Quantitative Data Analysis Using Stata

Awol Seid
Addis Ababa, Ethiopia

Contents

Chapter 1

Basic Statistical Concepts

1.1 Common Terms

- **Population**: A statistical population consists of *all objects* under study. The total number of objects in a population is called *population size* (N).
- **Sample**: is a subset of a population. The number of objects in a sample is also called *sample size* (n).
- **Datum**: It is an observed value representing one or more characteristics of an object. It is also known as an *observation* or an *item* or a *case* or a *unit*.
- **Data**: are a collection of observed values (observations or cases) of some objects.

1.2 Variables

A variable is a characteristic or an attribute that can assume different values. For example: height, family size, gender, marital status $\cdots$.

1.2.1 Types of Variables: Quantitative vs Qualitative

Based on the values that variables assume, variables can be classified as *quantitative* or *qualitative* (*categorical*).

- **Quantitative variables**: Quantitative variables are those variables which assume numeric values. These variables are numeric in nature. Height and family size are examples of quantitative variables.

  Quantitative variables are again further classified into two; *discrete* and *continuous* variables.

  - **Discrete variables**: Discrete variables are those variables that assume a countable number of distinct and recognizable whole number values. Family size, number of children in a family, number of cars at the traffic light, $\cdots$ are some examples of discrete variables. Discrete variables can assume a *finite* number of possible values or an *infinite countable* number of values. The values of these variables are obtained by counting (0, 1, 2, $\cdots$).

  - **Continuous variables**: Continuous variables can take any value including decimals. These variables theoretically assume an *infinite* number of possible values. Their values are obtained by measuring. Examples of continuous variables are height, weight, time, temperature, $\cdots$

- **Qualitative variables**: Qualitative variables are, on the other hand, those variables that assume non-numeric values called *categories* or *levels* or *groups*. For example, gender is a qualitative variable with two categories (levels): male and female. Marital status is also qualitative with, say, four categories: single, married, divorced, other.

  Based on the number of values that qualitative variables assume, they can be classified as *binary* (*dichotomous*) or *multinomial* (*polytomous*).

  - **Binary variables**: Binary variables often consist of '*either-or*' type responses. That is, these variables have only *two* categories (levels). For example, gender, exam result of a student (pass, fail), smoking (smoker, non-smoker)are binary variables.
  - **Multinomial variables**: Multinomial variables are those qualitative variables with *three or more* categories. For example, blood type (A, B, AB, O), marital status (single, married, divorced, other), religion (orthodox, muslim, protestant, $\cdots$), color (blue, red, green, black, $\cdots$) are multinomial variables.

Example 1.1. Classify each of the following variable as qualitative or quantitative and if it is quantitative classify as discrete or continuous.

1. Color of automobiles in a dealer's show room
2. Age
3. Number of seats in a movie theater
4. Temperature in the class room
5. Number of tomatoes on each plant on a field
6. Weight
7. Temperature (very cold, cold, hot, very hot).

1.2.2 Measurement Scales: Nominal, Ordinal, Interval and Ratio

Consider the following two cases.

Case 1:

- Mr A wears 5 when he plays foot ball.
- Mr B wears 6 when he plays foot ball.

Who plays better? What is the average t-shirt number?

Case 2:

- Mr A scored 5 in Stat quiz.
- Mr B scored 6 in Stat quiz.

Who did better? What is the average score?

Based on the number on the t-shirts, it is not possible to judge whether Mr B plays better. But, by using the test score, it is possible to judge that Mr B did better in the exam. Also it not possible to find the average t-shirt numbers because the numbers on the t-shirts are simply codes but it is possible to obtain the average test score.

In general, a scale of measurement shows the *amount of information* contained in the value of a variable, and what *mathematical operations* and *statistical analysis* are permissible to be done on the values of the variable. There are four levels of measurement. These levels, from the weakest to the strongest, in order are: *nominal*, *ordinal*, *interval* and *ratio*.

1. **Nominal variables**: Nominal variables are qualitative variables which show classification of individuals into *mutually exclusive* (*non-overlapping*) and *exhaustive* categories without any associated ranking. For example; gender, religion, ethnicity, eye color (black, brown, etc), $\cdots$ are nominal variables. Numbers may be assigned to the categories of these variables for coding purposes. But, it is not possible to compare individuals based on the numbers assigned to the categories. The only mathematical operation permissible on these variables is counting.

2. **Ordinal variables**: Ordinal variables are also qualitative variables whose values can be ordered and ranked. However, the ranks only indicate as to which category *greater* or *better* but there is *no precise difference* between the categories of the variable. Example: grade scores (A, B, C, D, F), academic qualifications (B.Sc., M.Sc., Ph.D.), strength (very weak, week, strong, very strong), health status (very sick, sick, cured), strength of opinions in likert scales (strongly agree, agree, neutral, disagree, strongly disagree)

3. **Interval variables**: Interval variables are quantitative variables and identify not only as to which category is greater or better but also *by how much*. It is the stronger form of measurement but, there is no true (absolute) zero. Zero indicates *low* than *empty*. Examples: temperature, $0\,^{\circ}\mathrm{C}$ does not mean there is no temperature but, rather, it is too cold. Similarly, if a student scores 0 in a certain course, it does not mean the student has no knowledge in the course at all.

4. **Ratio variables**: These scales are the highest form of measurements. Ratio variables are quantitative variables but, unlike the interval variables, zero shows absence of a characteristic. All mathematical operations are allowed to be operated on the values of these variables. Examples: height, weight, income, expenditure, consumption,$\cdots$.

Summary:

- All quantitative variables are either interval or ratio scales where as all qualitative variables are either nominal or ordinal scales.

- Most statistical analyses do not distinguish interval and ratio scale variables. As a result, in most practical aspects, they are grouped under metric (scale) variables.

- If a characteristic has only one value in a particular study it is not a variable; it is a constant.
  - Thus, marital status is not a variable if all participants are married.
  - Gender is not a variable if all participants in a study are female.

1.2.3 Role of Variables: Dependent vs Independent

Based on the role of variables in statistical analysis, variables can also be classified as *response* (*dependent*), *explanatory* (*independent*) or *extraneous* (*nuisance*) variables.

- A *dependent* variable is a variable, that is, of primary interest to be determined as an outcome. For example, the outcome of a certain treatment or the educational achievement level can be considered dependent variables.

- An *independent* variable, also called *covariate* (*factor*), is a variable to be used to determine the dependent variable. There are two types of independent variables: *attribute* (*measured*) and *active* (*manipulated*) variables.
  - An *attribute* independent variable is a variable whose values are *preexisting attributes* of objects under study. The values of such a variable cannot be systematically changed or manipulated. For example, education, sex, socio-economic status, $\cdots$.
  - An *active* independent variables can be experimentally manipulated. Such an active independent variable is a necessary (but not sufficient) condition to make *cause-and-effect* conclusions. For example, a researcher might investigate a new kind of therapy compared to the traditional treatment (the treatment group each person is assigned to). A second example could be a design to evaluate the effect of different fertilizers on crop yields. A third example might be to study the effect of a new teaching method, such as cooperative learning, on student performance. Studies with active independent variables are experimental studies.

  Even though a statistical analysis does not differentiate whether the independent variable is an attribute or active, there is a crucial difference in interpretation. For scientific researches in applied disciplines, the need to demonstrate that a given intervention or treatment causes change in behaviour or performance is extremely important. Only the approaches that have an active independent variable can provide data that allow one to infer that the independent variable caused the change or difference in the dependent variable. In contrast, a significant difference between or among persons with different values of an attribute independent variable should not lead one to conclude that the attribute independent variable caused the dependent variable to change.

- *Extraneous* variables are variables that are not of interest, but could influence the dependent variable, that need to be "controlled".

Based on the type and role of variables, the common statistical methods are shown in the following table.

Dependent Variable	Independent Variable	Method
Continuous	Binary	t test
Continuous	Multinomial	ANOVA
Continuous	Quantitative/Categorical/Both	Linear Regression
Categorical	Categorical	χ^2 test
Binary	Quantitative/Categorical/Both	Binary Logistic Regression
Multinomial	Quantitative/Categorical/Both	Multinomial Logistic Regression
Ordinal	Quantitative/Categorical/Both	Ordinal Logistic Regression
Discrete	Quantitative/Categorical/Both	Poisson Regression

Exercise 1.1. For each of the following objectives of a statistical investigation, identify the dependent and independent variables.

1. The effect of gender on academic performance.
2. The effect of age on passing a statistics course.
3. The effect of gender on passing a statistics course.
4. Evaluating the effect of different fertilizers on crop yields.
5. Comparing the academic performance of male and female students.
6. Studying the effect of a new teaching method, such as cooperative learning, on student performance.
7. The effect of age (classified as < 18, $18 - 24$, $25 - 34$, $35 - 64$ and > 65 years) on academic performance.

1.3 Coding Variables

Coding is the process of assigning numbers to the categories (levels) of a qualitative (categorical) variable. Below are some rules of coding variables.

1. All codes (data values) should be numeric. Even though, it is possible to use letters or words as codes, it is not desirable to do with most statistical software packages. For example, the variable gender may be coded as 1 for males and 2 for females (or 0 for males and 1 as females).

Variable	**Value**	**Value Label**
Gender	1	Male
	2	Female

If occupation has four categories, it can be coded as

Variable	**Value**	**Value Label**
Occupation	0	Government Employee
	1	NGO Employee
	2	Private Employee
	3	Other

Since, the categories of a nominal variable cannot be ordered, the starting category for coding is arbitrary.

2. It is better to use higher numbers (codes or values) for the higher ("agree", "good" or "positive" end) categories of an ordinal variable. For example,

Variable	**Value**	**Value Label**
Likert	1	Strongly Disagree
	2	Disagree
	3	Neutral
	4	Agree
	5	Strongly Agree

Sometimes we might see questionnaires that use 1 for "strongly agree" and 5 for "strongly disagree". This is not wrong as long as we are clear and consistent. For example, if education level has, say, 4 categories: no education, primary education (grade 1-8), secondary education (grade 9-12) and tertiary education (college or university), it can be coded as:

Variable	**Value**	**Value Label**
Education	0	Tertiary Education
	1	Secondary Education
	2	Primary Education
	4	No Education

Here, we have to remember an increase in the codes indicates a decrease in the education level. However, we are less likely to get confused when interpreting the results if high values have positive meaning.

3. All codes for a variable must be mutually exclusive. Two or more categories (levels) of a variable should not have the same code.

Chapter 2

Introducing Stata

This document is mainly aimed to help new users to get started with Stata statistical software. In order to comply with this the goal of the document, it starts with a general introduction to the software package and introduce then chapter by chapter more complicated notions. The reader is supposed to understand the statistical analysis and models presented here and know how and when to use them. The document will never achieve a final version since it is under constant review, and subject to changes and extensions. Comments and suggestions are always welcome.

2.1 Overview of Stata

Stata is an integrated statistical analysis package designed for research professionals. It has got started in California in the mid-1980s and it was written in the C programming language. At one time, the name S was considered, presently the name is changed to **Stata** (**Sta**tistics Da**ta**). Stata's main strengths are handling and manipulating large datasets.

There are 4 different packages available: StataMP (Multi-Processor) which is the most powerful, StataSE (Special Edition), StataIC (Inter-Cooled) and SmallStata. The main difference between these packages is the maximum number of variables and observations that can be handled. The one that we will use in this training is StataMP, version 13.

Stata is a command-driven package. Although it has pull-down menus from which different commands can be chosen, the best way to learn Stata is still by typing in the commands. This has the advantage of making the switch to programming much easier which will be necessary for any serious analytical work.

2.2 Opening Stata

The first thing to do is to get Stata itself going. On PCs, you can go to the Windows **Start** Menu → **All Programs** → **StataMP** and click on the Stata icon there. When Stata is opened, there will be five windows in the main interface as shown in figure 2.1.

1. *Variables* Window: This window displays a list of all variables (variable names and labels) in the dataset.

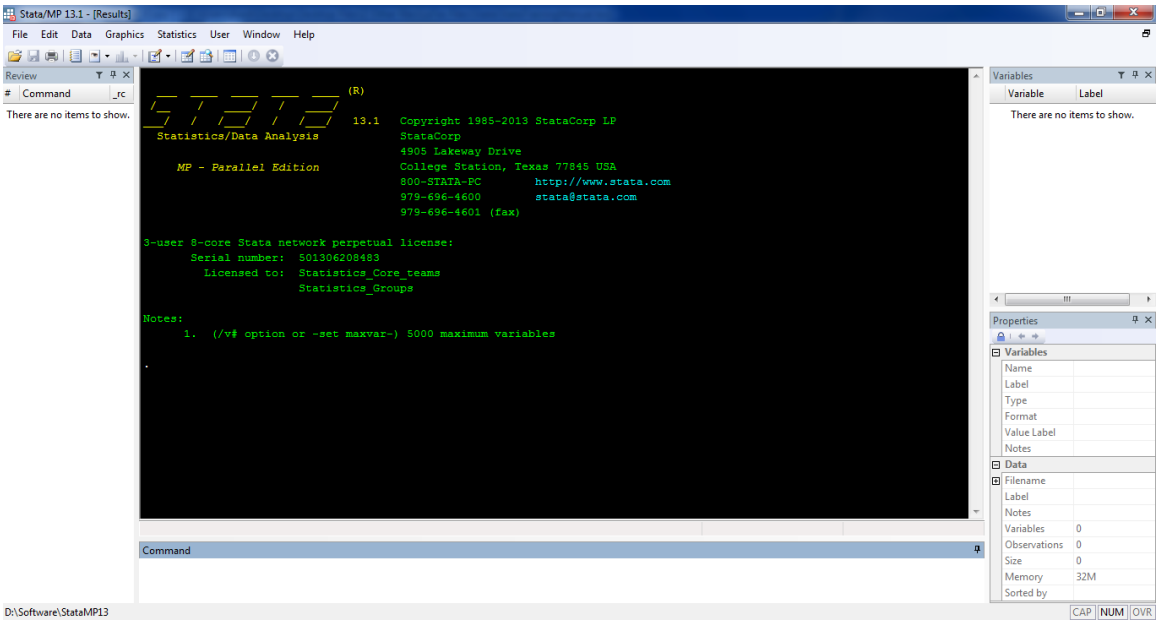


Figure 2.1: The five windows in the main interface of Stata

2. Variable *Properties* Window: This window displays the properties (name, label, type, format, $\cdots$) of each variable in the dataset. By clicking on one of variable names in the *Variables* window, its properties are shown in Variable *Properties* window.

3. *Command* Window: This window is the place where the commands are to be written. Also, clicking on any variable in the variables window will copy its name in the *Command* window. When pressing the *Enter* key, Stata immediately executes the command.

4. *Results* Window: This is the window where executed commands together with their outputs are to be displayed. The commands are shown in the *Results* window preceded by a period (.).

5. *Review* Window: This window displays a list of all commands we have used in the order we used them. Clicking on any command in this window will copy it to the *Command* window. This is especially helpful, when mistyping a command, to edit it and run again. Double-clicking on a command in this window will automatically executed.

A few points of emphasis:

1. The font style and size can be changed by right-clicking in any window and selecting **Font**. The default color schemes in the *Results* window can also be changed by right-clicking in it and selecting **Preferences**, and then choosing a different color scheme.

2. In case a window ever disappears, just click on **Window** tool bar and click on the missing window to make it reappear. Also we can stretch any window just as we would resize the window.

In addition to these windows, there are 4 additional windows: *Data editor*, *Do-file editor*, *Graphics* window and *Help viewer*.

2.3 Stata Language Syntax

Written commands must comply with Stata's grammatical rules. The basic structure of a Stata command is:

`command` Variable1 Variable2 Variable3 `,options`

For the most part, Stata will provide error messages when we type something wrong. The first thing to check is capitalization since variable names and commands are case-sensitive. All Stata commands are lower case. A variable name can be up to 32 characters long but it must start with a letter, and can contain letters and numbers. Spaces are not allowed; an underscore (_) can be used instead. Comments can be added preceding by an asterisk (*) because any input preceded by an asterisk will not be executed by Stata.

The second thing to check for errors is punctuation. Stata commands are modified by qualifiers preceding options; there must be a comma between the last qualifier and the first option. Also most command prefixes must be separated from the subsequent command by a colon. Also, filenames having spaces and string values, in commands, must be enclosed in double quotes (").

Some of the Stata commands can be abbreviated. In this manual, you might be wondering why some (one or more) of the letters of Stata commands and options are underlined but nothing else, it is to indicate the minimum acceptable abbreviation to be typed instead of typing all letters of the command or option. Therefore, the third thing to check is Stata's abbreviated commands and options.

One of the main feature of Stata is that it has now a Statistics Menu. When we use the menus (point-and-click), the command is generated for us, and appears in the *Review* and *Results* windows, just as if we had typed it. But, we will not go in detail on how to use menu as it is encouraged to learn typing Stata commands so as to take full advantage of Stata's capabilities.

Conventions

In this document, the following conventions are used: a `typewriter` font is used to represent the actual Stata commands and output that you see in the screen. Also, a sans serif font is used for items such as variable names, filenames which should be replaced with particular instances.

Chapter 3

Getting Data into Stata

3.1 The Data Editor Window

The Stata data editor window consists of rows and columns in a spreadsheet-like arrangement which can be used to enter new data, or to view or edit existing data. The columns represent variables and the rows represent cases (observations). To open the Data Editor, click on **Windows** → **Data Editor**. If there is a dataset in memory, it is displayed in the editor. Otherwise, we get a blank editor screen, like as figure 3.1.

