summarize**, **tabulate** and **table**, based on an example from section 2.

1. Click on **File** then **Open**
2. Select hh-file3.dta, Click on **Open**
3. Copy the command and paste it into the do-file editor

First we will use the **summarize** command:

1. From the **Statistics** menu select **Summaries, tables & tests** then **Summary and descriptive statistics** then **Summary statistics**

- The summarize - Summary statistics dialog box opens.
2. Select **cprod_ae** in the “**variables**” box.
 3. Be sure that the under “Options” in this tab, **Standard Display** has been selected.
 4. Click on the “**by/if/in**” tab.
 5. Click in the box “**Repeat command for groups defined by**”
 6. In the box below this option, select **district quart**
 7. Click on the copy button, switch to the do-file editor, paste the command, switch back to the dialog box and click on **Ok**.

For each combination of district and quart, we see the summary statistics. This output is difficult to read.

Next we will use the **tabulate** command:

1. From the menus click on **Statistics** then **Summaries, tables & tests** then **Tables** then **One/two-way table of summary statistics**

The tabsum - One/two-way table of summary statistics dialog box opens.

2. In the Variable 1: box select **district**
3. In the Variable 2 (optional): box select **quart**.
4. In the Summarize Variable: box select **cprod_ae**.
5. For output we are only interested in the mean, so check the boxes next to
 - ✓ Suppress standard deviation
 - ✓ Suppress frequencies
 - ✓ Suppress number of observations
5. Click on the copy button, switch to the do-file editor, paste the command, switch back to the dialog box and click on **Ok**.
- 6.

In the Results window we see:

Means of Calories per adult equivalent per day

district	Calorie production quartile				Total
	1	2	3	4	
monapo	1248.5475	2539.3641	3997.4884	9150.0217	4206.4673
ribaue	1502.242	2554.488	4062.3014	7607.719	3900.7966
angoche	1297.9691	2465.509	3698.807	8495.49	3950.2608
Total	1352.5022	2519.7353	3919.3795	8399.3828	4014.5181

Notice that the number of decimals is not uniform. We can fix that with the **table** command.

1. From the menus click on **Statistics** then **Summaries, tables & tests** then **Tables** then **Table of summary statistics (table)**.
2. Press the Reset button  to clear the boxes .

3. Under the **Main** tab select **district** in the **Row variable:** box
4. Click in the box next to **Column variable** and select **quart** in the box below.
5. In the **Statistics section, #1**, select **Mean** from the drop down box.
6. In the box to the right specify the variable to use for the **Mean** statistic - **cprod_ae**

We would also like to see the minimum and maximum values.

7. Click on the drop down box next to #2 and scroll down to **Maximum** and select that statistic. For the variable select **cprod_ae**
8. Click on the drop down box next to #3 and scroll down to **Minimum** and select that statistic. For the variable select **cprod_ae**
9. Under the **Options** tab check **Add row totals** and also check **Add column totals**
10. To format the numbers, check next **Override display format for numbers in cells**. Click on the **Create** button to the right of this box.
11. In the “**Create a display format**” dialog box , check “**Customize format**” and change the contents to read: **%11.2fc**. The **Help format...** button shows different formats that can be specified. This format says to use a width of 11 with 2 decimals. (fc) means fixed format with a comma. Click on **Ok**.
12. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on **Ok**.

The three Stata commands are:

```
by district quart sort: summarize cprod_ae

tabulate district quart, summarize(cprod_ae)
nostandard nofreq noobs

table district quart, contents( mean cprod_ae max
cprod_ae min cprod_ae ) row col format(%11.2fc)
```

For each district, the first row is the mean, the second row is the max and the third row is the min.

TABLE:1 Food Production in calories per adult equivalent per day (Mean, Maximum and Minimum)

district	Calorie production quartile				Total
	1	2	3	4	
monapo	1,248.547	2,539.364	3,997.488	9,150.021	4,206.467
	1,972.673	3,175.779	5,066.724	28,465.750	28,465.750
	294.101	1,984.114	3,225.948	5,107.123	294.101
ribaue	1,502.242	2,554.488	4,062.302	7,607.719	3,900.797
	2,030.398	3,141.388	4,983.722	13,123.971	13,123.971
	429.293	2,082.420	3,190.407	5,151.591	429.293
angoche	1,297.969	2,465.509	3,698.807	8,495.490	3,950.261
	2,023.654	2,996.365	4,691.524	20,485.100	20,485.100
	353.882	2,037.201	3,009.462	5,021.753	353.882
Total	1,352.502	2,519.735	3,919.379	8,399.383	4,014.518
	2,030.398	3,175.779	5,066.724	28,465.750	28,465.750
	294.101	1,984.114	3,009.462	5,021.753	294.101

Print a table from the Viewer

The **table** command permits you to specify more than one variable to summarize and also permits formatting of the contents of the table.

A simple way to print a table you have just created, is to open the **Viewer**, select the table and print.

1. Open the Viewer - Click on **File** then **View**. A dialog box opens, asking for the name of the file
2. Click on the **Browse** button and select the file **session3.smcl** and click on **Ok**
3. Scroll down to the table you want to print and block it.
4. Click on **File** then **Print** then **Viewer**. The **Print** dialog box opens. Under **Page Range** click on the radio button next to **Selection**. Then click on **Print**.
5. Another dialog box opens labeled **Output Settings** In this box you can specify a Header, a Name and a Project. If you do not want line numbers and the Stata logo to print, you should remove the ticks next to the boxes labeled **Print Line #'s** and **Print Logo**.
6. Click on **Ok** to print the selection.

Exercise 3.1

Produce a similarly formatted table using calories retained using the data file that was created in Exercise 2.1. **Include totals** by retained quartile. Your table should look similar to the table below:

district	Calories retained quartile				Total
	1	2	3	4	
monapo	1171.574	2239.088	3343.003	7619.102	3570.975
	1806.867	2554.892	4303.122	20873.971	20873.971
	224.490	1888.330	2685.971	4359.737	224.490
ribaue	1251.391	2171.697	3165.192	5828.970	3081.416
	1790.432	2566.006	3731.045	9464.901	9464.901

	429.293	1835.298	2578.604	3825.879	429.293
angoche	929.418	1718.789	2442.247	5022.290	2506.498
	1395.962	1984.059	3063.996	12674.862	12674.862
	207.908	1447.059	1997.711	3134.742	207.908
Total	1118.378	2040.130	2977.233	6135.476	3044.233
	1806.867	2566.006	4303.122	20873.971	20873.971
	207.908	1447.059	1997.711	3134.742	207.908

Multiple Response Questions

1) Multiple dichotomy (yes/no questions)

One of the types of question used in survey research asks the respondent to select multiple answers. A single variable cannot record the answers to this type of question adequately, because a variable can have only one value. The solution is to record each possible response in a different variable. The responses can be analyzed separately using commands you have already seen (**tabulate**), but ideally we want to analyze these related variables jointly.

Multiple dichotomy - yes/no questions: If a survey question asks the respondent to "check all that apply" from a set of ten choices, a separate variable is required for each of the ten responses. Each variable has a value to indicate whether the response was checked (1) or yes, or not checked (2) or no. An example of this type of question can be found in the household level survey questions (see appendix), Section V - Agricultural Sales, question 64 - have you increased the quantities sold over the last five years? All of the variable names associated with this question begin with H64.

Open the file:

1. Select **File** then **Open**
2. Select c-hh.dta
3. Click on **Open**.
4. Copy the command and paste it in the do-file editor. Delete the directory reference.

In this survey 1 = yes and 2 = no. Questions you might ask are:

- A. How many respondents increased sales quantities of maize?

To answer this question you can count the number of times the value of 1 appears in the variable associated with maize. To count the number of times a value appears in the variable. The command is **count**

The **count** command

```
count if h64a == 1
```

In the **Results** window you see the value of 86. You could also run a frequencies:

```
tabulate h64a
```

The tabulate shows that 147 did not increase sales of maize as well as 86 households who did. Now we change the question.

B. How many crops increased in sales within the household?

For this question we can sum the number of 1's in the variables associated with this question using the **egen** command. We need to recode the value of 2 to 0.

Recode:

1. Select **Create or change variables** from the **Data** menu
2. Select **Other variable transformation commands** then **Recode categorical variable**
The recode - Recode categorical variable dialog box opens.
3. Under the **Main** tab, click in the Variables box and select all the variables that start with h64, e.g. h64a h64b h64c h64d h64e h64f h64g h64h
4. In the box for **Required:** type **(2=0)**
5. Click on the copy button, switch to the do-file editor, paste the command, switch back to the dialog box and click on **Ok**.

The recode command

The Stata command is:

```
recode h64a h64b h64c h64d h64e h64f h64g h64h (2=0)
```

The values in the h64x variables are either 0=no or 1=yes. We are ready to count the number of crops that increased in sales:

The egen command

1. Select **Create or change variables** from the **Data** menu
2. Select **Create new variable (extended)**
The egen - Extensions to generate dialog box opens.
3. Under the **Main** tab, type the name of the new variable in the **Generate Variable** box: **ncrops**
4. For the **egen function** box, scroll down and highlight **Row total**.
5. In the box for **Generate variable as type:** select **integer**
6. Click on the **egen function argument - Variables** box type **h64***
6. Click on the copy button, switch to the do-file editor, paste the command, switch back to the dialog box and click on **Ok**.

The Stata command is:

```
egen ncrops = rowtotal(64*)
```

Now you can do a frequencies on the new variable:

```
tabulate ncrops
```

Two households had increased sales on all crops (ncrop = 8), 136 households had no increased sales on any of the crops (ncrop = 0).

C. What is the distribution of the crops?

The **tabstat** command

For this question we can use the **summarize** command, but we could also use the **tabstat** command:

1. From the menus click on **Statistics..** then **Summaries, tables & tests** then **Tables** then **Table of summary statistics (tabstat)**.
2. Under the **Main** tab type **h64*** in the Variables: box
3. Under “Statistics to display” place a tick in the first box and select **Sum** as the statistic.
4. Under the **Options** tab in the **Use as Columns** change to **Statistics**.
5. Click on the copy button, switch to the do-file editor, paste the command, switch back to the dialog box and click on **Ok**.

The Stata command is:

```
tabstat h64*, statistics( sum ) columns(statistics)
```

You see all the h64* variables with a count of the number of cases where yes was specified. Manioc (h64b) was the most frequent crop for which households had increased sales (115), sorghum (h64g) was the least (14).

2) Multiple response

Multiple Response: The other type of multiple response question is where the survey question asks the respondent to "list up to xx choices" from a set of ten choices. If four responses are requested, four variables must be used to code the responses. The set of possible responses are assigned sequential values and the same set of values are used for each of the 4 variables. The respondent must record a different value in each of the 4 variables. These types of variables are called *multiple response* variables.

Question 35 in the household questionnaire is an example of a multiple response question. It asks about crops grown

principally to be sold. Each household is asked to specify up to three main crops which are coded into variables **h35a**, **h35b**, and **h35c**. Codes are provided for five of the most common crops. The question is left open-ended, however, since a code of 6 is allowed for a crop not on the list. The name of the crop is written down on the questionnaire and later assigned a code.

Because the question was open ended, more categories were added to these variables than what appears in the annex. After the data are collected, the researcher assigns a code to each of the crops specified for "6-other" - this procedure is called "post-coding". Codes and value labels are entered into the data file and the data changed from the value of 6 to the appropriate code. As you will see, using the **tab1** command, eleven different crops were specified for question 35.

Stata does not have an official command that will tabulate data collected in this format. We can do frequencies of each variable or develop commands to pull out specific information. There is a user-written "ado" command called **tabw** (Peter Sasiemi, STB-25; Stata 3.1). For each variable in a list of 2 variables, this command will tabulate the number of times it takes on the values 0, 1, ..., 9; the number of times it is missing; and the number of times it is equal to some other value. String variables are not tabulated but are identified at the end of the displayed table. To download this ado file, connect to the Stata website.

1. Click on **Help** then **SJ and User Written Programs**.
2. In the Viewer click on **STB**
3. Scroll down to **stb25**. and click.
4. Click on **sg36** "Tabulating the counts of multiple categorical variables". In this screen, click on (**click here to install**)
5. The program will be installed in the directory:
C:\ado\plus\t
6. To use the program, in the **Command** window type

```
tabw h35a h35b h35c
```

The output is:

Variable	0	1	2	3	4	5	6	7	8	9	****	.
h35a	0	86	49	0	0	44	23	1	2	1	1	136
h35b	0	4	35	1	1	39	8	1	4	1	1	248
h35c	0	0	1	2	0	2	10	2	1	2	1	322

Using this type of analysis you could state the following: Cotton was the most frequent primary cash crop. 90 households grew this crop, peanuts and rice were the next most often grown for cash. You can also use the **tab1** command.

```
tab1 h35*
```

Other Types of Analyses

Weights

Stata provides for a method to analyze data using different types of weights. The type of weight that is to be used with a set of data will depend on the type of sampling that has been used.

See the table below for an explanation of the available weight types.

Sub-Command	type of weight	Definition
<u>f</u> weight or <u>f</u> requency	frequency weights	Number of replicated observations, this value is always an integer. If the fweight associated with an observation is 5, it means the observation represents 5 identical observations.
<u>p</u> weight	Sampling weights	inverse of the probability that this observation is included in the sample due to the sampling design. A pweight of 100 indicates that this observation represents 100 subjects in the population. There are qualifications to this weight when used with survey analysis commands
<u>a</u> weight or <u>c</u> ellsize	analytic weights	inversely proportional to the variance of an observation. The observations typically represent averages and the weights are the number of elements that produced the average
<u>i</u> weight	Importance weights	relative “importance” of the observation. This weight is generally used by programmers who want to produce a specific computation.