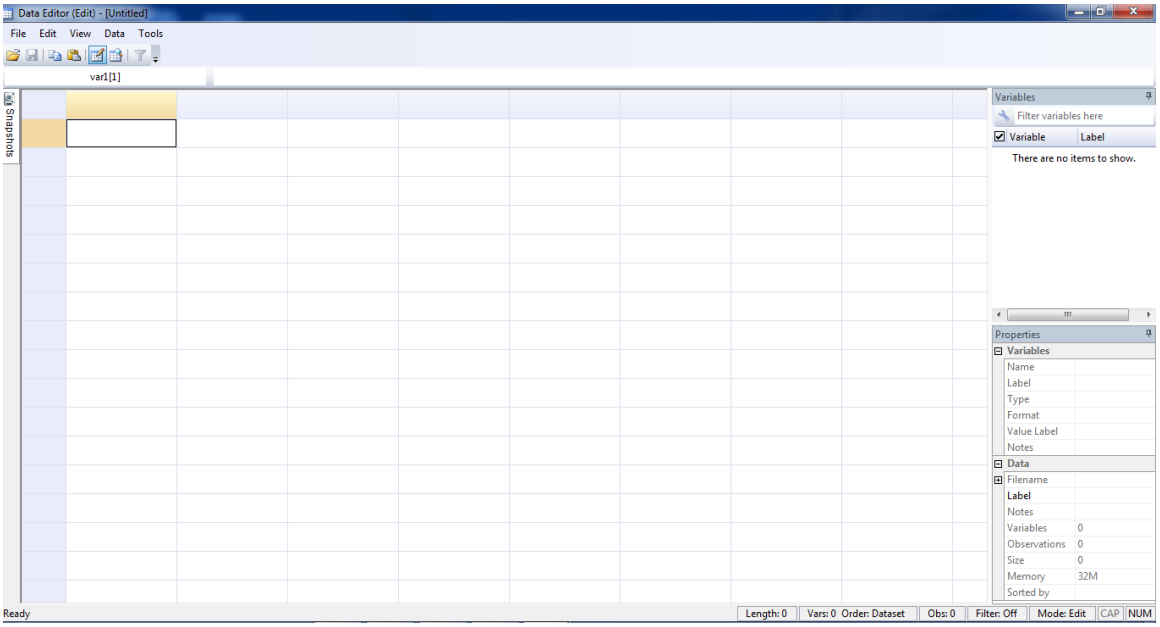


Figure 3.1: The data editor window of Stata

There are three basic variable formats in Stata: string, numeric and date. Strings are alphanumeric. They may consist solely of numerals but if a variable is declared as a string, mathematical operations cannot be performed on it.

In the data editor, the default color for strings variables is red. The default color for numeric data is black as are dates. Numerically coded variables with labels, such as "male" and "female", but the underlying value is really a number, say 1 and 2, are blue.

3.1.1 Data Entry and Coding

For illustration, let's enter the following data containing three variables (Weight, Sex and Marital Status (1=Single, 2=Married, 3=Divorced, 4=Other)) with five observations.

Weight	Sex	Marital Status
60.0	M	1
58.5	F	2
53.0	F	3
56.5	M	2
70.0	M	4

As we type 60 in the first cell of the data editor, it is displayed next to `var1[1]=60` that stands for variable 1 and observation 1, and corresponds to the highlighted cell. When we press Enter, the data we typed is entered into the highlighted cell. The display changes to `var1[2]`, and the corresponding cell is highlighted. After entering all values, the data editor will now look like as figure 3.2.

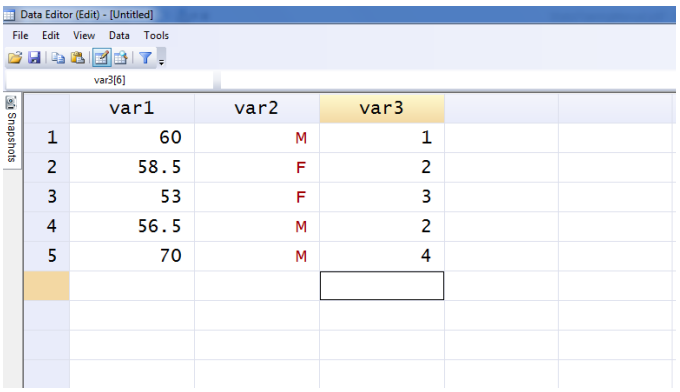


Figure 3.2: Entering data in the data editor

Stata assigned the default variable names `var1`, `var2`, `var3`. To change the variable names, click on any cell or the column heading of that variable and then, on the properties window **Name** field, remove the existing name and write the your preferred variable name. For example, to change `var1` to, say, Wei, click on any cell of the `var1` column. Next, on the properties window **Name** field, remove `var1` and write Wei, figure 3.3. Then when the **Enter** key is pressed, the name will be automatically changed. The **Label** field of the properties window is also used to write a longer description of that variable. For example, by clicking on any cell of the Wei variable, the label Weight of a student in kgs can be written in the **Label** field and press the **Enter** key.

The variable names `var2`, `var3` can be changed to Sex, Marital, respectively, in a similar fashion.

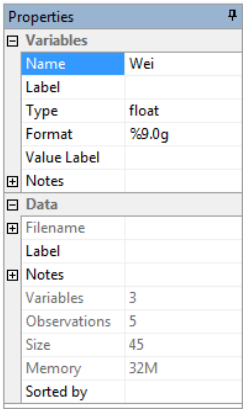


Figure 3.3: Changing variable names

Now save this data by clicking on **File** → **Save As** with a file name, say, Student. By default, Stata adds the extension `.dta` to indicate that it is in a Stata data file format.

3.1.2 Creating Value Labels

It is nice to have the values of a categorical variable labeled with their meaning. For example, Marital Status should have its categories as Single, Married, Divorced and Other rather than displayed as 1, 2, 3, and 4. Creating value labels in Stata has two components: creating a label that associates text with the codes and then assigning the label to one or more variables.

Let's create a value label for Marital Status. First, a label that associates a text with the codes should be created. In the variable *Properties* window, click on the **Value Label** field, figure 3.4.

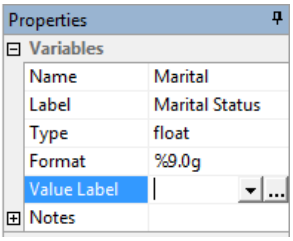


Figure 3.4: Creating a value label

Then, click on the ··· tab which is next to the dropdown tab. This opens the **Manage Value Labels** dialogue box shown in figure 3.5. Next click on the **Create Label** button which opens **Create Label** dialogue box shown in the same figure. Then, write the label name Mar (which can be the same to the variable name) in the **Label name:** field. Next,

- type 1 in the **Value:** field.
- write Single in the **Label:** field.

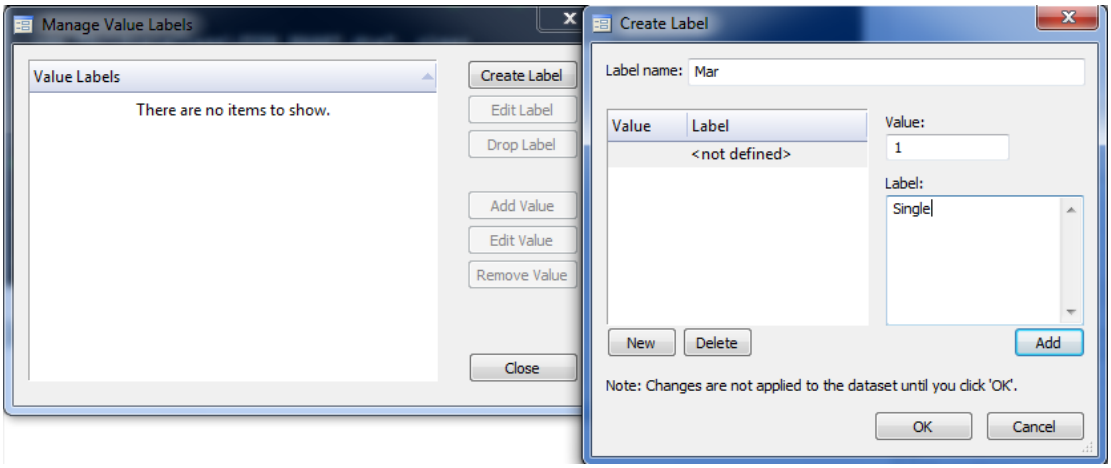


Figure 3.5: **Manage Value Labels** and **Create Label** dialogue boxes

Click on the **Add** tab to have this label added to the list and continue adding until all the values are labeled. Then, click on **OK** and finally click on **Close**.

Now we've completed creating a label Mar which contains 1=Single, 2=Married, 3=Divorced and 4=Other. But, this label is not attached with the variable Marital yet. Hence, to associate this label with the variable, click on any cell of the variable Marital and then click on the dropdown tab in front of the **Value Label** field of the variable *Properties* window. Select Mar, figure 3.6, to attach these value labels to Marital variable. Save the data and exit Stata.

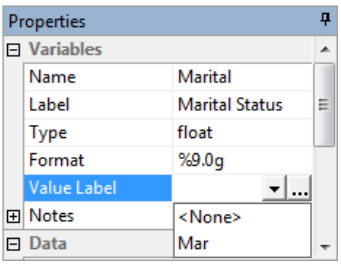


Figure 3.6: Attaching value labels

Exercise 3.1. Enter the data in following table in the data editor window and give the value labels: Height in meter, Blood Type (0=Type A, 1=Type B, 2=Type AB, 3=Type O), Sex (0=Male, 1=Female).

Height	Blood	Sex
1.55	1	0
1.6	0	1
1.72	1	0
1.5	2	1
1.85	3	0

3.2 Accessing Different Data Files

Stata can import data from and export data to different sources. Some of the data files are: Stata data files which have an extension `.dta`, Microsoft Excel data files which have an extension `.xls` or `.xlsx`, Text files which have an extension `.txt` or `.csv`, SPSS data files which have a `.sav` file extension, and other data files.

We can only work on one Stata dataset at a time. The data we are working on is stored in memory.

3.2.1 Changing the Working Directory: The `cd` Command

The working directory displayed at the bottom left hand corner of the Stata window is the default directory (to view the default directory type the command `pwd` in the command window and press enter). Therefore, Stata will save (open) all files to (from) this directory unless specified. Thus, before trying to access any dataset, it is better to create a directory (folder) in which all files will be kept. This helps to avoid writing the whole path of the file every time.

Let's create a new folder named StataClass on our preferred location. To change the working directory to this newly created directory, the command `cd` (change directory) followed by the directory name StataClass should be written in the *Command* window. That is, write

```
. cd D:\StataClass
```

in the command line and press the **Enter** key. Or click on the **File** menu and then **Change Working Directory** option. Then, select your preferred working directory folder.

To create a new directory within the current one (here, D:\StataClass), the command `mkdir` followed by the the directory name can be used. For example,

```
. mkdir FirstSession
```

creates a folder named FirstSession under the D:\StataClass directory.

3.2.2 Stata Data Files: The use and save Commands

Opening a Stata Data File: The use Command

The use command is used to open Stata data files, that is, data files having an extension `.dta`. To open such file, the command is followed by the file name.

```
. use FileName
```

But, a file cannot be opened when another file is already open. The file must be closed first. To do so, the `clear` command can be typed in the command line or it can be used as an option in the use command as below.

```
. use FileName ,clear
```

The `clear` command deletes all cases, variables, and labels from the memory. But, it does not delete any data saved to the hard-drive.

Let's open the Student.dta data which we saved previously. First we've to be sure that the data is in the working directory. Then, just write the following in the command window.

```
. use Student
```

or by specifying the full path location of the data, we can do as follows.

```
. use D:\StataClass\Student
```

When the data is successfully read, the variables of the dataset will be shown in the variables window.

Notes:

- If we do not include an extension, Stata assumes it is `.dta` .
- If we do not include a path, Stata assumes it is in the current directory folder.
- If the path name has spaces, we must enclose in double quotes: `use` "Food Expenditure".

Saving a Stata Data File: The save Command

One of the most important Stata commands to save data is the `save` command. The data in memory can be saved by typing the command `save` followed the name of the data file. You do not need to specify the `.dta` extension to filename, Stata will add it by default.

To save a data file:

```
. save FileName
```

Let's try to save the data we have opened using the filename Student.

```
. save Student
```

But wait, it did not work. Something like this popped up:

```
file Student already exists
```

Nuts! This is something to keep in mind-unless we explicitly specify it, Stata will not write to any files with the same name as a file we already have. If we want to replace the existing data file, the option `replace`, that forces Stata to overwrite the file, should be added in the `save` command. That is,

```
. save Student, replace
```

As a result, we get a satisfying:

```
file Student saved
```

3.2.3 Excel Data Files: The `import excel` and `export excel` Commands

Importing Data from an Excel File: The `import excel` Command

The `import excel` command is used to open an excel (`.xls`, `.xlsx`) data file. If the first row of the excel file contains variable names, do not forget to add the option `firstrow`.

```
. import excel FileName ,firstrow
```

In the folder given to you, there is an excel data file named CD4.xlsx. Let's import this data into Stata.

```
. import excel D:\StataClass\CD4 ,firstrow
```

Exporting Data to an Excel File: The `export excel` Command

Similarly, the `export excel` command is used to save data in an excel (`.xls`, `.xlsx`) file. If we need the first row of the excel file to contain the variable names, the option `firstrow(variables)` should be added.

```
. export excel FileName ,firstrow(variables)
```

Notes: There are some ground-rules to be followed when saving a `.xlsx`, `.csv` or `.txt` file for reading into Stata:

- Any extra lines below the data or to the right of the data (e.g. footnotes) will also be read in by Stata, so make sure that only the data itself is in the spreadsheet.
- The names of the variables should valid Stata variable names. If the file is laid out with years (e.g. 1980, 1985, 1990, 1995) on the top line, then Stata will run into problems. In such instances you can for example, place an underscore in front of each number (e.g. 1980 becomes _1980 and so on).
- Some notations for missing values can confuse Stata, e.g., it will read double dots (..) or hyphens (−) as text. Use find and replace to replace such symbols with single dots (.) or simply to delete them altogether.

3.2.4 The Menu Options of Opening and Saving Data

In addition to the commands, there are the menu options to import (export) data into (from) Stata.

1. The **Open** (`Ctrl+O`) option in the **File** menu can be easily used to open only a Stata data file (`.dta`).
2. The **Save** (`Ctrl+S`) and **Save As** (`Ctrl+Shift+S`) can be used to replace the existing file or to save in a different name in Stata data file (`.dta`) only.
3. The **Import** option under the **File** menu can be used to import excel data files (`.xls`, `.xlsx`), text data files (`.txt`, `.csv`) and other files.
4. The **Export** option under the **File** menu can also be used to export data to excel or text files.

3.2.5 SPSS Data Files: The savespss Command

The savespss command is used to export data in SPSS data file format. The structure of the command is

```
. savespss "FileName.sav"
```

Note here, the double quotes and the extension .sav should be written always. Otherwise, it does not work. Even if it works, the file cannot be opened.

Exercise 3.2. In the folder given to you, there are two datasets in excel file named JUSH_HAART and CD4Count. Both datasets were obtained from Jimma University Specialized Hospital - HIV/AIDS Outpatient Clinic, Ethiopia. The JUSH_HAART contains the baseline characteristics of 1464 HIV/AIDS patients who were 18 years old or older and who were under HAART treatment between 2007 and 2011 in Jimma University Specialized Hospital. Where as the CD4Count data contains the number of CD4 counts (per mm^3 of blood) of the same patients. The CD4 counts were measured approximately every 6 months; at the study entry, and again at the 6-, 12-, 18-, 24-, 30-, 36-, 42-, 48- and 54-month visits.

Import the JUSH_HAART data to Stata. Then, give the variables definitions in the table below and save it by giving a similar file name as the excel file.

Table 3.1: Variable descriptions of the JUSH_HAART data

Variable Name	**Variable Label**	**Value Label**
CardNum	Patient's Card Number	
Age	Age in Years	
Sex	Sex	
Wei	Weight in Kilograms	
MarStat	Marital Status	0=Never Married, 1=Married, 2=Divorced, 3=Separated, 4=Widowed
EducLev	Education Level	0=No Education, 1=Primary, 2=Secondary, 3=Tertiary
Emp	Employment Condition	0=Full-time, 1=Part-time, 2=Not Working, 3=Unemployed
ClinStag	Clinical Stage	1=Stage I, 2=Stage II, 3=Stage III, 4=Stage IV
FunStat	Functional Status	0=Working, 1=Ambulatory, 2=Bedridden
CD4	Number of CD4 Counts	
Status	Survival Outcome	0=Active, 1=Dead, 2=Transferred, 3=Lost-to-follow
Defaulter	Dropped Out Patient	0=Active, 1=Defaulted
SurvTime	Survival Time (Months)	

Chapter 4

Basic Data Management

After creating a new or opening an existing data file, it typically necessary to examine the data to identify possible problems. We need to answer questions like how much are missing? Which variables have missing data? Any variable has value(s) which seem illogical or implausible? For example an age of 150. Assessing internal consistency between variables, say, pregnancy and gender.

A number of commands are available for looking at the data directly, but the common ones are the `browse`, `edit`, `list` and `codebook` commands.

4.1 Viewing Data: The `browse` and `edit` Commands

The `browse` command is used to look at the data editor without the risk of changing any observation. Now type

```
. browse
```

and see what happens. This will open up the data browser which allows us to move around in, but not alter, the relevant data. But, if for some reason we want to alter the data, the `edit` command can be used.

```
. edit
```

If we want to alter anything, click on the cell we are looking to alter, type in the new value (just as we would in Excel). If we wish to make the changes permanent for future Stata sessions, we must save the latest data to a file. But, be careful, once saved, it is not possible to make like undo.

If there are a large number of variables in the data, we may want to look at only some of the variables. To do so, we can just type the commands followed by the variables we want to see.

```
. browse Age Sex
. edit Age Sex
```

Note also that when both `browse` and `edit` commands are executed, by default the labels of the variables (if coded) are shown in the data editor. But, if we want the values instead of the labels, we can add an option `nolabel` in both commands. That is,

```
. browse ,nolabel
. edit ,nolabel
```

4.2 Describing Data in Memory: The describe Command

The describe command gives some basic information of data: the number of observations (obs), number of variables (vars), and each variable's name variable name and label variable label.