To read more about weights look at the User manual - weights. If you use the generic “weight” sub-command, Stata will tell you which weight it assumes you want to use. Not all commands will allow a weight to be included. The format is

```
[type_of_weight=variable_in_file].
```

Let's use one of the Stata's sample data files to explore this sub-command.

1. Click on **File** then **Open**. A dialog box should open telling you that the data have changed and **do you want to continue and lose unsaved data**. We don't want to save any changes to the data file, click on **Yes**.
2. Select `census.dta`.
3. Click on **Ok**. - Remember to copy the command and paste it into the do-file editor.

Use the **Browse** button to look at the data. There is one observation for each state. The variable called **pop** is the total population for the state. The variable called **medage** is the median age of the population. First let's get the population-weighted mean.

1. From the **Statistics** menu select **Summaries, tables & tests** then **Summary and descriptive statistics** then **Summary statistics**.
The Summarize - Summary Statistics dialog box opens.
2. Select **medage** in the "**variables**" box.
3. Click on the "**weights**" tab.
Note that only 3 types of weights are available to choose from. There is also a help button on weights.
4. Select **Analytic weights**.
5. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on **Submit**.

Look at the output. The sum of the weight is 225,907,472. This is the population of the U.S. in the 1980 census. The weighted mean is 30.11. Now, return to the dialog box.

6. Click on **None** under the **Weight** tab.
7. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on **Ok**.

The unweighted mean is 29.54. The Stata commands are:

```
use "census.dta", clear
summarize medage [aweight=pop]
summarize medage
```

Survey weights are discussed in the next section.

Indicator variables

An indicator variable is a special case of a categorical variable. An indicator variable has two groups only, whereas other categorical variables can have more than two groups. Usually the values in indicator variables are 0 and 1 or no/yes.

Examples of indicator variables are:

Converting continuous variables to indicator variables

Is a person a citizen of the U.S.? (no/yes).
Does a farmer use fertilizer? (no/yes).

Stata can convert continuous variables to categorical and indicator variables and it can also convert categorical variables to indicator variables.

Suppose we want to create a new variable that indicates whether a person is 18 years old or older. You could have generated a new variable and assigned it a value of 1 if `ca3 >=18`. Then you would need a second step to recode the system missing to 0. There is another way to create this variable.

We will use the file `c_q1a.dta`. Open the file and then create a new variable using the **generate** command following the steps below:

1. Click on **File** then **Open** .
2. Select `c_q1a.dta` and click on **Open**. Copy the command to the do-file editor. Delete the reference to the directory.
3. Check to see if there are any missing values in the age variable - **ca3**. Use the list command
list if ca3 >=.

We are checking to see if there are missing values because Stata considers missing values to be greater than any number.

4. Select **Create or change variables** from the **Data** menu
5. Select **Create new variable**
The generate - Generate a new variable dialog box opens.
6. Under the **Main** tab, type the name of the new variable in the **Generate Variable** box: **age18p**
7. For the **Contents** box, type in
ca3>=18
8. Click on the **Generate variable as type** drop down box and change to **byte**.
9. Click on the copy button, switch to the do-file editor, paste the command, switch back to the dialog box and click on **Ok**
10. Run a **tabulate** to look at the results.
Note: if there had been a missing value for an observation, that observation would have been assigned a value of 1.

It would have been better to put a qualifier on the command to assign the values to cases where `ca3` was not missing (e.g. `ca3 < .`).

```
generate byte age18p = ca3>=18 if ca3 < .
tab1 age18p
```

Then, any missing values in **ca3** would also be missing in the new variable **age18p**.

Converting categorical variables to indicator variables

Suppose that you want to do regression analysis and control for effects of the different geographic regions. We have a variable called **district** which has 3 categories. We want to create indicator variables for the three districts. These types of variables are also called dummy variables. First let's run the **describe** command to look at the contents of the file:

```
describe
```

Next let's look at the values and labels for the variable **district**:

```
label list district
```

To make 3 indicator variables we can type:

```
tabulate district, generate(district)
```

Now, run the **describe** command again:

```
describe
```

Three new variables have been created, called **district1**, **district2**, and **district3**. We can examine the variables using the **tab1** command.

```
tab1 district*
```

The variables **district1**, **district2**, and **district3** can now be used for regression analysis as dummy variables. They contain either a 0 or a 1.

Stata 10 SAMPLE SESSION

SECTION 4 - Table and Graphs - how to bring them into a word processor, and survey estimation, accounting for design effects**How to move Stata results into other applications**

The objective of this section is to give you the tools necessary to prepare reports, i.e. to learn how to move Stata results into other applications. The method is simple: once a graph or a table has been produced, it can be printed or incorporated into reports prepared using word processors or publishing programs. Incorporating tables from Stata can be done using the copy and paste procedure. You should save the log file as well in case you need other tables that were created. Find the table in the session3.smcl file that showed the count of the “relation of head” to “age group” cross-tabulation:

Tables

1. Click on **File** then **Log** then **View...** In the **Choose File to View** dialog box, click on the **Browse** button.
2. Select session3.smcl .
3. Click on **Open** , Then click on **Ok**
4. Locate a table that you want to copy to your word processor. Use your mouse to block the table.
5. Press **Ctrl-C** (copy). This key sequence copies what you have blocked.
6. Now open your word processor software if it is not already open.
7. Place your cursor where you want the table to appear. Press **Ctrl-V** (paste) to paste the table.
8. In your word processor, block the text that you just pasted. Now change the font to a fixed font, e.g. Courier New or Letter Gothic. Click on **Format, Font**, and select the font. The size of the font may need to be adjusted depending on the margins of your paper. The default will be 12 and you may want to select 10 or 9 or 8.

Below is an example of a table copied into a word processor before the font is changed to a fixed pitch:

relation to head	age_gp				Total
	0 to 10	11 to 19	20 to 60	61 and older	
head		6	296	41	343
wife/husband		25	280	5	310
son/daughter	503	184	31		718
mother/father			5	1	6
other relative	70	55	16	2	143
Total	573	270	628	49	1,520

Below is the same table after the font is changed to a “fixed pitch” and the font size is adjusted so that the table will fit on the page.

relation to head	Age group				Total
	0 to 10	11 to 19	20 to 60	61 and older	
head		6	296	41	343
wife/husband		25	280	5	310
son/daughter	503	184	31		718
mother/father			5	1	6
other relative	70	55	16	2	143
Total	573	270	628	49	1,520

Copying tables from the Results window

You can also copy the information from the **Results** window into your word processor. Stata provides three choices from the Edit menu for copying tables. Click on **Edit** to look at the choices.

1. Copy text - Ctrl-C - copies the table as straight text.
2. Copy table - Shift-Ctrl-C - copies the table and includes tabs where it thinks there should be tabs
3. Copy table as HTML - Shift-Ctrl-Alt-C - copies the table into HTML format.

You may encounter problems with the second and third options if you use these. Stata determines if there should be tabs and may not make the correct decision. You might need to increase the width of the columns in the output to make sure that tabs are included. Below is an example of the same table using the Shift-Ctrl-C

relation to head	Age group				Total
	0 to 10	11 to 19	20 to 60	61 and ol	
head	0	6	296	41	343
	0.00	1.75	86.30	11.95	100.00
	0.00	2.22	47.13	83.67	22.57
wife/husband	0	25	280	5	310
	0.00	8.06	90.32	1.61	100.00
	0.00	9.26	44.59	10.20	20.39
son/daughter	503	184	31	0	718
	70.06	25.63	4.32	0.00	100.00
	87.78	68.15	4.94	0.00	47.24
mother/father	0	0	5	1	6
	0.00	0.00	83.33	16.67	100.00
	0.00	0.00	0.80	2.04	0.39

Quite a bit of editing is required to make the above table presentable.

Using Excel to create columns from the table

Another method is to copy the table from the Results window into Excel. Then use the method to convert text to columns that is provided in Excel. The left most cells in each column will contain the text for the entire row. Click on “Data”, then “Text to Columns”. The “Text to Columns Wizard will start. Follow the instructions in the wizard to divide the text into columns. Upon completion of the wizard, block the columns, copy and paste into your word processor. The table will be now in a Table in the word processor where you can easily manipulate the widths and other formatting as required. Below is an example of output from the Results window converted to columns in Excel and pasted into Word.

Relation to head	0 to 10	11 to 19	20 to 60	61 and older	Total
head	0	6	296	41	343
	0	1.75	86.3	11.95	100
	0	2.22	47.13	83.67	22.57
wife/husband	0	25	280	5	310
	0	8.06	90.32	1.61	100
	0	9.26	44.59	10.2	20.39
son/daughter	503	184	31	0	718
	70.06	25.63	4.32	0	100
	87.78	68.15	4.94	0	47.24
mother/father	0	0	5	1	6
	0	0	83.33	16.67	100
	0	0	0.8	2.04	0.39
other relative	70	55	16	2	143
	48.95	38.46	11.19	1.4	100
	12.22	20.37	2.55	4.08	9.41
Total	573	270	628	49	1,520
	37.7	17.76	41.32	3.22	100
	100	100	100	100	100

Exercise 4.1.

Select another table from your Session3.SMCL file. Use all three methods to copy another table from your log file into a word processor.

Graphs

The process to copy output to a word processor is basically the same for Graphics, such as pie charts and histograms, but there is more flexibility in the ways to save the file, along with more difficulties in getting just the look you want. As an example, we will look at the distribution of cashew tree ownership across households in the Mozambique data, using a histogram.

Open a new do file and place the requisite information at the top, e.g.

```

capture log close
log using session4, append

/* session 4 - copying Tables and Graphs to
a word processor */
/* tasks:
/* Your name - date */

version 10

/* set memory if you need to */

clear all

macro drop _all

/* set the directory where you will work */

cd "C:\Documents and Settings\aec_user\My
Documents\StataTraining"

```

Save this do file under the name **session4.do**.

We are now ready to open the household file that contains the tree ownership variable, **c-hh.dta**.

1. Click on **File/Open**
2. Select **c-hh.dta** and click on **Open**
3. Paste the command from the **Results** window to the **do file editor**. Remove the directory reference.

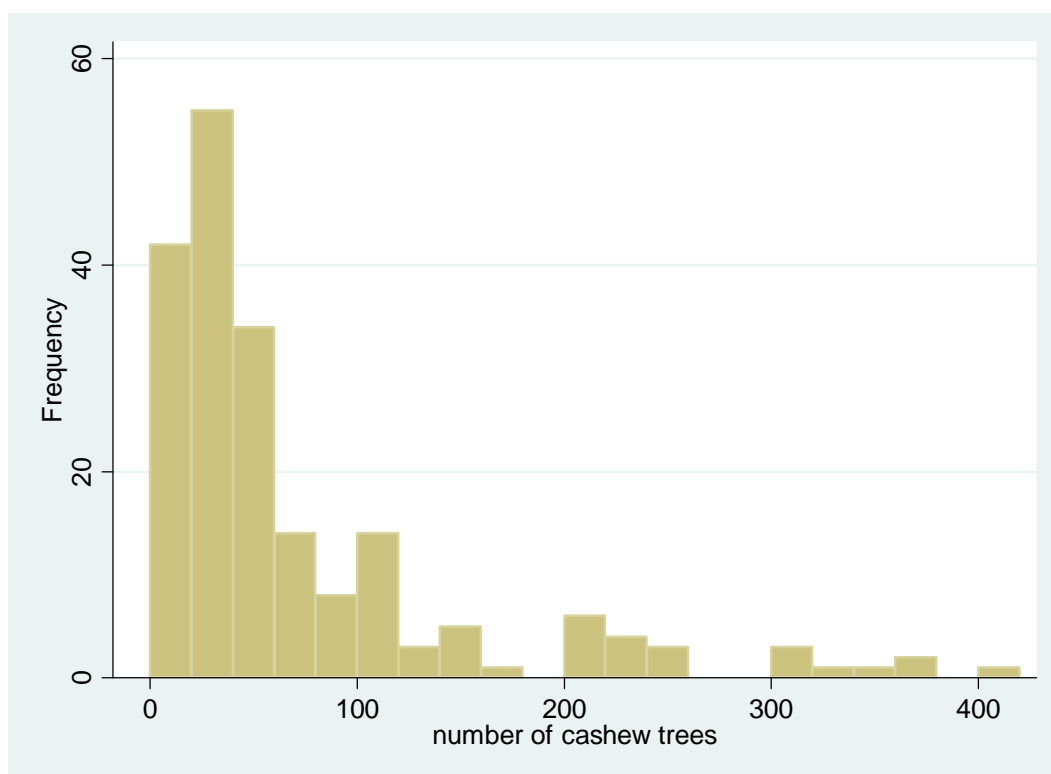
Create the Histogram chart using the variable **H57** (number of trees owned):

4. Select **Graphics** then **Histogram**.
5. In the Variable box select **H57** (Number of cashew trees) .
Note: you can specify whether the variable is continuous or discrete.
6. Under **Bins**, check the box next to **width of bin** and type **20**
7. Under **Y-Axis**, click on the radio button next to **Frequency**
8. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on **Ok**.