To describe the JUSH_HAART data, just type:

```
. describe
```

This command delivers the following output in the Results window.

```
Contains data from D:\StataClass\JUSH_HAART.dta
  obs:         1,464
 vars:            13                          24 Nov 2016 20:48
 size:       142,008
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
CardNum         double  %10.0g                Patient's Card Number
Age             double  %10.0g                Age in Years
Sex             str1    %1s                   Sex of the Patient
Wei             double  %10.0g                Weight in Kilograms
MarStat         double  %13.0g     MarStat    Marital Status
EducLev         double  %10.0g     EducLev    Education Level
Emp             double  %11.0g     Emp        Employment Condition
ClinStag        double  %10.0g     ClinStag   Clinical Stage
FunStat         double  %10.0g     FunStat    Functional Status
CD4             double  %10.0g                Number of CD4 Counts
Status          double  %10.0g     Status     Survival Outcome
Defaulter       double  %10.0g     Default    Dropped Out Patient
SurvTime        double  %10.0g                Survival Time in Months
-------------------------------------------------------------------------------
Sorted by:  CardNum
```

Let's explain some of the results.

1. variable label is longer name associated with each variable. For example, the variable SurvTime has a label Survival Time in Months and the variable CD4 has a label Number of CD4 Counts. Whenever possible, variable labels should include the unit.

2. value label is a name attached to each value of a categorical variable. For example, if the variable MarStat has four values, each value is associated with a name. The value label for MarStat=0 could be Never Married, MarStat=1 could be Married, and so on.

3. `storage type` tells whether a variable is numeric or string (`str`). Stata stores data in either of two ways: numeric or string. Numeric will store numbers while string will store text (it can also be used to store numbers, but numerical analysis can not be performed on those numbers). The type of string is represented by the prefix `str` followed by the number of characters (maximum of 244 characters). So, if there is a variable value that appears in the data as biostatistics, it is a `str13` variable. In the above case, Sex is a 1-character string. Everything else is numeric.

   The exact storage type for each numeric variable depends on its value (not a terribly important point). Stata, like any computer program, stores numbers in binary format using 1s and 0s. Binary numbers tend to take up a lot of space, so Stata will try to store the data in a more compact format. The different storage types are:

   - `byte`: Integers between -127 and 100.
   - `int`: Any number without a decimal point falling between the limits between -32767 and 32740 is a valid integer. The following are not valid integers: -1,000 (commas not allowed), 987. (contains a decimal point).
   - `long`: Integers between -2147483647 and 2147483620.
   - `float`: Floating-point real number is a real number with about 8 digits of accuracy. Examples of illegal real constant is -10 (no decimal point, integer).
   - `double`: Double precision numbers are real numbers with about 16 digits of accuracy.

4. `display format` tells how the values of a numeric variable are displayed in the data editor. There are mainly two types of display formats in Stata: the general (`g`) and the fixed (`f`) format. The general format depends on the number while the fixed format has a fixed number of decimals no matter what the number is. The percentage sign (`%`) is used to declare formats. For example, `%10.0g` means the variable will be displayed with up to 10 digits, and Stata will decide whether and how many decimal places to display. This works well in most cases, but at times we may need to force Stata to display decimal places. We can do this in the **Format** field of the variable *Properties* window. This formatting always begins with the percent sign (`%`) and ends with a letter (`g` or `f`). For example, to force Stata to display the weight variable using 2 digits with 1 decimal place, we can change the **Format** field of the Wei variable to `2.1f` and observe the difference by browsing the data.

   Generally, the `storage type` refers to the way the information is stored in the hard-drive and `display format` is the way the information is displayed in the data editor.

4.3 Compressing Data in Memory: The `compress` Command

How to choose the best storage type? Choosing the best storage type is a relative technical information. In practice, the command `compress` analyzes every variable and converts it to the minimum (best fitting) storage format without making any change that would cause Stata to lose data. This command avoids wasting memory for nothing. The command has no options or arguments.

```
. compress
```

Then, Stata says:

```
CardNum was double now int
Age was double now byte
MarStat was double now byte
EducLev was double now byte
Emp was double now byte
ClinStag was double now byte
FunStat was double now byte
CD4 was double now int
Status was double now byte
Defaulter was double now byte
(99,552 bytes saved)
```

Now we can observe the difference by using the describe command.

4.4 Listing Values of the Variables: The list Command

The list command is used to look at each individual observation within the dataset. Now try typing:

```
. list
```

The first two observations are:

```
     +-------------------------------------------------------------------------+
  1. | CardNum | Age | Sex |  Wei |       MarStat |    EducLev |         Emp |
     |    1202 |  40 |   F |   43 |     Separated |    Primary |  Unemployed |
     |-------------------------------------------------------------------------|
     |  ClinStag |    FunStat |  CD4 |     Status  | Defaulter  |  SurvTime  |
     |  Stage IV |    Working |  365 |     Active  |    Active  | 26.966667  |
     +-------------------------------------------------------------------------+

     +-------------------------------------------------------------------------+
  2. | CardNum | Age | Sex |  Wei |       MarStat |    EducLev |         Emp |
     |    1203 |  50 |   M |   58 |       Married |    Primary |   Part-time |
     |-------------------------------------------------------------------------|
     |  ClinStag |    FunStat |  CD4 |     Status  | Defaulter  |  SurvTime  |
     |  Stage II |  Bedridden |  434 |     Active  |    Active  | 33.333333  |
     +-------------------------------------------------------------------------+
```

Clearly, there is a lot of data there. Notice that all observations are not listed from the list command. By default, observations are listed a screen full at a time. The key here is that if the output from a Stata command does not fit on the Results window, Stata displays as much as fits on the screen. Such output can be controlled by switching the set more command on/off. In such case, type in:

```
. set more on
```

This merely tells Stata to pause at each screen and wait for user input before moving further along. Now type `list` again and see what happens.

```
. list
```

Stata displays as much as fits on the *Results* window, and pauses with the message `--more--` in the lower-left hand corner of the screen to mean there is more to see. You need to hit either **Enter** key to advance line-by-line or the **Space Bar** or **Esc** key to advance one screen at a time. Also by clicking `--more--`, the next full screen output will be displayed. When we get Stata running a seemingly never ending command because of long output, to stop the output we can just click on the break icon, which looks like a red circle with a white cross through it (), located in the toolbar under the Window menu.

Instead of listing all variables, let's say we are only interested in taking a look at the Age and Sex variables per observation. Thus, the `list` command can be restricted by specifying these variables as:

```
. list Age Sex
```

The first 5 observations of the output of this command is:

```
     +-----------+
     | Age   Sex |
     |-----------|
  1. |  40     F |
  2. |  50     M |
  3. |  35     F |
  4. |  45     M |
  5. |  44     M |
     |-----------|
```

4.5 Describing Data Contents: The `codebook` Command

Next, the most detailed look at a variable is available via the `codebook` command. The `codebook` command delivers frequencies for categorical variables and some descriptive statistics (mean, standard deviation and some percentiles) for quantitative variables. Thus, this command gives much information including the number of missing values. This is important to know early in a project as it could have a huge impact on the analysis.

Let's try to look at the contents of Age in the JUSH_HAART dataset.

```
. codebook Age

-------------------------------------------------------------------------------
Age                                                             Age in Years
-------------------------------------------------------------------------------

                  type:  numeric (byte)
```

```
            range:  [18,85]                     units:  1
     unique values:  52                       missing .:  0/1464

             mean:   34.0116
          std. dev:   9.16026

      percentiles:        10%       25%       50%       75%       90%
                           25        28        32        39        45
```

That's quite a detailed readout. The minimum and maximum age of the patients are 18 and 85 years, respectively. The mean age of the patients is 34.01 years with a standard deviation of 9.16 years. Also, 10% of the patients were $\leq$ 25 years, 25% of the patients were $\leq$ 28 years, 50% of the patients were $\leq$ 32 years, 75% of the patients were $\leq$ 39 years and 90% of them were $\leq$ 45 years.

Again, let's try to look at the data contents of Sex.

```
. codebook Sex
```

```
-------------------------------------------------------------------------------
Sex                                                                         Sex
-------------------------------------------------------------------------------

            type:  string (str1)

     unique values:  2                        missing "":  0/1464

        tabulation:  Freq.  Value
                       930  "F"
                       534  "M"
```

Of the total 1464 patients, 930 of them were females while the remaining 534 of them were males.

Similarly, if we want to describe the details of Wei, we do the same as before.

```
. codebook Wei
```

```
-------------------------------------------------------------------------------
Wei                                                       Weight in Kilograms
-------------------------------------------------------------------------------

            type:  numeric (double)

            range:  [16,96]                     units:  .1
     unique values:  103                      missing .:  4/1464

             mean:   51.8679
```

```
          std. dev:   10.1934

      percentiles:        10%       25%       50%       75%       90%
                           40        45        51     57.75        65
```

As can be seen from the above output, the minimum and maximum weights are 16 and 96 kilograms, respectively. The mean weight of the patients is 51.8679 with a standard deviation of 10.1934 kilograms. Also, 10% of the patients weights 40 kilograms or less, 25% of the patients weights 45 kilograms or less, 50% of the patients weights 51 kilograms or less, 75% of the patients weights 57.75 kilograms or less, and 90% of them weights 65 kilograms or less.

Note that there are 4 missing values in the weight variable as indicated by the period (.) sign.

Lastly, let's consider another example by describing the contents of EducLev as shown below.

```
. codebook EducLev
```

```
-------------------------------------------------------------------------------
EducLev                                                       Education Level
-------------------------------------------------------------------------------

                 type:  numeric (byte)
                label:  EducLev

                range:  [0,3]                        units:  1
        unique values:  4                    missing .:  5/1464

           tabulation:  Freq.   Numeric  Label
                          297         0  No Education
                          515         1  Primary
                          492         2  Secondary
                          155         3  Tertiary
                            5         .
```

Here, 297 patients were not educated, 515 patients were in primary education, 492 patients were in secondary education and 155 patients were tertiary education. Notice that 5 responses are missing as indicated by the period (.) sign.

The `codebook` command with no specified variable refers to all variables and tells Stata to list every single variable in the dataset. That is, to get the information for all the variables in the file, simply type `codebook` without specifying the variable(s).

```
. codebook
```

If we've altered a dataset or just want a quick look at the data, the `codebook` command is a good place to start.

4.6 Frequency Tables

Now, let's start with the simplest of analytical commands and create frequency tables. Stata can produce one-way and two-way frequency tables which are useful for categorical variables.

4.6.1 One-Way Tables: The tabulate and tab1 Commands

The Stata command tabulate creates frequency table. To begin, let's look at the reported distribution of patients' education level (EducLev) using the tabulate command.

```
. tabulate EducLev

   Education |
       Level |      Freq.     Percent        Cum.
-------------+-----------------------------------
No Education |        297       20.36       20.36
     Primary |        515       35.30       55.65
   Secondary |        492       33.72       89.38
    Tertiary |        155       10.62      100.00
-------------+-----------------------------------
       Total |      1,459      100.00
```

Now that's odd, isn't it? We know from our previous codebook command that we should expect 1,464 responses. We're short here by a number of responses. In fact, let's use Stata to calculate the number of responses by which we're actually short:

```
. display 1464 - 1459
5
```

After using the Stata's built-in calculator, we can tell that we've 5 observations not included in the table. What happened to them? By default, Stata leaves out all missing observations. If the missing option, which forces Stata to include the missing responses, is specified, missing values will be included in the frequency counts as shown below.

```
. tabulate EducLev ,missing

   Education |
       Level |      Freq.     Percent        Cum.
-------------+-----------------------------------
No Education |        297       20.29       20.29
     Primary |        515       35.18       55.46
   Secondary |        492       33.61       89.07
    Tertiary |        155       10.59       99.66
           . |          5        0.34      100.00
-------------+-----------------------------------
       Total |      1,464      100.00
```

Recall from above that missing observations are shown with a period (.). And, moreover, the 5 missing observations is exactly the same number as we got from the display command. Nice.

To ask for more than one frequency table in a single command, the tab1 command is used. It produces one-way frequency table for each variable in the variable list.

```
. tab1 Sex MarStat ,missing

-> tabulation of Sex

        Sex |      Freq.     Percent        Cum.
------------+-----------------------------------
          F |        930       63.52       63.52
          M |        534       36.48      100.00
------------+-----------------------------------
      Total |      1,464      100.00

-> tabulation of MarStat

      Marital |
       Status |      Freq.     Percent        Cum.
--------------+-----------------------------------
Never Married |        293       20.01       20.01
      Married |        739       50.48       70.49
     Divorced |        134        9.15       79.64
    Separated |        140        9.56       89.21
      Widowed |        154       10.52       99.73
            . |          4        0.27      100.00
--------------+-----------------------------------
        Total |      1,464      100.00
```

4.6.2 Two-Way Tables: The tabulate and tab2 Commands

For two or more categorical variables, the data can be summarized in a tabular form in which the cells of the table contain number of observations (frequencies) in the intersection categories of the variables. Such a table is called contingency table (cross-tabulation). Two-way frequency tables (contingency tables) are useful for determining the association between two categorical variables.

For example, using the JUSH_HAART data, in order to determine how the survival outcome (Status) varies by patient functional status (FunStat) type in:

```
. tabulate FunStat Status

Functional |              Survival Outcome
    Status |    Active       Dead  Transferr  Lost-to-f |     Total
-----------+--------------------------------------------+----------
   Working |       815         20         58        110 |     1,003
Ambulatory |       287         18         47         52 |       404
 Bedridden |        31          7         10          9 |        57
-----------+--------------------------------------------+----------
     Total |     1,133         45        115        171 |     1,464
```

Clearly, the table shows that there are 815 patients who were active and working, 287 patients who were ambulatory and active, and 31 patients who were bedridden and active. Also, of the

20 dead patients; 20, 18 and 7 of them were working, ambulatory and bedridden, respectively. The same description applies to the other frequencies.

But, I have one simple but critical question! Of the three functional status categories (Working, Ambulatory and Bedridden, which group was more likely to be dead? If your answer is Working, surprisingly you're wrong. Why?

To answer this question, we'd like to know about the proportions by functional status of the patient. How might we do that? In the two-way tables command, several options can be used. Of these, `row` gives row percentages and `col` gives column percentages. And, `cell` gives the overall percentage and `expected` reports the expected frequency in each cell. In addition, the `nofreq` option suppresses printing the frequencies while the `nolabel` option suppresses the use of value labels, showing the numeric values instead.

Hence, to know the proportions by functional status of the patient, we can add the `row` option in the `tabulate` command.

```
. tabulate FunStat Status ,row
```

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

Functional |              Survival Outcome
    Status |    Active       Dead  Transferr  Lost-to-f |     Total
-----------+--------------------------------------------+----------
   Working |       815         20         58        110 |     1,003
           |     81.26       1.99       5.78      10.97 |    100.00
-----------+--------------------------------------------+----------
Ambulatory |       287         18         47         52 |       404
           |     71.04       4.46      11.63      12.87 |    100.00
-----------+--------------------------------------------+----------
 Bedridden |        31          7         10          9 |        57
           |     54.39      12.28      17.54      15.79 |    100.00
-----------+--------------------------------------------+----------
     Total |     1,133         45        115        171 |     1,464
           |     77.39       3.07       7.86      11.68 |    100.00
```

Therefore, now, it is obvious that bedridden patients are more likely to be dead.

To ask for more than one two-way frequency table in a single command, the `tab2` command is used. This command produces all possible two-variable tables from the list of variables. For example, the command

```
. tab2 FunStat Status ClinStag
```

produces three two-way tables.

4.7 Descriptive Statistics

4.7.1 Summary Statistics: The summarize Command

The summarize command provides the number of valid observations, mean, standard deviation, as well as the minimum and maximum for each numeric variable.

Now try to type the summarize command and examine the results.

```
. summarize

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     CardNum |      1464    2551.764     658.542      1202       3587
         Age |      1464    34.01161    9.160258        18         85
         Sex |         0
         Wei |      1460    51.86788    10.19338        16         96
     MarStat |      1460    1.399315    1.211154         0          4
-------------+--------------------------------------------------------
     EducLev |      1459    1.346127    .9200576         0          3
         Emp |      1444    1.850416    1.375886         0          3
    ClinStag |      1464    2.247951     .898466         1          4
     FunStat |      1464    .3538251    .5538152         0          2
         CD4 |      1464    198.1906    171.2403         1       1914
-------------+--------------------------------------------------------
      Status |      1464    .5382514    1.052244         0          3
   Defaulter |      1464    .2260929    .4184429         0          1
    SurvTime |      1464    25.19768    16.70695   .0333333         54
```

Two notes to be considered here. First, note that the string variable Sex has no information listed, not even the number of observations. This is, because, once a variable is declared as string, Stata does not apply any mathematical operation or statistical analysis.

Second, note that even if the mean and standard deviation of the those categorical variables, which are coded as numeric, are calculated, such values are meaningless. Hence, the reported mean and standard deviations for MarStat, EducLev, Emp, ClinStag, FunStat, Status, Defaulter as well as CardNum have no meaning at all. Therefore, keep in mind that the summarize command is appropriate only for quantitative variables.

The detail option in the summarize command gives additional statistics (skewness, kurtosis, the four smallest values, the four largest values, various percentiles) for the specified variables.

```
. summarize Wei ,detail

                    Weight in Kilograms
-------------------------------------------------------------
      Percentiles      Smallest
 1%           31             16
 5%        36.75             18
```

```
10%          40              26       Obs                 1460
25%          45              27       Sum of Wgt.         1460

50%          51                       Mean           51.86788
                     Largest           Std. Dev.      10.19338
75%       57.75              90
90%          65              92       Variance       103.9049
95%          70              96       Skewness       .5189258
99%          80              96       Kurtosis       3.901936
```

4.7.2 Compact Table of Summary Statistics: The tabstat Command

The tabstat command displays summary statistics for a series of numeric variables in one table. The command by default gives the mean but allows us to specify a list of descriptive statistics with the option statistics (mean, median, min, max, range, sd, variance, skewness, kurtosis, percentiles: p1, p2, $\cdots$, p99).

```
. tabstat Age Wei CD4 SurvTime ,stat(n mean min p25 p50 p75 max)
```

```
   stats |       Age       Wei       CD4  SurvTime
---------+--------------------------------------------
       N |      1464      1460      1464      1464
    mean |  34.01161  51.86788  198.1906  25.19768
     min |        18        16         1  .0333333
     p25 |        28        45        85  10.83333
     p50 |        32        51       158     23.95
     p75 |        39     57.75       266     39.05
     max |        85        96      1914        54
----------------------------------------------------
```

Like the summarize command, the tabstat results are valid only for quantitative variables. That is why, in the above command, we have listed quantitative variables only.

As shown above output, the tabstat command displays the variables in columns. But, it is helpful to put statistics in columns. The columns(statistics) is used to do so, instead of the default option columns(variables).

```
. tabstat Age Wei CD4 SurvTime ,stat(n min max mean sd p50) col(stat)
```

The output is:

```
    variable |         N       min       max      mean        sd       p50
-------------+--------------------------------------------------------------
         Age |      1464        18        85  34.01161  9.160258        32
         Wei |      1460        16        96  51.86788  10.19338        51
         CD4 |      1464         1      1914  198.1906  171.2403       158
    SurvTime |      1464  .0333333        54  25.19768  16.70695     23.95
----------------------------------------------------------------------------
```

The `tabstat` command also provides summary statistics broken down on (conditioned by) another variable using the `by()` option. For example, to obtain the above descriptive statistics for male and female, we can do as follows.

```
. tabstat Age Wei CD4 SurvTime ,stat(n min max mean sd p50) col(stat) by(Sex)
```

Then Stata provides the result:

```
Summary for variables: Age Wei CD4 SurvTime
     by categories of: Sex (Sex of the Patient)

   Sex |         N       min       max      mean        sd       p50
-------+--------------------------------------------------------------
     F |       930        18        85  32.01828  8.600241        30
       |       927        16        96  49.68285  9.916851        49
       |       930         2      1914  209.7118  178.1555       173
       |       930        .1        54  25.81667  16.63067     24.55
-------+--------------------------------------------------------------
     M |       534        18        79  37.48315  9.077253        36
       |       533        29        92  55.66811  9.538948        55
       |       534         1      1352  178.1255   156.627       140
       |       534  .0333333        54  24.11966  16.80015      23.3
-------+--------------------------------------------------------------
 Total |      1464        18        85  34.01161  9.160258        32
       |      1460        16        96  51.86788  10.19338        51
       |      1464         1      1914  198.1906  171.2403       158
       |      1464  .0333333        54  25.19768  16.70695     23.95
---------------------------------------------------------------------
```

4.8 Selecting Cases: The `in` and `if` Qualifiers

Instead of just wanting to look at all possible values and observations for a particular variable, we'd like to restrict the output of various commands. How would we do that? We'd use a conditional statement. The `in` and the `if` qualifiers (parameters) can be added to any Stata command to select a subset of cases to be used.

The `in` Qualifier

The `in` qualifier selects cases based on their position in the data file. For example, to list the first 3 observations using the `in` command:

```
. list in 1/3
```

To list observations from the end of the file, negative numbers are used. Hence, to list the last 3 observations:

```
. list in -3/-1
```

The `if` Qualifier

The `if` qualifier tests for equality and it helps to select cases that satisfy some logical criterion. It is extremely useful and works with many commands. The symbols to use for building logical expressions are:

Meaning	Symbol
Equal to	==
Greater that or equal to	>=
Less than or equal to	<=
Greater than	>
Less than	<
Not equal to	! = or ~=
And	&
Or	|

To specify a particular string value, enclose it in double quotes. Stata is case-sensitive, so each of the following `if gender== "male", if gender== "Male", if gender== " male", if gender== "male "` is evaluated as a unique string. Also note the use of the double equal sign.

Let's start with a basic command, `count`, which counts the number of cases (observations) in the dataset. Simply typing `count` in the command window and pressing the Enter key shows that there are 1464 observations or cases.

Say we're looking for patients whose age (Age) is under 40. So, how might we structure a command that counts age if the value is under 40? We can do this as:

```
. count if Age <40
```

Hence, there are 1111 patients whose age was under 40 years. Similarly, we can count those patients who were 40 years. Apparently 88 patients were 40 years as determined by the command:

```
. count if Age==40
```

Now, let's find the number of patients who were above 95 kilograms. The command

```
. count if Wei>95
```

tells us there are 6 patients (observations) whose weight was above 95 kgs. In fact, this value is wrong. Why?

Careful consideration must be paid to missing values in a data file. By default, Stata uses a '.' for numeric missing values. Internally, '.' is stored as a very, very large number. Hence, since there are 4 missing values in the variable Wei, the above command adds the number of missing values as values greater than 95. Therefore, **be careful!**

While using the > and >= conditions with variables having missing values, the missing values must be excluded. Now to exclude the missing values from the counting and listing of the Wei variable, the command must be written as:

```
. count if Wei>95 & Wei < .
```

or

```
. count if Wei>95 & Wei != .
```

which indicates that there are only 2 observations. To list these observations, we can the usual `list` command together with the `if` qualifier. That is,

```
. list if Wei>95 & Wei < .
```

or

```
. list if Wei>95 & Wei != .
```

list the two observations as follows.

```
      +-------------------------------------------------------------------+
1189. | CardNum | Age | Sex | Wei  | MarStat  |   EducLev  |        Emp  |
      |    3229 |  40 |   F |  96  | Married  | Secondary  |  Full-time  |
      |---------------------+-----------------------------------------------|
      |  ClinStag | FunStat | CD4 | Status | Defaul~r | Days |  SurvTime |
      |   Stage I | Working |  57 | Active |   Active |  251 | 8.3666667 |
      +-------------------------------------------------------------------+

      +-------------------------------------------------------------------+
1245. | CardNum | Age | Sex | Wei  | MarStat  |   EducLev  |        Emp  |
      |    3294 |  40 |   F |  96  | Married  | Secondary  | Unemployed  |
      |---------------------+-----------------------------------------------|
      |  ClinStag | FunStat | CD4 | Status | Defaul~r | Days |  SurvTime |
      | Stage III | Working | 224 | Active |   Active |  914 | 30.466667 |
      +-------------------------------------------------------------------+
```

4.9 Manipulating Data

4.9.1 Sorting Observations: The `sort` and `gsort` Commands

By default, the `sort` command sorts observations in ascending order (from smallest to largest or from A to Z) only, based upon the values of the variables specified.

```
. sort VarName
```

Here, the order of observations that have equal values with the sorting variable is randomized.

To sort observations in ascending order, but maintaining the relative order of equal values prior to the sort, the `stable` option can be added.

```
. sort VarName ,stable
```

The `stable` option insures that the observations stay in the same relative order that they held before the sort.

The `gsort` command sorts in either descending or ascending order. The commands

```
. gsort VarName
```

and

```
. gsort +VarName
```

are equivalent to the previous command; the observations are sorted in ascending order the specified variable.

To sort observations in descending order, the command is:

```
. gsort -VarName
```

The following command sorts the observations by ascending order (from A to Z) of Var1 and then within each Var1 value, the data is sorted in a descending order of Var2.

```
. gsort Var1 -Var2
```

4.9.2 Sorting Variables: The `order` and `aorder` Commands

While the `sort` and `gsort` commands sort data vertically, the commands `order` and `aorder` allow sorting the data horizontally, meaning changing the order of the variables. Normally this is not a very important feature, but there are situations when it might be necessary (e.g. to have the identifier of the observations at the beginning for easier use).

The `order` command sorts the variables in the dataset based on the list after it.

```
. order Var2 Var1 Var3
```

Also, there is a command `aorder` which allows to order of variables an alphabetical order. By simply writing

```
. aorder
```

without any list of variable, all the variables will be ordered alphabetically, where capital letters come first, followed by special symbols like underscores (_) and then lower case letters.

4.9.3 Deleting Observations (Variables): The `drop` and `keep` Commands

The `drop` and `keep` commands can be used to delete either observations or variables. Be careful when using these commands as we will not be able to get back to the original data once it is saved.

When dealing with observations, these commands are mostly combined with the `if` and `in` qualifiers. The `drop` command deletes records satisfying the logical criteria while the `keep` command deletes everything except those observations satisfying the logical criteria.

For example, if we want to remove all observations in which the weight is missing in the JUSH_HAART data, we can do it using both the `drop` and `keep` commands. That is, both the following commands

```
. drop if Wei==.
```

or

. keep if Wei! = .

removes observations in which the weight is missing.

If we want to delete all observations in which the weight is less than 50, then the command is

. drop if Wei<50

or

. keep if Wei>=50

Similarly, to keep all observations in which the weight is greater than 95, we do as

. drop if Wei<=95 & Wei== .

or

. keep if Wei>95 & Wei== .

To delete the first five observations, the `in` qualifier is used as follows.

. drop in 1/5

This command tells Stata to drop the first five observations.

The `drop` and `keep` commands are also used to get rid of the unnecessary variables. The structures of the commands are:

. drop VarList
. keep VarList

The `drop` command deletes variables that are listed after it while the `keep` command deletes everything except the specified variables. In fact, whether to use the `drop` or `keep` command depends upon the number of variables we want. If there are only few variables that we do not want, then we can use the `drop` command. If, however, there are more variables that we do not want, then we can use the `keep` command.

4.9.4 Identifying Duplicate Values: The `duplicates` Command

Note that any data should have a unique identifier (ID) to each observation. The unique identifier of each observation can be a single variable, for example, CardNum for the JUSH_HAART data. Also, the unique identifier might consist of a series of variables (e.g., PrimaryID and SecondaryID). For example, multiple cases share a common primary ID value but different secondary ID values, such as family members who all live in the same house.

The `duplicates` command is used to identify whether there are duplicates based on one variable, for example, by CardNum for the JUSH_HAART data. To list such duplicate observations, the command is:

. duplicates list CardNum

It results

```
Duplicates in terms of CardNum

(0 observations are duplicates)
```

indicating that there is no duplicate. Had we get duplicated CardNum's and decide to remove multiple observations that are exact matches, the following command is used.

```
. duplicates drop CardNum ,force
```

4.9.5 Renaming a Variable: The rename Command

If there is a need of renaming a variable, the rename command can be used so that the variable name can be edited.

```
. rename OldName NewName
```

For instance, to rename the Wei variable by Weight, then the command is written

```
. rename Wei Weight
```

It changes the name Wei by the new name Weight.

4.9.6 Replacing Values: The replace Command

This command is used to change the contents of a variable. It is often used with the if qualifier. The structure of the command for a numeric variable is:

```
. replace VarName=NewValue if VarName==OldValue
```

which replaces all the old values by the new value. But, if the variable is string, the values should be enclosed in double quotes as follows.

```
. replace VarName= "NewCharacters" if VarName== "OldCharacters"
```

This replaces all the old characters by the new characters.

For example, the first of the following examples corrects typos in a string variable. The second changes missing values that were coded as -99 to Stata's "." missing value.

```
. replace Sex = "Male" if Sex == "Mael"
. replace Wei=. if Wei==-99
```

In the JUSH_HAART data, Sex is string with values M and F. Let's replace the M value of Sex by Male and the F value by Female.

```
. replace Sex = "Male" if Sex == "M"
. replace Sex = "Female" if Sex == "F"
```

4.9.7 Creating New Variables: The generate and replace Commands

Variable transformation is a way of creating new variables using existing continuous variables and formulae. To create a new variable, use the generate command. Constants and variables can both be used to create new variables. But, spacing is not important; operators can have spaces before and/or after or none at all.

Basic arithmetic expressions are formed using the operators: + for addition (numeric), - for subtraction, $*$ for multiplication, / for division and $\wedge$ for power. Some of the common mathematical functions that can be used in creating a variable are sqrt(VarName), log(VarName), log10(VarName), abs(VarName).

```
. generate NewVarName = Formula
```

The following command generates a new variable, RootCD4, by taking the square root of the CD4 variable.

```
. generate RootCD4=sqrt(CD4)
```

If we want to change an existing variable, we need to use the command replace instead of the generate command. That is,

```
. replace CD4=sqrt(CD4)
```

which replaces the original CD4 values by their square root values.

4.9.8 Changing Numeric Variables to String: The tostring Command

A string variable can consist solely of numbers, but mathematical operations cannot be performed with it. Therefore, it can be a good idea to format numbers for which a mathematical operation is inappropriate, such as identification numbers, as string variables. For our dataset, CardNum is patients' card number which must be changed to string.

```
. tostring CardNum ,generate(CardNumStr)

CardNumStr generated as str4
```

4.9.9 Changing String Variables to Numeric: The destring Command

If a variable was mistakenly imported as a string variable when it should have been numeric, the destring command will convert it. Before trying to convert the variable to a numeric format, the input error that caused the variable to be stored as a string must be fixed. For example, if the following set of numbers was imported as the variable xyz, the data will be stored as a string because of the letter "b" in the first observation, the letters "n" and "a" in the third observation, and the space in the second observation.

0.3b08
na
0.2 72
0.215
0.299

If there are many instances of such a specific problem, such as the use of a comma separator or 'na' rather than a '.' for missing values, then the `ignore` option under the `destring` command can fix the problem. Creating a new variable with the `generate` option is much safer than overwriting the existing variable with the `replace` option.

```
. destring VarName ,ignore("CharactersToBeRemoved") generate(NewVarName)
```

Let's enter the above five observations in excel under a variable name xyz and save it. If we import this data to Stata, the variable will be treated as string due to the above mentioned nonnumeric characters. To remove the characters that causes the variable xyz to be string and generate a new numeric variable named xyz_new, the following command can be used.

```
. destring xyz ,ignore("b" "na" " ") gen(xyz_new)
xyz: characters b space n a removed; xyz_new generated as double
(1 missing value generated)
```

This command removes each of the five characters; b, n, a and space. Since, the `ignore` option removes each character one by one, the command can also be written as:

```
. destring xyz ,ignore("bna ") gen(xyz_new2)
xyz: characters b n a space removed; xyz_new2 generated as double
(1 missing value generated)
```

Therefore, both new variables xyz_new and xyz_new2 are the same as we can see using the `list` command.

```
     +-----------------------------+
     |    xyz   xyz_new2   xyz_new |
     |-----------------------------|
  1. | 0.3b08       .308      .308 |
  2. |     na          .         . |
  3. | 0.2 72       .272      .272 |
  4. |   .215       .215      .215 |
  5. |   .299       .299      .299 |
     +-----------------------------+
```

4.9.10 Coding a String Variable: The `encode` Command

If there is a string variable, such as Sex where the values are "F" and "M" in our case, we may want to assign numeric values to them. The `encode` command is a convenient way to assign numeric values and create value labels using the distinct values of a string variable.

```
. encode StringVar ,generate(NewVar)
```

The command automatically assigns the values and labels to the categories of the variable StringVar and generates a numeric variable NewVar.

Now let's code the Sex variable and then generate a new variable, say, Gender.

```
. encode Sex ,generate(Gender)
```

If we look at the data editor, it will appear that the original variable Sex and the newly generated variable Gender are the same except that the original variable values are red and the new one is blue. Also, if we execute the following `list` command,

```
. list Sex Gender
```

results

```
     +--------------+
     | Sex   Gender |
     |--------------|
  1. |   F        F |
  2. |   M        M |
  3. |   F        F |
  4. |   M        M |
  5. |   M        M |
     |--------------|
--more--
```

which seems there is no difference between Sex and Gender. But, the new variable, Gender in this example, is using the label and the underlying value is numeric. Use the `nolabel` option in the `list` command to see the underlying values.

```
. list Sex Gender ,nolabel
```

```
     +--------------+
     | Sex   Gender |
     |--------------|
  1. |   F        1 |
  2. |   M        2 |
  3. |   F        1 |
  4. |   M        2 |
  5. |   M        2 |
     |--------------|
--more--
```

Or we can observe the difference between Sex and Gender using the command `codebook` as follows.

```
. codebook Sex Gender
```

```
-------------------------------------------------------------------------------
Sex                                                                         Sex
-------------------------------------------------------------------------------

                  type:  string (str1)

         unique values:  2                        missing "":  0/1464
```

```
            tabulation:  Freq.  Value
                           930  "F"
                           534  "M"

-------------------------------------------------------------------------------
Gender                                                                      Sex
-------------------------------------------------------------------------------

                  type:  numeric (long)
                 label:  Gender

                 range:  [1,2]                        units:  1
         unique values:  2                          missing .:  0/1464

            tabulation:  Freq.   Numeric  Label
                           930         1  F
                           534         2  M
```

4.9.11 Redefining a Categorical Variable: The `recode` Command

The numeric values of Gender for 'F' and 'M' patients are 1 and 2, respectively. That's not what we want. Suppose we want the typical 0 = F, 1 = M setup. How might we do this? Yes, you guessed it. We'll do what we call 'recoding' a variable.

The `recode` command is used to redefine the values of such a categorical variable according to the rules specified. Below are some examples of using this command.

`recode` x 1=2 ⇒ changes all values of x=1 to x=2
`recode` x 1=2 3=4 ⇒ in the variable x, changes 1 to 2 and 3 to 4
`recode` x 1=2 2=1 ⇒ in the variable x, exchanges the values 1 and 2
`recode` x 1=2 ∗=3 ⇒ in the variable x, changes 1 to 2 and all other values to 3
`recode` x 1/5=2 ⇒ in the variable x, changes 1 through 5 to 2
`recode` x 1 3 4 5 = 6 ⇒ in the variable x, changes 1, 3, 4 and 5 to 6
`recode` x .=9 ⇒ in the variable x, changes missing to 9
`recode` x 9=. ⇒ in the variable x, changes 9 to missing

Notice that we can use some special symbols in the recode command. For example, . means missing values, 2 4 means values 2 and 4, 2/4 means all values from 2 to 4 and ∗ means all other values.

Also, the `generate` option with `recode` command creates a new recoded variable instead of replacing the existing variable.

. `recode` VarName (OldValue1=NewValue1) (OldValue2=NewValue2) ,`generate`(NewVar)

This command changes OldValue1 into NewValue1 and OldValue2 into NewValue2 of the variable VarName, and generates a new variable NewVar. In this command, the labels of the new variable can also be specified as follows.