The Stata command is:

```
histogram h57, frequency width(20)
```

Another window will open and you should see a histogram chart that looks like the one below:



To copy this graph to your word processor,

1. Click on **Edit / Copy graph**.
You could also right-click on the graph itself and select "Copy Graph".
2. Open your word processor and click on **Edit / Paste**

You will not be able to edit this graph, other than the size, placement, wrapping of text and other basic aspects allowed by your word processor.

You can also save a Stata chart to a file.

1. Click on **File** then **Save**.
2. A dialog box opens where you can type the name of the file. The default extension is `.gph` which is the format that Stata recognizes as a graph file. If you save the file with a `.gph` extension, you can then open the graph again within Stata. In the filename box, type **Cashew_trees** and click on **Save**.
3. Copy the command from the results window, switch to the **do-file editor** and paste the command.

You can also save the graph into several different formats, such as "windows metafile (wmf)". Click on the drop-down arrow next to the box labeled as "Save as Type": to see the different formats. Word processors can import a graph with the extension `wmf` or `tif` into a graphic box.

Once the graph window has been closed, you cannot reopen it unless you have saved the graph to a file. You can rerun the

Scatter plot using “by” subcommand

command that created the graph to see the graph again. You cannot have more than one graph window open at a time.

Let’s look at another graph. We will use the file created in the last session, `hh-file4.dta`. We can plot adult equivalents per household with total calories produced.

1. Click on **File** then **Open**
2. Select `hh-file4.dta` and click on **Open**. Copy the command to open the file to the do-file editor.
3. Select **Graphics / Two-way graphs (scatterplot, lines, etc.)**.
4. The dialog box opens for the twoway graph. Click on the **Create** button to define the graph. The default type of plot is **scatter**. You could pick Line, Connected, Area, Bar, Spike, or Dropline.
5. For the **Y variable** select **ae** from the dropdown arrow, for the **X variable**, select **cprod_tt** from the dropdown arrow.
6. Click on the **Accept** button. You now see that **Plot 1** has been defined and is highlighted.
7. Click on the copy button, switch to the do-file editor, paste the command, switch back to the dialog box and click on **Submit**.

You can now view the graphic.

8. Close the graph and return to the dialog box. We want to see the distribution by district. Click on the “**By**” tab. check the box next to **Draw subgraphs for unique values of variables**.
9. In the **Variables** box select **district**
10. Click on the copy button, switch to the do-file editor, paste the command, switch back to the dialog box and click on **Ok** button to view the graphic.

What are these graphs telling you?

Close the graph. Note you could have also added a title and other options as well.

Overlaid graphs

Graphs can also be overlaid.

1. Select **Graphics / Two-way graphs (scatterplot, lines, etc.)**.
You will see Plot 1 already defined
2. Click on the **Create** button to define a second plot. Click on the radio button next to **Fit plots**. Highlight **Linear prediction** in the box labeled Fit plots (select one).
3. For the **Y axis** select the variable **ae**. For the **X axis**, select the variable **cprod_tt**.
4. Click on the **Accept** button. You now see that **Plot 2** has been defined and is highlighted.
5. Click on the copy button, switch to the do-file editor,

paste the command, switch back to the dialog box and click on **Submit**. to view the graphic.

What are these graphs telling you?

6. Close the graph, Return to the dialog box, highlight **Plot 2** and click on **Edit** .
7. Change the type of plot to **quadratic prediction plot w/CI**. Click on the **Accept** button.
8. Click on the **Submit** button to view the graphic.
What are these graphs telling you?
9. If we want to see the distribution by district, click on the “**By**” tab. In the **Variables** box select **district**
10. Click on the **Ok** button to view the graphic.
What are these graphs telling you?

The Stata commands are:

```
twoway (scatter ae cprod_tt)
```

```
twoway (scatter ae cprod_tt), by(district)
```

```
twoway (scatter ae cprod_tt) (lfit cprod_tt)
```

```
twoway (scatter ae cprod_tt) (lfit ae cprod_tt),  
by(district)
```

```
twoway (scatter ae cprod_tt) (qfitci ae cprod_tt)
```

```
twoway (scatter ae cprod_tt) (qfitci ae cprod_tt),  
by(district)
```

Survey Estimation - Accounting for Design Effects

Stata provides statistical commands that have been developed specifically for survey analyses. The Stata User’s Guide discusses these commands as well as the manual called Survey Data. Most of these commands begin with the letters **svy**. There are a few of the survey commands that do not begin with these letters.

Survey data generally have three importance characteristics:

1. The weights applied to survey data are sampling weights - also called probability weights
2. The sample is clustered
3. Stratification is used in selecting the sample

If data meets any one of the above characteristics, the survey commands can be used for analysis. Briefly, sampling weights are used in analysis to give estimators that are approximately unbiased for whatever is being estimated for the whole population, i.e. one observation represents many elements in the population from which the sample is drawn.

Clustering by districts or villages is used in almost all survey sampling rather than selecting an independent sample. Further

sub-sampling may occur within a district or a village as well. Units at the first level of sampling are called the “*primary sampling unit*” or “PSU” or cluster.

To summarize, weights are used to obtain the correct point estimates. Clustering and stratification are used to get the correct standard errors.

The `svy` commands also calculate the design effects of `deff` and `deft`. *Deff is equal to the design-based variance estimate divided by an estimate of the variance that would have been obtained if the survey was carried out using simple random sampling. Deft is approximately equal to the square root of deff.* Further explanation of these two terms can be found in the Survey Data manual under the command **svymean**.

We will use a data set from Zambia from the Post harvest survey of the 2001/2002 agricultural season where the area planted for specific types of crops is tested.

1. Click on **File** then **Open**
2. Select **Zambia_PHS0102_crop_area.dta** and click on **Open**.
3. Paste the command into the do-file editor and delete the reference to the directory.

Use the browse command to look at the data or click on the browse icon.

browse

In Zambia for surveys conducted in the 1990s and early 2000, a stratified random sampling method was used. This method divided the districts into census supervisory areas (CSA). Within the CSA, Standard Enumerator Areas (SEA) were defined. The primary sampling unit (PSU) for this sample is the SEA. To identify each SEA as being unique the three variables - district, CSA and SEA, must be combined into one variable. District has 3 numbers, CSA has 3 numbers and SEA has 2 numbers. To create a new variable with these variables one must multiply the district variable by 100,000, add CSA multiplied by 100, and add SEA. The Stata command is:

```
gen float cluster1 = dist*100000 + CSA*100 + SEA
```

We want to change the format of this variable so that we can easily read it to verify the variable has been created correctly. Use the format command.

```
format cluster1 %9.0f
```

Clusters may further be sampled in groups which are called strata. The Zambia example uses province - district as the

strata. Strata are considered to be statistically independent and can be analyzed as such.

A weight has already been calculated for each household. The variable which contains this value is called **hhwgt**.

We need to compute the cluster variable. We can use **dist** for the strata variable since it already contains the province value as part of the district code.