```
. recode VarName (OldValue1=NewValue1 "Label of the NewValue1")
                 (OldValue2=NewValue2 "Label of the NewValue2") ,gen(NewVar)
```

When the labels are specified, the brackets are mandatory.

Recall Gender was generated with labels 1=F and 2=M. Now let's recode Gender using the setup 0=F and 1=M, and add the labels Female and Male.

```
. recode Gender (1=0 "Female") (2=1 "Male") ,generate(Gender_new)
(1464 differences between Gender and Gender_new)
```

Note a single equal sign (=) is used in assignment expressions while a double equal sign (==) is a logical operator. Now let's observe the difference between Gender and Gender_new using the `list` and `codebook` commands.

```
. list Gender Gender_new
```

```
     +-------------------+
     | Gender   Gender~w |
     |-------------------|
  1. |      F     Female |
  2. |      M       Male |
  3. |      F     Female |
  4. |      M       Male |
  5. |      M       Male |
     |-------------------|
```

The above partial output shows the labels of the two variables. Let's also see the values of the variables using the `nolabel` option.

```
. list Gender Gender_new ,nolabel
```

```
     +-------------------+
     | Gender   Gender~w |
     |-------------------|
  1. |      1          0 |
  2. |      2          1 |
  3. |      1          0 |
  4. |      2          1 |
  5. |      2          1 |
     |-------------------|
```

Clearly, the codes for `Gender` are 1 and 2 while the codes for `Gender_new` are 0 and 1. Also, using the `codebook` command, we can easily observe the difference the values and the labels of the two variables.

```
. list Gender Gender_new
```

```
-------------------------------------------------------------------------------
Gender                                                                      Sex
-------------------------------------------------------------------------------

                 type:  numeric (long)
                label:  Gender

                range:  [1,2]                        units:  1
        unique values:  2                    missing .:  0/1464

           tabulation:  Freq.   Numeric  Label
                          930         1  F
                          534         2  M

-------------------------------------------------------------------------------
Gender_new                                          RECODE of Gender (Sex)
-------------------------------------------------------------------------------

                 type:  numeric (long)
                label:  Gender_new

                range:  [0,1]                        units:  1
        unique values:  2                    missing .:  0/1464

           tabulation:  Freq.   Numeric  Label
                          930         0  Female
                          534         1  Male
```

4.9.12 Creating Value Labels: The label Command

If we use the `recode` command with no labels as:

```
. recode Gender (1=0) (2=1) ,generate(Gender_alt)
```

then, this newly generated variable Gender_alt is displayed as 0 and 1. Hence, it would be nice to have the values of the Gender_alt labeled with their meaning, rather than displayed as 0 and 1.

We already know labeling values has two components in section 3.1.2. First creating a label that associates text with the codes, and then second assigning the label to one or more variables. The corresponding commands are `label define` and `label values`.

```
. label define Gender_lab 0 "Female" 1 "Male"
```

These labels are not yet associated with any variable. To associate these labels to the variable Gender_alt, we do as follows.

```
. label values Gender_alt Gender_lab
```

Now we can examine the variable in the data editor or using the `codebook` command. Note that Gender_new and Gender_alt are both exactly the same even if we have used two different methods of creating value labels for the recoded variable.

4.9.13 Collapsing a Continuous Variable: The `recode` Command

The `recode` command is also useful to collapse a continuous variable into categorical groups. For example, to code values from the smallest to `a` as 0, values from `b` to `c` as 1 and values from `c` to the largest as 2, the command takes the following form.

```
. recode VarName (min/a=0) (b/c=1) (c/max=2) ,generate(NewVar)
```

if `min`$< a < b < c <$`max`.

As an example, let's consider categorizing Wei into four categories (weight $\leq$30, 30.5-50, 50.5-70, $>$70) and generate a new variable, Wei_rec.

```
. recode Wei (min/30=0) (30.5/50=1) (50.5/70=2) (70.5/max=3) ,gen(Wei_rec)
(1460 differences between Wei and Wei_rec)
```

Now we can look at the recoded Wei_rec variable.

```
. codebook Wei_rec

-------------------------------------------------------------------------------
Wei_rec                                                 RECODE of Wei (Weight)
-------------------------------------------------------------------------------

                  type:  numeric (double)

                 range:  [0,3]                        units:  1
         unique values:  4                    missing .:  4/1464

            tabulation:  Freq.  Value
                           13  0
                          690  1
                          687  2
                           70  3
                            4  .
```

Now let's create a label for this new variable because it is nice to have the values of the above newly recoded weight Wei_rec labeled with their meaning ($\leq$ 30 kg, 30.5-50 kg, 50.5-70 kg and $>$70 kg), rather than displayed as 0, 1, 2, and 3.

```
. label define Wei_lab 0 "<=30 kg" 1 "30.5-50 kg" 2 "50.5-70 kg" 3 ">70 kg"
```

To associate these labels to the variable Wei_rec, we do:

```
. label values Wei_rec Wei_lab
```

Now we can examine this variable using the `codebook` command.

Also, we can create the labels while recoding the variable. That's;

```
. recode Wei (min/30=0 "<=30 kg") (30.5/50=1 "30.5-50 kg") (50.5/70=2
"50.5-70kg") (70.5/max=3 ">70 kg") ,generate(Wei_new)
```

automatically creates the labels and associate with the variable. Thus, Wei_rec and Wei_new are both the same.

4.9.14 Creating Dummy Variables: The tabulate Command

In section 6.2, we've seen that `tabulate` command is used to construct one-way and two-way frequency tables. In addition, this command together with the `generate` option is useful for creating a set of design (dummy) variables (variables with a value of 0 or 1) depending on the value of an existing categorical variable.

```
. tabulate OldVar ,generate(NewVar)
```

For example, since MarStat has 5 categories, there are 5 possible dummy variables. To create these dummy variables from MarStat, the command

```
. tabulate MarStat ,generate(Marital)
```

automatically creates the five dummy variables in addition to the one-way frequency table. The five new binary variables, defined as follows:

Marital1=1 if MarStat=0 and 0 otherwise
Marital2=1 if MarStat=1 and 0 otherwise
Marital3=1 if MarStat=2 and 0 otherwise
Marital4=1 if MarStat=3 and 0 otherwise
Marital5=1 if MarStat=4 and 0 otherwise

In this example, notice that there are 293 patients in MarStat=0 (Never Married) and the same number of patients for which Marital1=1. Again there are 739 patients in MarStat=1 (Married) and the same number of patients for which Marital2=1, and so on.

4.10 Restructuring Longitudinal (Panel) Data

Longitudinal data can be arranged in two different forms: long or wide. In the long (person-period) format, there are, potentially, multiple rows per subject and observations on a variable for different time periods (or dates) held in extra rows for each individual. For example, consider the following data on Students' GPA:

StudentID	Semester	GPA	Female
251	1	3.51	0
251	2	3.25	0
251	3	3.63	0
251	4	3.70	0
251	5	3.65	0
251	6	3.20	0
257	1	3.67	1
257	2	3.90	1
257	3	3.78	1
257	4	3.50	1
257	5	3.82	1
257	6	3.90	1

This data is arranged in the person-period format which is characterized by

1. a time-invariant unique identifier for each unit (Panel Variable) (StudentID)
2. an indicator for time (Time Variable) (Semester)
3. a time-varying outcome variable (GPA)

This form is usually the most convenient one which needed for most statistical analysis. On the other hand, the data can be arranged in the wide format in which there is only one row per subject and observations on a variable for different time periods (or dates) held in different columns. The response variable name contains time values at the end (suffix) or in the middle, but not at the beginning. For example, the above data can be arranged as follows.

StudentID	GPA1	GPA2	GPA3	GPA4	GPA5	GPA6	Female
251	3.51	3.25	3.63	3.70	3.65	3.20	0
257	3.67	3.90	3.78	3.50	3.82	3.90	1

Note that in the wide format, a time variable is not there. Rather it is with the response variable name as a suffix.

4.10.1 Changing from Long-to-Wide Form: The `reshape wide` Command

The `reshape` command is used to convert the longitudinal data from long to wide form and vice versa. When using this command, the above three characteristics (panel variable, time variable and response) are needed.

The structure of the command for converting long-to-wide is:

```
. reshape wide Response ,i(PanelVar) j(TimeVar)
```

Load the CD4Count.dta to Stata. In this data, the panel variable is CardNum, the time variable is ObsTime and the response is CD4. Since, the data is in long form, to change it to wide form, the command is written as follows.

```
. reshape wide CD4 ,i(CardNum) j(ObsTime)
```

```
(note: j = 0 6 12 18 24 30 36 42 48 54)

Data                               long   ->   wide
-----------------------------------------------------------------------------
Number of obs.                     4655   ->    1464
Number of variables                   3   ->      11
j variable (10 values)          ObsTime   ->   (dropped)
xij variables:
                                    CD4   ->   CD40 CD46 ... CD454
-----------------------------------------------------------------------------
```

As the output above tells, the number of cases were 4655 in long form and now the number of cases is 1464 in the wide form. There were 3 variables in the long form and now there are 11 variables in the wide form. In addition, when converting from long-to-wide, the existing time variable will be dropped.

Note also that in changing from long-to-wide form, it can contain the symbol @ for denoting where j is to appear in the column name in the wide form. For example, when we wrote `reshape wide response`, we could have written "`reshape wide response@`" because j by default ends up as a suffix. Had we written respon@se, then the wide variables would have been named as like respon1se, respon2se, $\cdots$.

4.10.2 Changing from Wide-to-Long Form: The `reshape long` Command

In the wide form, there is no time variable. As a result, when converting from wide-to-long, the time variable is generated as a new variable. To convert from wide-to-long, the syntax structure is as follows.

```
. reshape long Response ,i(PanelVar) j(TimeVar)
```

Here the response are column names of variable names. It may contain @ for denoting where j appears in the wide form.

Given the following hypothetical data on a random sample of former smokers with year after stopping smoking and weight (Wei) in kilograms at that time for 3 individuals.

Subject	Wei0	Wei1	Wei2	Wei3	Wei4	Wei5	Wei6	Wei7
1	56	56	57	58	59	61		
2		54	54	55	55	56	56	
3			51	52	52	54	54	55

After entering these data in the data editor as it is, it can be easily converted into long form using the following command.

```
. reshape long Wei ,i(Subject) j(Time)

(note: j = 0 1 2 3 4 5 6 7)

Data                               wide   ->   long
```

```
-----------------------------------------------------------------------------
Number of obs.                        3   ->      24
Number of variables                   9   ->       3
j variable (8 values)                     ->   Time
xij variables:
                     Wei0 Wei1 ... Wei7   ->   Wei
-----------------------------------------------------------------------------
```

Again to convert from long to wide back:

```
. reshape wide Wei ,i(Subject) j(Time)
```

or we can just type

```
. reshape long
```

because we have used the `reshape wide` command previously. Then, Stata provides the following result.

```
(note: j = 0 1 2 3 4 5 6 7)

Data                               long   ->   wide
-----------------------------------------------------------------------------
Number of obs.                       24   ->       3
Number of variables                   3   ->       9
j variable (8 values)              Time   ->   (dropped)
xij variables:
                                    Wei   ->   Wei0 Wei1 ... Wei7
-----------------------------------------------------------------------------
```

And to go back to wide form after using `reshape long` command, just type

```
. reshape wide
```

4.11 Combining Data Sets

It is often needed to merge several datasets into one. Stata is very efficient in such kind of data handling. There are two main situations for combining datasets. The first situation is when there are two datasets with *same variables but different observations (cases)* in which additional observations from one data file can be added to the end of another. And the second situation is when there are two datasets with the *same observations (cases) but different variables* in which additional variables contained in one file can be added to corresponding observations in another.

The data file in memory (the one that is currently opened) is referred to as the 'master' (working) file. The file that is to be joined with the 'master' is known as the 'using' data file. Both files must be Stata files.

4.11.1 Adding Observations: The `append` Command

The `append` command adds observations from the using file to the end of the master file. The files are stacked vertically. The variables in both datasets should have the same names and format.

The structure of the `append` command is:

```
. append using UsingFile
```

Example 4.1. In the folder given to you, there are two data files, DataForAppend_1 and DataForAppend_2 which have some same variables but different cases. Let's add the cases from DataForAppend_2 to DataForAppend_1. After opening DataForAppend_1, just type:

```
. append using DataForAppend_2
```

Now it can easily checked in the data editor that there are 8 observations.

4.11.2 Adding Variables: The `merge` Command

The `merge` command adds additional variables to the data in memory. That is, the command combines two data files with different variables into one file. Both files should have a matching variable (or variables) that is used to associate an observation from the master file with an observation in the using file. Before the files being merged, they must be sorted by the matching variable(s).

There are four different types of merge.

- **One-to-one** merging: In a one to one match (`1:1`), each observation in the master file has a corresponding observation in the using file.
- **One-to-many** merging: In a one to many merge (`1:m`), the using file has multiple observations per each unique key variable in the master file.
- **Many-to-one** merging: In a many to one merge (`m:1`), the working file has multiple observations per each unique key variable in the using file.
- **Many-to-many** merging: In a many to many merge (`m:m`), both the working and the using files have multiple observations per each unique key variable.

The structure of the command is as follows.

```
. merge 1:1 KeyVar using UsingFile
```

When the merging is succeeded, a system variable named `_merge` will be created. The `_merge` variable has five possible values, but in most cases, unless the using file is being used to update the master file, only the first three are of interest:
`_merge==1` observation found in master file only,
`_merge==2` observation found in using file only,
`_merge==3` observation built from both files. Normally, this is the desired value.

Example 4.2. In the folder given to you, again there are two data files, DataForMerge_1 and DataForMerge_2 which have same observations (cases) but different variables. Let's add the variables from DataForMerge_2 to DataForMerge_1. After opening DataForMerge_1, just type:

```
. merge 1:1 ID using DataForMerge_2
```

If the merging procedure succeeded, Stata will give a frequency table for the system generated _merge variable as follows.

```
Result                           # of obs.
-----------------------------------------
not matched                             0
matched                                 5  (_merge==3)
-----------------------------------------
```

All observations are found in both the DataForMerge_1 and DataForMerge_2 data files.

If we want to merge another data file, the system generated _merge variable must be dropped or renamed.

Example 4.3. Let's now consider an example. In the JUSH_HAART, each patient has a unique card number. But, in the CD4Count, the card numbers are repeated because the CD4 was measured approximately every six months for a period of five years. Hence, each patient may have made many (1-10) records for CD4 counts, so there may be multiple observations for each card number.

Before trying to merge, both datasets need to be sorted by the patient's card number (CardNum). Then, after opening JUSH_HAART, the `merge 1:m` command is used to join variables from the CD4Count file by CardNum. That is,

```
. merge 1:m CardNum using CD4Count
```

If the merging procedure succeeded, Stata will give a frequency table for the system generated _merge variable as follows.

```
Result                           # of obs.
-----------------------------------------
not matched                            34
    from master                         7  (_merge==1)
    from using                         27  (_merge==2)

matched                             4,628  (_merge==3)
-----------------------------------------
```

This indicates there are 7 observations that are found in the master file only (but not in the using file) and there are 27 observations that are found in the using file (but not in the master file). And 4,628 observations with same CardNum are found in both data files.

If we want to merge another data file, the system generated _merge variable must be dropped or renamed. Otherwise, the merging procedure will not work because the _merge variable

already exists.

Note also that two options are available with the `merge` command; `update` and `replace`. The `update` option replaces missing values in the master file with values from the using file. The `replace` option, which is used in conjunction with `update`, replaces missing and non-missing values in the master file with values found in the using file.

Chapter 5

Saving Command and Output Files

5.1 The Do-File Editor Window

A do-file editor window (batch mode) is used to type a series of Stata commands and save it in the do-file (`.do`) format. It is like any text editor having basic features of any text editor: cutting, copying, pasting, and so on. The advantages of using do-files are to reproduce results exactly and quickly without re-typing those commands, continue analyses from the point we left off and to recall reasoning.

To open the do-file editor, click on **Window** → **Do-file Editor** → **New do-file Editor** or click on the Do-file Editor icon which looks like paper and pencil.

Typing one command per line is the same as that would have done in the interactive mode (command window). A single comment in a line can begin with an "$*$" or a "//". The "//" can also be used to include comments on the same line as a command. Multiple lines of comments should be inclosed by a "/$*$" at the beginning and $*$/ at the ending.

If there is a need of breaking up a long command to more than one line, three slashes (///) can be used as a continuation symbol (this works in the Do-editor only, not on the command line).

To execute all the commands, select **Tools** → **Execute (do)**. If we want to execute only some of the commands in the Do-file editor, we have to select those commands we want.

Note that the **Save** and **Open** file menu selections in the Do-file editor window can only be used to save and open do-files.

5.2 The Log-File

As explained above, do-files are the best way to document our work because the results can always be replicated, and they serve as documentation of our Stata session. A way to document our entire Stata output, whether we work in interactive or batch (do-file) mode, is the log-file.

All output appearing in the *Results* window can be can be captured in a log file. The log file can be saved, retaining all the formatting as seen in the *Results* window, as a Stata Markup and Control Language (SMCL) format (`.smcl`). The `.smcl` file is readable only in Stata and it cannot be edited.

The output can also be save in a text file format (`.log`). The `.log` is a plain text file that can be opened for editing or printing in any text editor or word processor.

To start an `.smcl` log, use

```
. log using FileName
```

To overwrite Filename.smcl log, use

```
. log using FileName ,replace
```

To start a text log, use

```
. log using FileName.log
```

To pause a log, type `log off` which temporarily suspends log file. Also to resume a log file, type `log on`. These commands can be useful to create a log that contains only results and not intermediate programming. To close the current log file `log close`.

To translate a log file created in `.smcl` to text, go to **File** → **Log** → **Translate**.

Chapter 6

Descriptive Statistics Using Stata

6.1 JUSH HAART Data to be used

Exercise 6.1. In the folder given to you, there are two datasets in excel file named JUSH_HAART and CD4Count. Both datasets were obtained from Jimma University Specialized Hospital - HIV/AIDS Outpatient Clinic, Ethiopia. The JUSH_HAART contains the baseline characteristics of 1464 HIV/AIDS patients who were 18 years old or older and who were under HAART treatment between 2007 and 2011 in Jimma University Specialized Hospital. Where as the CD4Count data contains the number of CD4 counts (per mm^3 of blood) of the same patients. The CD4 counts were measured approximately every 6 months; at the study entry, and again at the 6-, 12-, 18-, 24-, 30-, 36-, 42-, 48- and 54-month visits.

In most of the illustrations in this training, we'll use this data. The descriptions of the variables are given below.

6.2 Frequency Tables

For a single qualitative variable, the data can summarized by counting the number of observations (frequency) in each category. The sample proportions in the categories estimate the category probabilities. For two or more categorical variables, the data is summarized in a tabular form in which the cells of the table contain number of observations (frequencies) in the intersection categories of the variables.

Now, let's start with the simplest of analytical commands and create frequency tables. Stata can produce one-way and two-way frequency tables which are useful for categorical variables.

6.2.1 One-Way Tables: The tabulate and tab1 Commands

The Stata command tabulate creates frequency table. To begin, let's look at the reported distribution of patients' education level (EducLev) using the tabulate command.