Close the browser and use the **gen** command to create the variable “cluster1”.

To be able to use the survey commands, we must first define the stratified random sampling method that was used to account for weighting, clustering and stratification. We will use the **svyset** command to specify the method.

1. Click on **Statistics** then **Survey data analysis**
2. Then click on **Setup & utilities** then **Declare survey design for dataset**
3. In the **Primary sampling unit:** box select **cluster1**
4. In the **Strata:** box select **dist**
5. Click on the **Weights** tab
6. Click on the radio button next to **Sampling Weight Variable**.
7. Click on the drop-down arrow for the **Sampling weight variable:** box and select **hhwgt**
8. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on **Ok**.

The Stata command is:

```
svyset cluster1 [pweight=hhwgt], strata(dist)
vce(linearized) singleunit(missing)
```

After running the command we see a summary of the command in the Results window:

```
pweight: hhwgt
VCE: linearized
Single unit: missing
Strata 1: dist
SU 1: cluster1
FPC 1: <zero>
```

We can use the **syvdesc** command to look at the strata and PSU arrangement of the dataset.

1. Click on **Statistics** then **Survey data analysis**
2. Then click on **Setup & utilities** then **Describe survey data**

3. We can specific a variable or just run the command to look at the complete dataset. If we were interested to know which strata have only one sampling unit, we could put a tick next the box labeled “Display only the strata with a single sampling unit”
4. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on **Ok**.

Once the survey design has been specified and the file saved, it is not longer necessary to specify it again. The specification is saved with the data file.

We can use the **svytotal** command to look at the total estimates.

1. Click on **Statistics / Survey data analysis**
2. Then click on **Means, proportions, ratios, totals** then **Totals**
3. In the **Variables** box select **maisea ricea milleta sunfa**
4. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on **Submit**

```
. svy linearized : total maizea ricea milleta sunfa
(running total on estimation sample)

Survey: Total estimation

Number of strata =      69          Number of obs   =    6601
Number of PSUs  =     394          Population size =  807414
                                          Design df      =    325

-----
              |              Linearized
              |              Total  Std. Err.   [95% Conf. Interval]
-----+-----|-----
      maizea |    649230.9   25105.89   599840.3   698621.5
        ricea |    14472.95   2360.009   9830.125   19115.77
    milleta  |    61770.91   7346.125   47318.95   76222.87
        sunfa |    24319.15   3418.858   17593.26   31045.04
-----
```

Let's run the same analysis with only the weight specified to see the difference.

5. Click on the tab labeled **SE/Cluster** then click on the button labeled **Survey settings**.
6. Click on the button labeled **Clear settings**
7. Click on the **Weights** tab
8. Click on the radio button next to **Sampling Weight Variable**.
9. Click on the drop-down arrow for the **Sampling weight variable:** box and select **hhwgt**
10. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on **Ok**.
11. Click on the task **svy:total -...** on the Windows task bar.
12. Click on the copy button, switch to the **do-file editor**, paste the command, switch back to the dialog box and click on **Ok**

Note, we have gotten the same point estimate as the design-based estimate, but the standard errors are much smaller. The second table does not account for the sampling design.

```
. svyset _n [pweight=hhwgt], vce(linearized) singleunit(missing)

      pweight: hhwgt
          VCE: linearized
Single unit: missing
  Strata 1: <one>
      SU 1: <observations>
      FPC 1: <zero>

. svy linearized : total maizea ricea milleta sunfa
(running total on estimation sample)

Survey: Total estimation

Number of strata =          1          Number of obs   =       6601
Number of PSUs  =       6601          Population size =      807414
                                          Design df      =       6600

-----
          |                Linearized
          |                Total  Std. Err.  [95% Conf. Interval]
-----+-----
      maizea |      649230.9   14013.13   621760.6   676701.2
        ricea |      14472.95   1327.559   11870.5   17075.39
    milleta  |      61770.91   3942.684   54041.97   69499.84
        sunfa |      24319.15   1907.919   20579.01   28059.29
-----
```

Stata SAMPLE SESSION

Annexes**ANNEX I – Stata Commands**

This annex provides a brief reference guide and to explain the various functions of the Stata commands most commonly used. This annex was developed by Ellen Payongayong.

The commands in the table below do not contain the full Stata syntax.

Note that commands can be abbreviated. In the Help Syntax Viewer, the syntax explanation will show how much of the command must be typed, e.g. “Summarize” can be shortened to “su” or “sum”. In this Help viewer, the letters that are required for the command are underlined.

Command	Description
<code>pwd</code>	tells you which directory you're in
<code>cd {c d e...}</code> :	cd c: changes drives to c drive
<code>cd ..</code>	changes directory one level higher
<code>cd (path)</code>	changes current directory to that specified in path
<code>cd\</code>	takes you to the root directory
<code>dir</code>	lists contents of current directory
<code>use filename1</code>	loads file into memory
<code>save filename2</code>	saves current file in memory into <i>filename1</i> . if filename already exists, stata will not let you overwrite it
<code>save filename2, replace</code>	saves current file in memory into <i>filename2</i> , overwriting any file in working directory that is currently named <i>filename2</i>
<code>save, replace</code>	saves current file in memory into filename of that which is currently in memory
<code>edit</code>	brings up the data editor
<code>browse</code>	brings up the same data “editor” as in <code>edit</code> , but will not allow you to change data
<code>describe</code>	gives a description of the data file: number of observations, number of variables, list of variables, variable type and width, variable labels (if any)
<code>summarize</code>	gives basic summary statistics: number of valid observations, mean, standard deviation, minimum value and maximum value
<code>list</code>	lists observations
<code>keep</code>	retains in memory only those variables or cases specified
<code>drop</code>	discards from memory all variables or cases specified
<code>tabulate</code>	generates one- and two-way frequency tables
<code>tab1</code>	generates one-way table for each variable specified after the command.
<code>log using filename</code>	saves all commands and related output into specified file. the default format is SMCL for Stata Markup and Control Language. file is given extension .smcl
<code>log using filename, text</code>	saves all commands and related output into an ASCII file with extension .txt.
<code>log { off on close }</code>	off temporarily suspends the log file (switches it “off”); on

Command	Description
	switches the log “on” and close closes the log file
log using <i>filename</i> , append	adds subsequent commands to an existing log
log using <i>filename</i> , replace	saves all commands and related output into the specified file, overwriting said file if it already exists
<p>By opening a log file with cmdlog instead of log, you record only what you type in the command window (results are suppressed). The same basic syntax applies for both cmdlog and log. You can open both an <i>smcl</i> file and a log file.</p>	
clear all	clears data set from memory
help <i>command</i>	accesses help feature of Stata
exit	exits stata
sort <i>varlist</i>	sorts observations in ascending order according to the specified variable.
(1) note : “...” (2) note <i>varname</i> : “...” (3) notes	(1) allows you to enter notes about the dataset (2) allows you to enter notes about variable <i>varname</i> (3) calls up all notes in memory. Notes are saved in the dataset.
label variable <i>varname</i> “ <i>lblname1</i> ”	assigns a variable label to variable specified
(1) label define <i>lblname</i> # “ <i>label1</i> ” [# “ <i>label2</i> ”]... (2) label values <i>varname1</i> <i>lblname</i>	(1) assigns labels to integers (#) and stores these in the value label <i>lblname</i> (2) associates the value label <i>lblname</i> to the variable <i>varname1</i> e.g. label define gender 1 “female” 2 “male” label values sexhead gender
label list	lists all value labels
recode	modifies the value of a variable using rules specified
generate	creates a new variable
set memory	changes the amount of memory allocated to the data area; Stata suggests setting the memory to at least one and half times the size of the file you want to load in the memory of the computer.
replace	changes the value of an existing variable
count	when used with if , it counts the number of observations that meet the specified condition; otherwise, it counts the number of observations in the dataset
rename	changes the name of an existing variable
collapse	converts the data file in memory into another data set of means, medians, etc.
merge <i>varlist</i> using <i>filename</i>	merge joins corresponding observations from the dataset currently in memory (called the master dataset) with those from the Stata-format dataset stored as <i>filename</i> (called the using dataset) into single observations; performs a match merge on <i>varlist</i> when these are specified. the variable <code>_merge</code> , which gives information on the results of the merge command, is added to the file. <code>_merge==1</code> obs. from master data <code>_merge==2</code> obs. from using data <code>_merge==3</code> obs. from both master and using data
merge <i>varlist</i> using <i>filename</i> ,	“nokeep” causes merge to ignore observations in the using data

Command	Description
<code>nokeep</code>	that have no corresponding observation in the master.
<code>do</code>	executes a do-file
<code>assert</code>	assert verifies that an expression is true. if it is, the command produces no output; if it is not, assert informs you that the "assertion is false".
<code>append using</code>	append appends a STATA-format dataset stored on disk to the end of the dataset in memory.
<code>mvencode varlist, mv (#), [override]</code>	changes all occurrences of missing to # in the variable listing specified. override specifies the protection provided by mvencode is to be overridden. without this option, mvencode refuses to make the requested change if # is already used in the data.
<code>mvdecode varlist, mv (#)</code>	changes all occurrences of # to missing in the variable list
<code>egen</code>	creates a new variable equal to the specified functions and its arguments
<code>regress depvar varlist</code>	regress estimates a model of the dependent variable on variables in <i>varlist</i>
<code>xi: regressi.variable</code>	constructs categorical dummy variables for variables omitting the first category.
<code>predict variable</code>	stores the predicted values from the regression in <i>variable</i> . what this command can do is determined by the previous command.
<code>probit</code>	probit estimates maximum-likelihood probit models.
<code>search</code>	searches the keyword database. Use search when you are not certain of the command, e.g., search string shows all commands associated with strings.
<code>tables</code>	calculates and displays tables of statistics.
<code>reshape</code>	converts data from wide to long form and vice versa. 'wide' and 'long' refer to how data are organized. See reshape notes below.
<code>fillin varlist</code>	adds observations with missing data so that all combinations of <i>varlist</i> exist, thus rectangularizing the file. the variable <code>_fillin</code> is added to the data. <code>_fillin</code> is 1 for created observations and 0 for previously existing observations.
(<code>svy</code> commands)	these are commands prefixed with 'svy' and they pertain to commands used in analyzing survey data.
<code>tables</code>	calculates and displays tables of statistics.
<code>format varlist %fmt</code>	formats numeric variables as follows--number before the decimal indicates the length of the variable, number after the decimal indicates number of decimal places: %#.#g - general numeric format (%5.0g) %#.#f - fixed numeric format (e.g., %5.2f) %#.#e -base 10 power strings are formatted as follows and can be 81 chars long: %#s (e.g., %10s)

Reshape notes: The **reshape** command is particularly useful for files such as that shown in the following example:

Households were asked about the number of livestock owned for three types of livestock coded 330, 331 and 335. To save on data entry time, only those entries reporting any livestock were entered. Missing livestock codes in the file therefore means that the household did not own the livestock associated with the code. The file looks like this.

```

hh animcode  num
206  331    70
217  331    65
217  335     8
221  330  1200
221  331    200

```

The above file could have been organized such that each household has only one line of information, and the three animal types appear as three different variables. Such a file would be the wide form of the data. The file as it is organized now is the long form of the data.

The following reshape command converts the file from long to wide form such that each animal code is now a variable, and the file becomes a household-level file.

```

. reshape wide num, i (hh) j (animcode)
. list, nol nod noo

hh num330 num331 num335
206 . 70 .
217 . 65 8
221 1200 200 .

```

When followed by this next command, the file is re-converted from wide to long. But note that the file has become rectangularized, that is, the three animal codes now appear for each household.

```

. reshape long num, i (hh) j (animcode)
. list, nol nod noo

hh animcode  num
206  330     .
206  331    70
206  335     .
217  330     .
217  331    65
217  335     8
221  330  1200
221  331    200
221  335     .

```

The command **fillin** would have also generated the same rectangularized file as in the preceding example.

Do-file suggested commands to place at the beginning of a do-file to set the parameters before starting to work:

1. Commands in a do-file may be delimited by a carriage return or a semi-colon. To set the semi-colon as the delimiter, the command is:

```
#delimit ;
```

This command will only work in a do-file. The delimiter cannot be changed from the console.

If you wish to revert back to the carriage return as the delimiter, the command is:

```
#delimit cr
```

2. The next command will clear the memory:


```
clear all;
```
3. There are several “set” commands that are useful to put at the beginning of the do-file as well.


```
set memory 70000; (sets the size of memory)
set matsize 100 ; (limits number of variables that can be specified in an estimation command)
```

ANNEX II - Questionnaire

**Socio-Economic Survey of Family Sector Farms
in the Province of Namputa
(Angoche, Monapo e Ribaúe)**

July/August 1991

Departamento de Preços e Mercados
Food Security Project

Name of Household Head _____

Household Number _____ HH

Aldeia _____ VIL

Distrito _____ DIST

(Subset of questions from original questionnaire)

Filename: c-hh.dta

I. HOUSEHOLD CHARACTERISTICS

- H1** _____ 1. How many persons are in this household?
- H4** _____ 4. Has your family always lived in this village?
1=yes 2=no
- H8** _____ 8. Is your family registered as "deslocada"?
1=yes 2=no
- H19** _____ 19. Do you presently have lands in fallow?
1=yes 2=no
- H21** _____ 21. What is the total area of these fallowed parcels? (hectares)
- H24** _____ 24. Do you have lands that you have completely abandoned?
1=yes --> question 25 2=no --> question 27
- H25** _____ 25. What is the total area of these abandoned lands? (hectares)
- H26** _____ 26. What was the principal motive for abandoning these lands?
1=no security
2=lands lost fertility
3=lack of labor
4=insect attacks
5=other

[We would like to ask you about the food crops you grow.]