```
. tabulate EducLev

  Education |
      Level |      Freq.     Percent        Cum.
```

Table 6.1: Variable descriptions of the JUSH_HAART data

Variable Name	Variable Label	Value Label
CardNum	Patient's Card Number	
Age	Age in Years	
Sex	Sex	
Wei	Weight in Kilograms	
MarStat	Marital Status	0=Never Married, 1=Married, 2=Divorced, 3=Separated, 4=Widowed
EducLev	Education Level	0=No Education, 1=Primary, 2=Secondary, 3=Tertiary
Emp	Employment Condition	0=Full-time, 1=Part-time, 2=Not Working, 3=Unemployed
ClinStag	Clinical Stage	1=Stage I, 2=Stage II, 3=Stage III, 4=Stage IV
FunStat	Functional Status	0=Working, 1=Ambulatory, 2=Bedridden
CD4	Number of CD4 Counts	
Status	Survival Outcome	0=Active, 1=Dead, 2=Transferred, 3=Lost-to-follow
Defaulter	Dropped Out Patient	0=Active, 1=Defaulted
SurvTime	Survival Time (Months)	

```
-------------+-----------------------------------
No Education |        297       20.36       20.36
     Primary |        515       35.30       55.65
   Secondary |        492       33.72       89.38
    Tertiary |        155       10.62      100.00
-------------+-----------------------------------
       Total |      1,459      100.00
```

Now that's odd, isn't it? We know from our previous `codebook` command that we should expect 1,464 responses. We're short here by a number of responses. In fact, let's use Stata to calculate the number of responses by which we're actually short:

```
. display 1464 - 1459
5
```

After using the Stata's built-in calculator, we can tell that we've 5 observations not included in the table. What happened to them? By default, Stata leaves out all missing observations. If the `missing` option, which forces Stata to include the missing responses, is specified, missing values will be included in the frequency counts as shown below.

```
. tabulate EducLev ,missing

   Education |
```

```
       Level |      Freq.     Percent        Cum.
-------------+-----------------------------------
No Education |         297       20.29       20.29
     Primary |         515       35.18       55.46
   Secondary |         492       33.61       89.07
    Tertiary |         155       10.59       99.66
           . |           5        0.34      100.00
-------------+-----------------------------------
       Total |       1,464      100.00
```

Recall from above that missing observations are shown with a period (.). And, moreover, the 5 missing observations is exactly the same number as we got from the `display` command. Nice.

To ask for more than one frequency table in a single command, the `tab1` command is used. It produces one-way frequency table for each variable in the variable list.

```
. tab1 Sex MarStat ,missing

-> tabulation of Sex

        Sex |      Freq.     Percent        Cum.
------------+-----------------------------------
          F |         930       63.52       63.52
          M |         534       36.48      100.00
------------+-----------------------------------
      Total |       1,464      100.00

-> tabulation of MarStat

      Marital |
       Status |      Freq.     Percent        Cum.
--------------+-----------------------------------
Never Married |         293       20.01       20.01
      Married |         739       50.48       70.49
     Divorced |         134        9.15       79.64
    Separated |         140        9.56       89.21
      Widowed |         154       10.52       99.73
            . |           4        0.27      100.00
--------------+-----------------------------------
        Total |       1,464      100.00
```

6.2.2 Two-Way Tables: The tabulate and tab2 Commands

For two or more categorical variables, the data can be summarized in a tabular form in which the cells of the table contain number of observations (frequencies) in the intersection categories of the variables. Such a table is called contingency table (cross-tabulation). Two-way frequency tables (contingency tables) are useful for determining the association between two

categorical variables.

For example, using the JUSH_HAART data, in order to determine how the survival outcome (Status) varies by patient functional status (FunStat) type in:

```
. tabulate FunStat Status
```

```
Functional |              Survival Outcome
    Status |    Active       Dead  Transferr  Lost-to-f |     Total
-----------+-------------------------------------------+----------
   Working |       815         20         58        110 |     1,003
Ambulatory |       287         18         47         52 |       404
 Bedridden |        31          7         10          9 |        57
-----------+-------------------------------------------+----------
     Total |     1,133         45        115        171 |     1,464
```

Clearly, the table shows that there are 815 patients who were active and working, 287 patients who were ambulatory and active, and 31 patients who were bedridden and active. Also, of the 20 dead patients; 20, 18 and 7 of them were working, ambulatory and bedridden, respectively. The same description applies to the other frequencies.

But, I have one simple but critical question! Of the three functional status categories (Working, Ambulatory and Bedridden, which group was more likely to be dead? If your answer is Working, surprisingly you're wrong. Why?

To answer this question, we'd like to know about the proportions by functional status of the patient. How might we do that? In the two-way tables command, several options can be used. Of these, `row` gives row percentages and `col` gives column percentages. And, `cell` gives the overall percentage and `expected` reports the expected frequency in each cell. In addition, the `nofreq` option suppresses printing the frequencies while the `nolabel` option suppresses the use of value labels, showing the numeric values instead.

Hence, to know the proportions by functional status of the patient, we can add the `row` option in the `tabulate` command.

```
. tabulate FunStat Status ,row
```

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

Functional |              Survival Outcome
    Status |    Active       Dead  Transferr  Lost-to-f |     Total
-----------+-------------------------------------------+----------
   Working |       815         20         58        110 |     1,003
```

```
           |     81.26       1.99       5.78      10.97 |    100.00
-----------+----------------------------------------------+----------
Ambulatory |       287         18         47         52 |       404
           |     71.04       4.46      11.63      12.87 |    100.00
-----------+----------------------------------------------+----------
 Bedridden |        31          7         10          9 |        57
           |     54.39      12.28      17.54      15.79 |    100.00
-----------+----------------------------------------------+----------
     Total |     1,133         45        115        171 |     1,464
           |     77.39       3.07       7.86      11.68 |    100.00
```

Therefore, now, it is obvious that bedridden patients are more likely to be dead.

To ask for more than one two-way frequency table in a single command, the `tab2` command is used. This command produces all possible two-variable tables from the list of variables. For example, the command

```
. tab2 FunStat Status ClinStag
```

produces three two-way tables.

6.3 Summary Statistics

Measures of Central Tendency

Typical values which tend to lie centrally within a set of observations when arranged according to magnitudes are called *measures of central tendency*. The common measures of central tenderly are mean, median and mode.

- Mean is the usual average.
- Median is a value that divides the dataset in two equal parts.
- Mode is a value with the highest frequency.

Measures of Variation

The following table displays the price of a certain commodity in four cities. Find the mean and median prices of the four cities and interpret it.

A	30	30	30	30	30
B	28	29	30	31	32
C	10	15	30	45	50
D	0	5	30	55	60

All the four datasets have mean 30 and median is also 30. But by inspection it is apparent that the four datasets differ remarkably from one another. So measures of central tendency alone do not provide enough information about the nature of the data. Thus, to have a clear picture of the data, one needs to have a measure of dispersion or variability among

observations in the dataset.

Variation or *dispersion* may be defined as the extent of scatteredness of value around the measures of central tendency. Thus, a measure of dispersion tells us the extent to which the values of a variable vary about the measure of central tendency.

The common measures of variation are variance and standard deviation, which explain the average amount of variation on either sides of the mean.

For a population containing N elements, the population variance is denoted by the square of the Greek letter σ (sigma), σ^2:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(X_i - \mu)^2$$

For a sample of n elements, the sample variance denoted by S^2 and calculated using the formulae:

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2.$$

Standard deviation is the positive square root of variance. If the standard deviation of the data is small, the values are concentrated near the mean and if it large, the values are scattered away from the mean.

6.3.1 Summary Statistics: The summarize Command

The summarize command provides the number of valid observations, mean, standard deviation, as well as the minimum and maximum for each numeric variable.

Now try to type the summarize command and examine the results.

```
. summarize

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     CardNum |      1464    2551.764     658.542      1202       3587
         Age |      1464    34.01161    9.160258        18         85
         Sex |         0
         Wei |      1460    51.86788    10.19338        16         96
     MarStat |      1460    1.399315    1.211154         0          4
-------------+--------------------------------------------------------
     EducLev |      1459    1.346127    .9200576         0          3
         Emp |      1444    1.850416    1.375886         0          3
    ClinStag |      1464    2.247951     .898466         1          4
     FunStat |      1464    .3538251    .5538152         0          2
         CD4 |      1464    198.1906    171.2403         1       1914
-------------+--------------------------------------------------------
      Status |      1464    .5382514    1.052244         0          3
   Defaulter |      1464    .2260929    .4184429         0          1
    SurvTime |      1464    25.19768    16.70695   .0333333         54
```

Two notes to be considered here. First, note that the string variable Sex has no information listed, not even the number of observations. This is, because, once a variable is declared as string, Stata does not apply any mathematical operation or statistical analysis.

Second, note that even if the mean and standard deviation of the those categorical variables, which are coded as numeric, are calculated, such values are meaningless. Hence, the reported mean and standard deviations for MarStat, EducLev, Emp, ClinStag, FunStat, Status, Defaulter as well as CardNum have no meaning at all. Therefore, keep in mind that the summarize command is appropriate only for quantitative variables.

The detail option in the summarize command gives additional statistics (skewness, kurtosis, the four smallest values, the four largest values, various percentiles) for the specified variables.

```
. summarize Wei ,detail

                    Weight in Kilograms
-------------------------------------------------------------
      Percentiles      Smallest
 1%           31             16
 5%        36.75             18
10%           40             26       Obs                1460
25%           45             27       Sum of Wgt.        1460

50%           51                      Mean           51.86788
                       Largest        Std. Dev.      10.19338
75%        57.75             90
90%           65             92       Variance       103.9049
95%           70             96       Skewness       .5189258
99%           80             96       Kurtosis       3.901936
```

6.3.2 Compact Table of Summary Statistics: The tabstat Command

The tabstat command displays summary statistics for a series of numeric variables in one table. The command by default gives the mean but allows us to specify a list of descriptive statistics with the option statistics (mean, median, min, max, range, sd, variance, skewness, kurtosis, percentiles: p1, p2, $\cdots$, p99).

```
. tabstat Age Wei CD4 SurvTime ,stat(n mean min p25 p50 p75 max)

   stats |       Age       Wei       CD4  SurvTime
---------+----------------------------------------
       N |      1464      1460      1464      1464
    mean |  34.01161  51.86788  198.1906  25.19768
     min |        18        16         1  .0333333
     p25 |        28        45        85  10.83333
     p50 |        32        51       158     23.95
     p75 |        39     57.75       266     39.05
```

```
     max |        85        96      1914        54
-------------------------------------------------
```

Like the summarize command, the tabstat results are valid only for quantitative variables. That is why, in the above command, we have listed quantitative variables only.

As shown above output, the tabstat command displays the variables in columns. But, it is helpful to put statistics in columns. The columns(statistics) is used to do so, instead of the default option columns(variables).

```
. tabstat Age Wei CD4 SurvTime ,stat(n min max mean sd p50) col(stat)
```

The output is:

```
    variable |         N       min       max      mean        sd       p50
-------------+--------------------------------------------------------------
         Age |      1464        18        85  34.01161  9.160258        32
         Wei |      1460        16        96  51.86788  10.19338        51
         CD4 |      1464         1      1914  198.1906  171.2403       158
    SurvTime |      1464  .0333333        54  25.19768  16.70695     23.95
-------------------------------------------------------------------------
```

The tabstat command also provides summary statistics broken down on (conditioned by) another variable using the by() option. For example, to obtain the above descriptive statistics for male and female, we can do as follows.

```
. tabstat Age Wei CD4 SurvTime ,stat(n min max mean sd p50) col(stat) by(Sex)
```

Then Stata provides the result:

```
Summary for variables: Age Wei CD4 SurvTime
     by categories of: Sex (Sex of the Patient)

   Sex |         N       min       max      mean        sd       p50
-------+--------------------------------------------------------------
     F |       930        18        85  32.01828  8.600241        30
       |       927        16        96  49.68285  9.916851        49
       |       930         2      1914  209.7118  178.1555       173
       |       930        .1        54  25.81667  16.63067     24.55
-------+--------------------------------------------------------------
     M |       534        18        79  37.48315  9.077253        36
       |       533        29        92  55.66811  9.538948        55
       |       534         1      1352  178.1255   156.627       140
       |       534  .0333333        54  24.11966  16.80015      23.3
-------+--------------------------------------------------------------
 Total |      1464        18        85  34.01161  9.160258        32
       |      1460        16        96  51.86788  10.19338        51
       |      1464         1      1914  198.1906  171.2403       158
       |      1464  .0333333        54  25.19768  16.70695     23.95
-----------------------------------------------------------------------
```

Chapter 7

Hypothesis Testing

7.1 Statistical Inference

The primary objective of a statistical analysis is drawing statistically valid conclusions about the characteristics of the population based on the results obtained from sample. There are two important facts that are key to statistical inference. These are (population) *parameter* and (sample) *statistic*. A *parameter* is a fixed (but usually unknown) summary measure of the characteristic of a population. For example, population mean (μ), population variance (σ^2), population proportion (π) are parameters.

Statistical inference can be defined as the process of making conclusions for population parameters using sample statistics. It generally takes two forms, namely, *estimation* of a parameter and *testing of a hypothesis*.

7.1.1 Estimation

For the purpose of general discussion, let θ be the population parameter and $\hat{\theta}$ be the corresponding statistic. The statistic $\hat{\theta}$ intended for estimating a parameter θ is called an *estimator* of θ. A specific numerical value of an estimator calculated from the sample is called the *estimate*. The process of obtaining an estimate of the unknown value of a parameter by a statistic is called *estimation*. There are two types of estimations. One is *point* estimation and the other is *interval* estimation.

Point Estimation

Point estimation is the process of obtaining a *single* sample value (point estimate) that is used to estimate the desired population parameter. The estimator is known as point estimator. For example, $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ is a point estimator of μ, $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$ is a point estimator of σ^2.

Interval Estimation

Point estimator has some drawbacks. First, a point estimator from the sample *may not exactly locate the population parameter*, that is, the value of point estimator *is not likely to be exactly equal to the value of the parameter*, resulting in some margin of uncertainty. If the sample value is different from the population value, the point estimator *does not indicate the extent of the possible error*. Second, a point estimate *does not specify as to how confident we can be that the estimate is close to the parameter* it is estimating. That is, we *cannot attach any degree of confidence* to such an estimate as to what extent it is closer to the value of the parameter. Because of these limitations of point estimation, interval estimation is considered desirable. *Interval estimation* involves the determination of an *interval (a range of values)* within which the population parameter must lie with a *specified degree of confidence*. It is the *construction of an interval on both sides of the point estimate* within which wan reasonably confident that the true parameter will lie.

7.1.2 Hypothesis Testing

A statistical hypothesis is an assumption (a conjecture) about a population parameter. Such an assumption usually results from speculation concerning observed behavior, natural phenomena, or established theory. Hence, hypothesis testing is a statistical procedure which leads to take a decision about a statistical hypothesis for being supported or not by the sample data. It starts by making a set of two mutually-exclusive and exhaustive hypotheses about the parameter(s) in question.

The first hypothesis is called a *null* hypothesis (denoted by H_0) which states there is no difference between a parameter and a hypothesized value. For any parameter θ and an assumed value θ_0, the null hypothesis is written as $H_0 : \theta = \theta_0$.

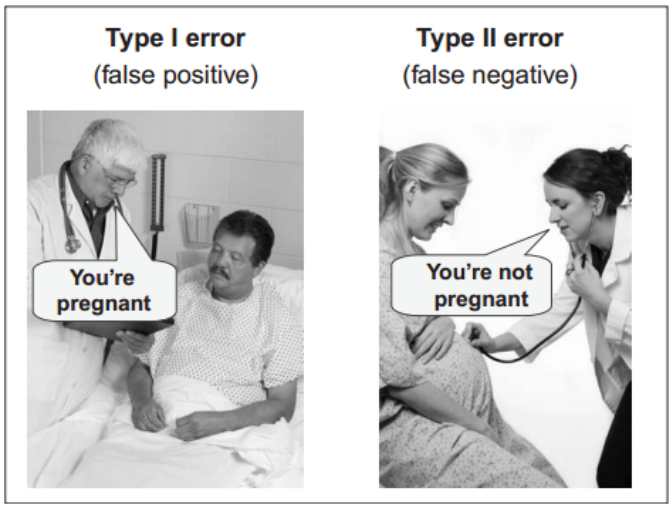
The second hypothesis, is called an *alternative* hypothesis (denoted by H_1), contradicts the null hypothesis, that is, it states there is a difference between a parameter and a hypothesized value. This hypothesis may have three different forms:

- Two-sided test: $H_1 : \ \theta \neq \theta_0$.
- One-sided test: $H_1 : \ \theta > \theta_0$ (right tailed test)
- One-sided test: $H_1 : \ \theta < \theta_0$ (left tailed test)

Types of Errors in Hypothesis Testing

There are two types of errors in hypothesis testing.

- **Type I Error**: It is an error occurred if one rejects the null hypothesis which is actually true. The probability of making such an error is called *significance level* (denoted by α).
- **Type II Error**: It is an error occurred if one failed to reject the null hypothesis which is actually false. The probability of making type II error is denoted by β. The probability of correctly rejecting the null hypothesis which is actually false, called *power* of a test, is, therefore, $1 - \beta$.



In statistical hypothesis testing, the maximum acceptable probability of rejecting a true null hypothesis, the significance level (α), is specified first.

Steps in Testing a Hypothesis

A statistical hypothesis test can be formally summarized as a five-step process. These are:

- **Step 1**: State both null and alternative hypotheses, that is, H_0 and H_1.
- **Step 2**: Specify the maximum acceptable level of significance (α) and obtain the critical value (T_{tab}). The most common choices of significance levels are $\alpha = 10\%$, $\alpha = 5\%$ and $\alpha = 1\%$. For a one sided test, $T_{tab} = T_{\alpha}$ and for a two sided test, $T_{tab} = T_{\alpha/2}$. This critical value is used to define the *rejection* (critical) region of the null hypothesis (H_0).
- **Step 3**: Define a test statistic and find its calculated value (T_{cal}).
- **Step 4**: Make a decision about H_0, that is, reject H_0 if $|T_{cal}| \geq T_{tab}$, otherwise do not reject H_0.
- **Step 5**: Put a conclusion.

7.2 Testing about a Population Mean: The `ttest` Command

A one-sample t-test helps to determine whether the population mean (μ) is equal to a hypothesized value (μ_0). The underlying assumption of the t-test is that the observations are random samples drawn from normally distributed populations.

Steps:

1. The null hypothesis to be tested is $H_0 : \mu = \mu_0$ and the alternative hypothesis can be $H_1 : \mu \neq \mu_0$, $H_1 : \mu < \mu_0$ or $H_1 : \mu > \mu_0$.
2. Choose a level of significance (α): common choices are 0.01, 0.05 and 0.10.

3. The test statistic is: $t = \dfrac{\bar{y} - \mu_0}{s/\sqrt{n}}$ where $\bar{y} = \dfrac{1}{n}\sum_{i=1}^{n} y_i$ is the sample mean, $s^2 = \dfrac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2$ is the sample variance (hence s is the sample standard deviation), n is the sample size and μ_0 is the assumed value. The test statistic has a t distribution with $n-1$ degrees of freedom.

4. Decision:

   - For a two sided test, H_0 is rejected if $|t| > t_{\alpha/2}(n-1)$.
   - For a one sided case, H_0 is rejected if $|t| > t_{\alpha}(n-1)$.

   In both cases, if the $p-$value is less than the specified α, H_0 should be rejected otherwise do not.

5. Conclusion.

In Stata, the command `ttest` is used for a one sample test. The command structure is:

```
. ttest VarName=mu0
```

If we want to change the default confidence level (95%), we can add the option `level(99)`, for example, to make the 99% confidence level.

Example 7.1. The thermostat in the classroom is set at 72°F, but we think the thermostat is not working well. On seven randomly selected days, we measure the temperature (in degrees Fahrenheit) as 71, 73, 69, 68, 69, 70, and 71. Let's test whether the mean temperature is different from 72°F. First, enter the data in Stata's Data Editor under the variable name Temp.

```
. list

     +------+
     | Temp |
     |------|
  1. |   71 |
  2. |   73 |
  3. |   69 |
  4. |   68 |
  5. |   69 |
     |------|
  6. |   70 |
  7. |   71 |
     +------+
```

Here, we want to test $H_0 : \mu = 72°F$ vs $H_1 : \mu \neq 72°F$.

```
. ttest Temp=72 ,level(95)
```

Note that the default confidence level, `level(95)`, can be omitted or changed. After executing this command or pressing the **Enter** key, the output looks the following.

```
One-sample t test
------------------------------------------------------------------------------
Variable |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
    Temp |       7    70.14286    .6335302    1.676163    68.59266    71.69305
------------------------------------------------------------------------------
    mean = mean(Temp)                                             t =  -2.9314
Ho: mean = 72                                     degrees of freedom =        6

    Ha: mean < 72                 Ha: mean != 72                 Ha: mean > 72
 Pr(T < t) = 0.0131         Pr(|T| > |t|) = 0.0262          Pr(T > t) = 0.9869
```

The $t-$statistics and the $p-$value of the $t-$statistics are automatically provided for both the one and two-sided alternatives. If we want to test one-sided, we'll need to divide the $p-$value of the two-sided test by 2 - AND check that the sign is in the expected direction!!

Now from the above output, we can see that the $p-$value $= 0.0262$ which is less than $\alpha = 0.05$. Hence, we should reject the null hypothesis and conclude that the average temperature is not $72°F$. In particular, the average temperature is less than $72°F$.

The Stata command is `ttesti` can also be used if the summary statistics are given as n $\bar{y}$ s μ_0.

```
. ttesti 7 70.14286 1.676163 72
```

Note that the $p-$values here may differ slightly from the previous output if we have rounded the sample statistics in the `ttesti` command above.

7.3 Comparing Two Population Means

7.3.1 Paired Samples: The `ttest` Command

For two paired variables, the difference of the two variables, $d_i = Y_{1i} - Y_{2i}$, is treated as if it were a single sample. This test is appropriate for pre-post treatment responses. The null hypothesis is that the true mean difference of the two variables is D_0, $H_0 : \mu_d = D_0$. The difference is typically assumed to be zero unless explicitly specified.

Steps:

1. The null hypothesis to be tested is $H_0 : \mu_d = 0$ and the alternative hypothesis may be $H_1 : \mu_d \neq 0$, $H_1 : \mu_d < 0$ or $H_0 : \mu_d > 0$.
2. Choose a level of significance (α)
3. The test statistic is: $t = \dfrac{\bar{d} - \mu_d}{s_d/\sqrt{n}} \sim t(n-1)$ where $\bar{d} = \dfrac{1}{n}\sum_{i=1}^{n} d_i$ is the sample mean of the differences, $s_d^2 = \dfrac{1}{n-1}\sum_{i=1}^{n}(d_i - \bar{d})^2$ is the sample variance of the differences and n is the sample size. This test statistic has a t distribution with $n-1$ degrees of freedom.

4. Decision:

   - For a two sided test, H_0 is rejected if $|t| > t_{\alpha/2}(n-1)$.
   - For a one sided case, H_0 is rejected if $|t| > t_{\alpha}(n-1)$.

   In both cases, if the $p-$value is less than the specified α, H_0 should be rejected otherwise do not.

5. Conclusion.

In Stata, the command for paired test is:

```
. ttest Pre_Response=Post_Response ,level(95)
```

Example 7.2. A researcher is interested in investigating whether alcohol has a positive or negative effect on the heart beat of individuals. S/he has measured the heart beat (per minute) of six persons before and after drinking Alcohol. The data is:

Before Drinking Alcohol	86	90	75	72	78	68
After Drinking Alcohol	97	96	80	76	77	73

Let's test the hypothesis using Stata. Enter these data, naming the first variable of the pair Before and the second After. Then, the `list` command displays as follows.

```
     +----------------+
     | Before   After |
     |----------------|
  1. |     86      97 |
  2. |     90      96 |
  3. |     75      80 |
  4. |     72      76 |
  5. |     78      77 |
     |----------------|
  6. |     68      73 |
     +----------------+
```

Then

```
. ttest Before=After

Paired t test
------------------------------------------------------------------------------
Variable |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
  Before |       6    78.16667    3.429448    8.400397    69.35099    86.98234
   After |       6    83.16667    4.315991    10.57198    72.07206    94.26127
---------+--------------------------------------------------------------------
    diff |       6          -5    1.570563    3.847077    -9.03726   -.9627405
------------------------------------------------------------------------------
     mean(diff) = mean(Before - After)                             t =  -3.1836
 Ho: mean(diff) = 0                         degrees of freedom =        5
```

```
Ha: mean(diff) < 0          Ha: mean(diff) != 0          Ha: mean(diff) > 0
Pr(T < t) = 0.0122        Pr(|T| > |t|) = 0.0244         Pr(T > t) = 0.9878
```

From the above results, we can conclude that alcohol has an increasing effect in the heart beat of individuals.

Alternatively, we may first compute the difference between the two variables, and then conduct one-sample t-test. For instance, for the above example, we can do

```
. generate Diff=Before-After
. ttest Diff=0
```

7.3.2 Independent Samples: The `ttest` and `sdtest` Commands

Comparing Two Population Means

1. The null hypothesis to be tested is $H_0 : \mu_1 = \mu_2$ and the alternative hypothesis may be $H_1 : \mu_1 \neq \mu_2$, $H_1 : \mu_1 < \mu_2$ or $H_1 : \mu_1 > \mu_2$.

2. Choose a level of significance (α).

3. The test statistic is: $t = \dfrac{(\bar{y}_1 - \bar{y}_2) - (\mu_1 - \mu_2)}{s_p\sqrt{\dfrac{1}{n_1} + \dfrac{1}{n_2}}}$ where $\bar{y}_1 = \dfrac{1}{n_1}\sum_{i=1}^{n} y_{1i}$ is the sample mean of the first group and $\bar{y}_2 = \dfrac{1}{n_2}\sum_{i=1}^{n} y_{2i}$ is the sample mean of the second group, $s_p^2 = \dfrac{(n_1-1)s_1^2 + (n_2-1)s_2^2}{n_1+n_2-2}$ is the pooled variance of the both groups (note $s_1^2 = \dfrac{1}{n_1-1}\sum_{i=1}^{n}(y_{1i} - \bar{y}_1)^2$ is the sample variance of the first group and $s_2^2 = \dfrac{1}{n_2-1}\sum_{i=1}^{n}(y_{2i} - \bar{y}_2)^2$ is the sample variance of the second group), n_1 is sample size of the first group and n_2 is sample size of the second group. The test statistic has a t distribution with $n_1 + n_2 - 2$ degrees of freedom.

4. Decision:

   - For a two sided test, H_0 is rejected if $|t| > t_{\alpha/2}(n_1 + n_2 - 2)$.
   - For a one sided case, H_0 is rejected if $|t| > t_{\alpha}(n_1 + n_2 - 2)$.

   In both cases, if the $p-$value is less than the specified α, H_0 should be rejected otherwise do not.

5. Conclusion.

The above test statistic is only used when the two distributions are assumed to have the same variance. If the two population variances are different, then they must be estimated

separately and the test statistic is a little bit modified as

$$t = \frac{(\bar{y} - \bar{y}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}.$$

This modified test, also known as Welch's t-test, has a t distribution with v degrees of freedom where

$$v = \frac{(s_1^2/n_1 + s_2^2/n_2)^2}{(s_1^2/n_1)^2/(n_1 - 1) + (s_2^2/n_2)^2/(n_2 - 1)}.$$

Note that the true distribution of the test statistic actually depends (slightly) on the two unknown variances. Therefore, to determine which test statistics to be used for comparing two population means, first the equality of variances should be checked.

Comparing Two Population Variances

1. The null and alternative hypotheses to be tested are:

$$H_0 : \sigma_1 = \sigma_2$$
$$H_1 : \sigma_1 \neq \sigma_2$$

2. Choose a level of significance (α).

3. The test statistic is: $F = \frac{s_1^2}{s_2^2}$ where $s_1^2 = \frac{1}{n_1 - 1}\sum_{i=1}^{n}(y_{1i} - \bar{y}_1)^2$ is the sample variance of the first group and $s_2^2 = \frac{1}{n_2 - 1}\sum_{i=1}^{n}(y_{2i} - \bar{y}_2)^2$ is the sample variance of the second group, n_1 is sample size of the first group and n_2 is sample size of the second group. This statistic has an F distribution with $n_1 - 1$ and $n_2 - 1$ degrees of freedom.

4. Decision: If $F > F_\alpha(n_1 - 1, n_2 - 1)$ or if the p-value is less than the specified α, then H_0 is rejected indicating that the common variance assumption does not hold.

5. Conclusion.

In Stata, the `sdtest` command is used for testing equality of variances. In the case of two independent populations, the data may be arranged in two different ways which both can be handled by Stata. In the first way, the groups may be in different columns, and in the second and most convenient way, the response variable may be stacked in one column and a groping variable will be in another column. In the first case, the `sdtest` command is used as:

```
. sdtest Group1=Group2
```

In the second case, that is, if the response is stacked and has a corresponding grouping variable as for example female and male or coded as 0 and 1, the `sdtest` command needs the `by(Group)` option to identify the grouping variable. That is,

```
. sdtest VarName ,by(Group)
```

Also with the reported summary statistics, i.e, n_1, $\bar{y}_1$, s_1, n_2, $\bar{y}_2$, s_2, we can use the sdtesti command.

```
. sdtesti n1 . s1 n2 . s2
```

If the two-sided $p-$value is larger than the specified α, then the assumption of equal variance holds.

Then, for comparing two population means, again we've to consider the arrangement the data. If the groups are in different columns, the usual `ttest` command with the `unpaired` option is used.

```
. ttest Var1=Var2 ,unpaired
```

But, if the response is stacked in one column and the grouping variable is in another column, the `ttest` command needs the by(Group) option:

```
. ttest VarName ,by(Group)
```

The `ttest` command for comparing two population means, by default, assumes equal variance. If we need to conduct the test assuming unequal variance, we've to add the option `unequal` to the `ttest` command.

Example 7.3. Company officials were concerned about the length of time a particular drug product retained its toxin's potency. A random sample of 8 bottles of the product was drawn from the production line and measured for potency. A second sample of 10 bottles was obtained and stored in a regulated environment for a period of one year. The readings obtained from each sample are given below.

Sample 1	10.2	10.5	10.3	10.8	9.8	10.6	10.7	10.2		
Sample 2	9.8	9.6	10.1	10.2	10.1	9.7	9.5	9.6	9.8	9.9

Using Stata, let's test the null hypothesis that the drug product retains its potency. First, let's stack the response (Potency) in one column and the grouping variable (Sample: 1 and 2) in another column. When using the `list` command, the data looks:

```
     +------------------+
     | Sample   Potency |
     |------------------|
  1. |      1      10.2 |
  2. |      1      10.5 |
  3. |      1      10.3 |
  4. |      1      10.8 |
  5. |      1       9.8 |
     |------------------|
  6. |      1      10.6 |
  7. |      1      10.7 |
  8. |      1      10.2 |
  9. |      2       9.8 |
 10. |      2       9.6 |
     |------------------|
```

```
 11. |      2      10.1 |
 12. |      2      10.2 |
 13. |      2      10.1 |
 14. |      2       9.7 |
 15. |      2       9.5 |
     |------------------|
 16. |      2       9.6 |
 17. |      2       9.8 |
 18. |      2       9.9 |
     +------------------+
```

Then, to compare the drug's potency between the two samples, first we've to compare the variability between the two samples (groups).

```
. sdtest Potency ,by(Sample)
```

```
Variance ratio test
------------------------------------------------------------------------------
   Group |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
       1 |       8     10.3875     .115631    .3270539    10.11408    10.66092
       2 |      10        9.83    .0760847    .2406011    9.657884    10.00212
---------+--------------------------------------------------------------------
combined |      18    10.07778    .0930793    .3949022    9.881398    10.27416
------------------------------------------------------------------------------
    ratio = sd(1) / sd(2)                                         f =   1.8478
Ho: ratio = 1                                 degrees of freedom =     7, 9

    Ha: ratio < 1               Ha: ratio != 1                 Ha: ratio > 1
  Pr(F < f) = 0.8076         2*Pr(F > f) = 0.3847           Pr(F > f) = 0.1924
```

or using the summary statistics:

```
. sdtesti 8 . 0.3270539 10 . 0.2406011
```

This result supports the assumption of equal variance. Thus, the command for the $t-$test for comparing the mean potency between the two samples assuming equal variance is:

```
. ttest Potency ,by(Sample)
```

By default, this test assumes equal variance. The output is:

```
Two-sample t test with equal variances
------------------------------------------------------------------------------
   Group |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
       1 |       8     10.3875     .115631    .3270539    10.11408    10.66092
       2 |      10        9.83    .0760847    .2406011    9.657884    10.00212
---------+--------------------------------------------------------------------
combined |      18    10.07778    .0930793    .3949022    9.881398    10.27416
```

```
---------+--------------------------------------------------------------------
    diff |               .5575    .1336258                .2742259    .8407741
------------------------------------------------------------------------------
    diff = mean(1) - mean(2)                                      t =   4.1721
Ho: diff = 0                                     degrees of freedom =       16

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.9996         Pr(|T| > |t|) = 0.0007          Pr(T > t) = 0.0004
```

or using the summary statistics

```
. ttesti 8 10.3875 0.3270539 10 9.83 0.2406011
```

Since the $p-$value is less than 0.05, we can conclude the mean potency in the first sample is larger than that of the second sample. In other words, storing the drug in a regulated environment for a period of one year reduces its potency.