- H29** _____ 29. Over the last five years, have you increased or decreased the amount of land in food crops?
1=increased 2=decreased 3=no change
- H31** _____ 31. During a normal year, is your farm production sufficient to feed your entire family?
1=yes 2=no

[We would like to ask you about the cash crops you grow on your farm?]

- H34** _____ 34. Do you grow any crops that are principally destined for the market?
1=yes 2=no
35. Which crops are grow principally to be sold? (List the three most important)
- H35A** _____ 1=cotton 4=sunflower
H35B _____ 2=peanuts 5=rice
H35C _____ 3=sesame 6=other
- H36** _____ 36. Over the last five years, have you changed the area grown in these cash crops?
1=increased
2=decreased
3=no change
- H39** _____ 39. Do you normally grow cotton?
1=yes 2=no
- H52** _____ 52. Since your involvement with the cotton companies, have you reduced your area dedicated to food crops, such as maize and manioc?
1=yes 2=no

IV. PRODUCTION

- H56** _____ 56. Do you have cashew trees?
1=yes 2=no
- H57** _____ 57. How many trees do you presently have? (number)
- H57A** _____ 57A. Of these trees, from how many did you harvest during the last year? (number)

V. AGRICULTURAL SALES

We would like to ask about the marketing of your agricultural products since August of 1990.

64. Over the last five years, have you increased the quantities marketed of the following crops:

- H64A** _____ a. maize 1=yes 2=no
H64B _____ b. manioc 1=yes 2=no
H64C _____ c. rice 1=yes 2=no
H64D _____ d. cotton 1=yes 2=no
H64E _____ e. peanuts 1=yes 2=no
H64F _____ f. beans 1=yes 2=no
H64G _____ g. sorghum 1=yes 2=no
H64H _____ h. cashew nuts 1=yes 2=no

- H65** _____ 65. Compared with five years ago, has the marketing of these products been more difficult or easier?
1=more difficult --> question 66
2=easier --> question 67
- H66** _____ 66. If more difficult, why?
1=fewer buyers
2=transportation problems
3=security problems
4=low prices
5=lack of consumer goods
6=other

H67 _____ 67. If easier, why?
 1=more buyers
 2=better transportation
 3=better security
 4=attractive prices
 5=more consumer goods
 6=other

H83 _____ 83. Does your family usually receive traditional gifts or participate in exchange relations?
 1=yes 2=no

H84 _____ 84. If yes, how often?
 1=only when there is a lack of food
 2=only during feasts and rituals
 3=frequently

XI. **TYPICAL CONSUMPTION PATTERNS.**

H86 _____ 86. How many meals did these people have yesterday? (Number of meals)

H89 _____ 89. Do you consider these meals adequate to maintain the health of all the household members?
 1=yes 2=no

We would also like to ask you about your diet during the hungry period (January to May).

H91 _____ 91. How meals do you customarily prepare daily during hungry period?

H92 _____ 92. In general, are these hungry period meals adequate to maintain the health of all household members?
 1=yes 2=no

H96 _____ 96. During the hungry period, was there always food available to purchase from the market or from your neighbors?
 1=yes 2=no

I. HOUSEHOLD CHARACTERISTICS

Table IA: Household Characteristics

Filename: c-q1a.dta

Name	Family Member Number	This person works on-farm or off-farm 1=yes 2=no	Relation to Head 1=head 2=spouse 3=child 4=parent 5=other kin 6=other	Age	Sex 1=m 2=f	Level of Schooling (enter the last completed year) 0=illiterate 12=post-high school 98=no formal schooling but literate	Marital Status 1=monogamous 2=polygamous 3=single 4=widowed 5=divorced 6=emigrant wife (husband out longer than six months)
	MEM	CA1	CA2	CA3	CA4	CA5	CA6
	1		Head				
	2						
	3						
	4						
	5						
	6						
	7						
	8						
	9						
	10						
	11						

IV. PRODUCTION

Table IV: Characteristics of Production

Filename: c-q4.dta

Product		Quantity harvested		Quantity harvested in a normal year		Existing stocks at harvest time		Month in which last year's stock ran out	Amount to be stored from this year's harvest for consumption		How long will this year's stocks last?	Quantity reserved for seed	
		Unit	Qt	Unit	Qt	Unit	Qt	(enter the month)	Unit	Qt	(enter the month or "all year", if appropriate)	Unit	Qt
1=corn	8=peanuts	1=sack 100		1=sack 100		1=sack 100			1=sack 100			1=sack 100	
2=beans	9=cashew nuts	2=sack 50		2=sack 50		2=sack 50			2=sack 50			2=sack 50	
3=manteiga beans	10=cashew drink	3=kilo		3=kilo		3=kilo			3=kilo			3=kilo	
4=manioc	11=cane drink	4=liter		4=liter		4=liter			4=liter			4=liter	
5=rice	12=coconut	5=can 20		5=can 20		5=can 20			5=can 20			5=can 20	
6=sorghum	13=coconut drink												
7=cotton	others											other	
PROD		P1A	P1B	P2A	P2B	P3A	P3B	P4	P5A	P5B	P6	P7A	P7B

V. **AGRICULTURAL SALES**

Table V: Sales of Farm Products

Filename: c-q5.dta

Sale	Crop	Quantity sold		Period of sale	Motive for sale at this time	Buyer	Locale of sale	Distance from the farm	Why sold to this buyer	Value of Sales		Who in the household is responsible for the sale
		Units	No. of Unit							meticais	Unit	
	1=corn 2=manteiga bean 3=beans 4=manioc 5=rice 6=cotton 7=peanuts 8=cashew nut 9=cashew drink 10=cocos others	1=sack 100 2=sack 50 3=kilo 4=liter 5=can 20		1= planting (Aug-Dec.) 2= hungry period (Jan-April) 3=this year's harvest 4= various times	1=needed money 2=buyers available 3=consumer goods available 4=attractive price	1=lojista 2=wholesaler 3=AGRICOM 4=ambulante 5=brigada 6=company	1=farmgate/ house 2=village 3=locality 4=district 5=province	(enter the kms between farmer and point of sale)	1=the only one available 2=always sell to this one 3=best price 4=transportation provided 5=carries consumer goods		1=unit price 2=total value	1=husband 2=wife
VEN	PROD	V2A	V2B							V9A	V9B	
1												
2												
3												
4												
5												
6												
7												
8												
9												

N.B. Not all of the variables that appear in the printed table are in file c-q5.dta. Only variables **VEN**, **V2a**, **V2b**, **V9a** and **V9b** were kept for this exercise. The **PROD** variable replaces the **V1** variable.