Had we assume unequal variances, the above command needs only the `unequal` option:

```
. ttest Potency ,by(Sample) unequal
```

Exercise 7.1. A quick but impressive method of estimating the concentration of a chemical in a rat has been developed. The sample from this method has 8 observations and the sample from the standard method has 4 observations. Assuming different population variances, test whether the quick method gives under-estimate result. The data in the two samples are:

Standard Method	25	24	25	26				
Quick Method	23	18	22	28	17	25	19	16

7.4 Comparing Several Population Means: ANOVA

Despite its name, analysis of variance (ANOVA) is used to compare the means of more than two groups based on the variance ratio test. The principle underlying the ANOVA is that the total variability in a dataset is partitioned into its component parts. The sources of variation comprise one or more factors, each resulting in variability which can be accounted for (explained by the levels or categories of the factor), and also unexplained (residual) variation which results from uncontrolled biological variation and technical error.

Note that the null hypothesis is that the all group means are equal. That is, if there are g groups, then $H_0 : \mu_1 = \mu_2 = \cdots = \mu_g$. The alternative hypothesis is at least one of the means is significantly different from the other.

Assumptions of the one-way ANOVA

1. The samples are independently and randomly drawn from source population(s).
2. The source populations are reasonably normal distributions.
3. The samples have approximately equal variances.

If the samples are equal size, no main worry about these assumptions because one-way ANOVA is quite robust (relatively unperturbed by violations of its assumptions). But if the samples are different size and the assumption of equal variance does not hold, an appropriate non-parametric alternative for one-way ANOVA, which is called the Kruskal-Wallis test, to be discussed in section **??**, should be used.

Stata has `oneway` and `anova` command routines, either of which can be used for one-way analysis of variance. The `oneway` command is quicker than the `anova` command and allows us to perform multiple comparison tests. We'll use `oneway` now and the corresponding two-way ANOVA will show how to use the `anova` command.

7.4.1 Oneway ANOVA: The `oneway` Command

The basic syntax of the `oneway` command is:

```
. oneway Response Factor ,tabulate
```

The option `tabulate` is added to get descriptive statistics across the groups because the `oneway` command, by default, does not provide descriptive statistics per group.

Example 7.4. Suppose a university wishes to compare the effectiveness of four teaching methods (Slide, Self-Study, Lecture and Discussion) for a particular course. Twenty four students are randomly assigned to the teaching methods. At the end of teaching the students with their assigned method, a test (out of 20%) was given and the performance of the students were recorded as follows:

Slide	Self-Study	Lecture	Discussion
9	10	12	9
12	6	14	8
14	6	11	11
11	9	13	7
13	10	11	8
	5	16	6
			7

Let's examine whether there is any difference among the teaching methods. After entering the teaching Method in one column and the Score in other column of the Stata Data Editor, the `oneway` command can be used.

```
. oneway Score Method ,tabulate

   Teaching |          Summary of Score
     Method |        Mean   Std. Dev.       Freq.
------------+------------------------------------
      Slide |        11.8   1.9235384           5
  Self-Stud |   7.6666667   2.2509257           6
    Lecture |   12.833333   1.9407902           6
  Discussio |           8   1.6329932           7
```

```
------------+-----------------------------------
      Total |   9.9166667   2.9476102          24

                      Analysis of Variance
    Source              SS         df      MS            F     Prob > F
------------------------------------------------------------------------
Between groups      124.866667      3   41.6222222     11.10     0.0002
 Within groups      74.9666667     20   3.74833333
------------------------------------------------------------------------
    Total           199.833333     23    8.6884058

Bartlett's test for equal variances:  chi2(3) =   0.5193  Prob>chi2 = 0.915
```

Stata adds Bartlett's test for equal variances. As we'll recall, one of the assumptions of ANOVA is that the variances are the same across groups. The small value for Bartlett's statistic confirms that this assumption is not violated in these data, so the use of ANOVA is ok! The significant F value of 11.10 tells us that at least one treatment effect differs from zero, i.e., the means are not all equal.

However, the above result does not tell us where the differences are. To identify the differences in each pair of group means, different mean separation methods (multiple comparison tests) are available as options under the oneway command. Of these, the bonferroni, scheffe and sidak are the common ones.

```
. oneway Score Method ,bon

                    Comparison of Score by Teaching Method
                                (Bonferroni)
Row Mean-|
Col Mean |      Slide   Self-Stu    Lecture
---------+---------------------------------
Self-Stu |   -4.13333
         |      0.013
         |
 Lecture |    1.03333    5.16667
         |      1.000      0.001
         |
Discussi |       -3.8    .333333   -4.83333
         |      0.019      1.000      0.001
```

The result indicates that lecture and slide teaching methods better than the other two.

7.5 The χ^2 Test of Association

The χ^2 test is used for testing the independence of two categorical variables. The null hypothesis, H_0 :, states there is no statistical association between the two categorical variables. In Stata, the χ^2 test is found as an option (chi2) in the tabulate command. Also, the option

all, in general, provides a variety of tests and measures of association for two-way contingency tables.

Example 7.5. Let's check whether there is a statistical association between the functional status (FunStat) and survival outcome (Status) in the JUSH_HAART data.

```
. tab FunStat Status ,chi2
```

The result is:

```
Functional |              Survival Outcome
    Status |    Active       Dead  Transferr  Loss-to-f |     Total
-----------+-------------------------------------------+----------
   Working |       815         20         58        110 |     1,003
Ambulatory |       287         18         47         51 |       403
 Bedridden |        31          7         10          9 |        57
-----------+-------------------------------------------+----------
     Total |     1,133         45        115        170 |     1,463

          Pearson chi2(6) =  51.1775   Pr = 0.000
```

Since, the $p-$value is very small, we can conclude that there is an association between functional status and survival outcome.

Chapter 8

Correlation and Linear Regression

Correlation is a statistical tool desired towards measuring the degree of linear relationship (association) between quantitative variables. If the change in one variable affects the change in the other variable, then the variables are said to be correlated.

8.1 Scatter Plot: The `twoway scatter` Command

Correlation that involves only two variables is called simple correlation. The simplest way to present bivariate data is to plot the values (x_i, y_i), $i = 1, 2, \cdots, n$ on the xy plane. This is known as *scatter plot*. This gives an idea about the correlation of the two variables. But, it will give only a vague idea about the presence and absence of correlation and the nature (direct or inverse) of correlation. It will not indicate about the strength or degree of relationship between two variables.

Example 8.1. A researcher wants to find out if there is a relationship between the heights of sons with the heights and weights of fathers. In other words, do taller fathers have taller sons? The researcher took a random sample of 8 fathers and their 8 sons. Their height in inches and the weight of fathers in kilograms are given below.

Son Height (SonH)	66	68	65	67	69	70	71	60
Father Height (FathH)	65	67	66	67	68	69	69	62
Father Weight (FathW)	67	66	52	66	69	64	80	50

Figure 8.1 and figure 8.2 show the scatter plot of sons' height and fathers' height, and the scatter plot of sons' height and fathers' weight, respectively. In addition, the scatter plot of fathers' height and fathers' weight is shown in figure 8.3.

```
. twoway (scatter SonH FathH) (lfit SonH FathH)
```

```
. twoway (scatter SonH FathW) (lfit SonH FathW)
```

```
. twoway (scatter FathH FathW) (lfit FathH FathW)
```

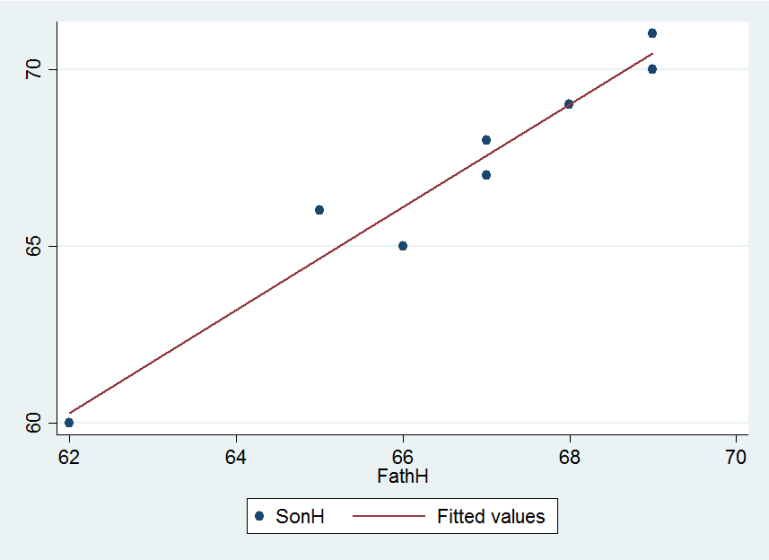


Figure 8.1: Scatter plot of sons' height and fathers' height

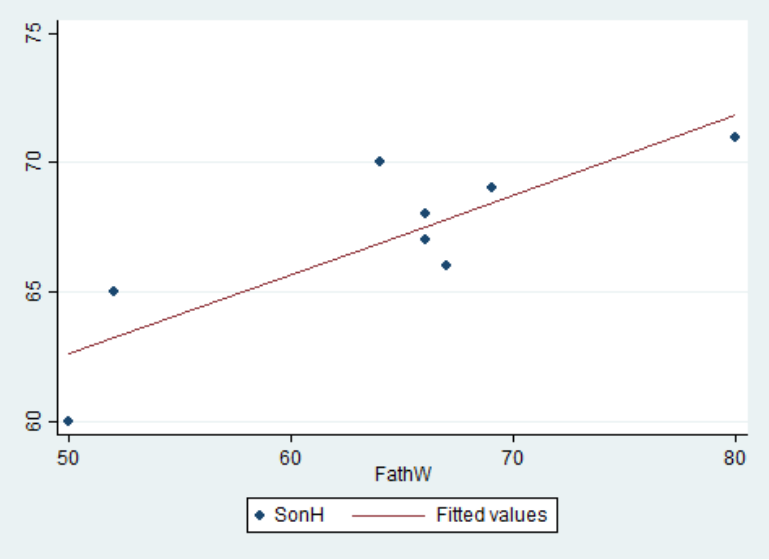


Figure 8.2: Scatter plot of sons' height and fathers' weight

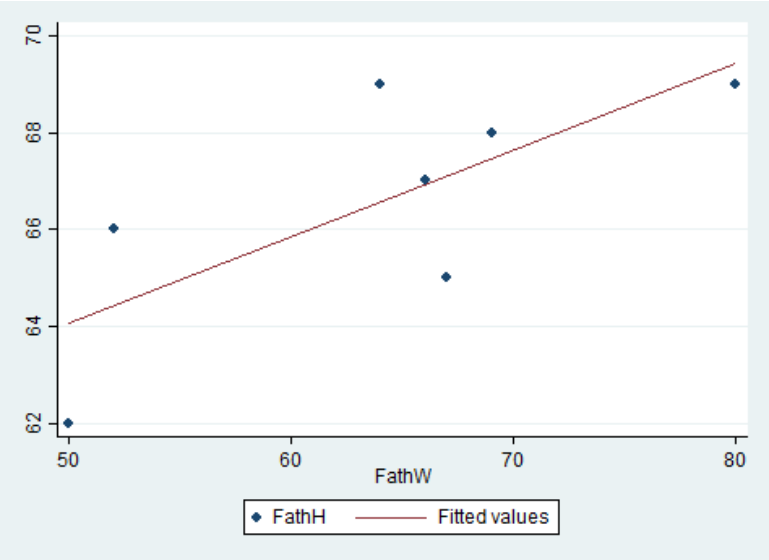


Figure 8.3: Scatter plot of fathers' height and fathers' weight

8.2 Covariance and Correlation: The `correlate` Command

Covariance is a measure of the joint variation between between two variables, i.e., it measures the way in which the values of the two variables vary together. Recall the sample covariance between two variables is defined as:

$$S_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}).$$

If the covariance is zero, there is no linear relationship between the two variables. Positive covariance indicates there is a direct linear relationship between the variables while negative covariance implies an inverse linear relationship between them.

The coefficient of correlation is a measure of the degree or strength of the linear association between two variables. It is defined as a ratio of the covariance between the two variables and the product of the standard deviations of the two variables. The sample correlation coefficient is denoted by r and the population correlation coefficient is denoted by the Greek letter ρ, rho.

$$r = \frac{S_{xy}}{S_x S_y}$$

The value of the correlation coefficient lies between the limits -1 and +1; that is $-1 \leq r \leq 1$.

- If the value of r is approximately -1 or +1, there is a strong inverse (indirect) or positive (direct) linear relationship between the variables, respectively.
- If the value of r is approximately -0.5 or +0.5, there is a medium inverse (indirect) or positive (direct) linear relationship between the variables, respectively.
- If the value of r is near zero, there is no linear association between the two variables.

Although, the sign of the sample correlation and sample covariance are the same, the correlation is ordinarily easier to interpret as:

- its magnitude is bounded, that is, $-1 \leq r \leq 1$.
- it is unitless.
- it takes the variability into account.

But the major disadvantage of correlation is it does not measure non-linear associations.

The `correlate` command displays the correlation matrix for a group of variables (the option `covariance` can be added to obtain the covariance matrix). If no variable is specified, the matrix is displayed for all variables in the data.

Example 8.2. Using the data given on example 8.1, let's perform correlation analysis of among sons' height, fathers' height and fathers' weight.

```
. correlate SonH FathH FathW

(obs=8)

             |     SonH    FathH    FathW
-------------+---------------------------
        SonH |   1.0000
       FathH |   0.9751   1.0000
       FathW |   0.8427   0.7320   1.0000
```

To obtain the covariance,

```
. correlate SonH FathH FathW ,cov

(obs=8)

             |     SonH    FathH    FathW
-------------+---------------------------
        SonH |       12
       FathH |  7.85714  5.41071
       FathW |  27.8571    16.25  91.0714
```

In addition, the `pwcorr` command with the option `sig` can also be used to test all the pairwise correlation coefficients between the specified variables.

8.3 Regression Analysis: The `regress` Command

Regression may be defined as the estimation of the unknown value of one variable from the known values of one or more variables. The variable whose values are to be estimated is known as *dependent* (*response*) variable while the variables which are used in determining the value of the dependent variable are called *explanatory* (*factor*) variables.

A *linear regression model* is a linear function of the explanatory variable(s) that gives the best estimate of the response variable for any given value(s) of explanatory variable(s).

Model: $y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki} + \varepsilon_i;\ i = 1, 2, ..., n$
where

- y_i is the i^{th} actual value of the dependent variable.
- x_{ij} is the i^{th} actual value of the j^{th} explanatory variable.
- β_0 is the intercept.
- β_j is the (partial) slope of the j^{th} independent variable.
- ε_i is i^{th} value the error term, which is $\varepsilon_i \sim N(0, \sigma^2)$

The parameters (β_0, β_1, β_2, $\cdots$, β_k) are interpreted as follows:

- β_0 is the value of the dependent variable when the values of all the independent variables are zero.
- β_j is the change in the value of the dependent variable when the value of the j^{th} independent variable increases by 1 unit assuming all the others the same.

The objective in the above model is to estimate the regression parameters, β_0 and $\beta_j;\ j = 1, 2, \cdots, k$ using sample data. To do so, there are assumptions:

- Normal distribution: The errors are normally distributed.
- Homoscedasticity: The variance of the errors is constant for all values of the explanatory variable.
- Independence: Errors are independent and have a zero mean.
- No multicollinearity: The explanatory variables are not linearly correlated.
- No omitted variables

Hence, the estimated regression model is: $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \hat{\beta}_2 x_{2i} + \cdots + \hat{\beta}_k x_{ki};\ i = 1, 2, ..., n$
where

- $\hat{y}_i$ is the i^{th} fitted value of the dependent variable.
- x_{ij} is the i^{th} actual value of the j^{th} explanatory variable.
- $\hat{\beta}_0$ is the estimated intercept.
- $\hat{\beta}_j$ is the estimated (partial) slope of the j^{th} explanatory variable.

The Stata command to run an OLS regression is `regress`:

```
. regress DepVar IndepVars
```

For example, the command

```
. regress y x1 x2 x3
```

fits the response variable y on the three explanatory variables x_1, x_2 and x_3.

Example 8.3. Recall example 8.1. Let's perform regression analysis of sons' height on fathers' height and fathers' weight.

```
. regress SonH FathH FathW
```

```
      Source |       SS       df       MS              Number of obs =       8
-------------+------------------------------           F(  2,     5) =  183.85
       Model |  82.8731177     2  41.4365588           Prob > F      =  0.0000
    Residual |  1.12688232     5  .225376464           R-squared     =  0.9866
-------------+------------------------------           Adj R-squared =  0.9812
       Total |          84     7          12           Root MSE      =  .47474

------------------------------------------------------------------------------
        SonH |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       FathH |    1.14947   .1132307    10.15   0.000     .8584014    1.440539
       FathW |   .1007808   .0275995     3.65   0.015     .0298342    .1717275
       _cons |  -16.05862   6.363865    -2.52   0.053    -32.41745     .300218
------------------------------------------------------------------------------
```

The `Prob > F = 0.0000` is the p-value for the overall significance of the regression model. Since its value is less than 0.05, it assures the regression model is significant which means at least one of the two explanatory variables is significant.

The `Adj R-squared = 0.9812` and `Adj R-squared = 0.9812` are values of the *coefficient of determination* and *adjusted coefficient of determination*, respectively. Both tell how well the estimated model fits the data. That is, they measure *the proportion (percentage) of the variation in the dependent variable explained by the explanatory variables.* The draw back of *coefficient of determination* (R^2) is that it increases when the number of explanatory variables increases. Hence, the *adjusted coefficient of determination* (R^2_{adj}) penalizes for an increment in the number of variables (degrees of freedom) which is a modified version of the *coefficient of determination.* From the above output, both measures indicate about 98% of the variation in the height of sons is explained by both the height and weight of their fathers.

Looking at the p-value associated with each explanatory variable, it indicates each explanatory variable is significantly associated with the response (height of sons). In particular, an 1 inch increase in the height of fathers leads to a 1.149 inches increase in the height of sons, assuming the weight of the fathers constant. Similarly, an increase in the weight of fathers by 1 kg leads to a 0.101 inches increase in the height of sons, assuming the height of the fathers constant.

We've seen that both of the explanatory variables are significantly associated with the height of sons. But, of the two explanatory variables, which one is more important for determining the height of sons? The answer to this question is determined using the standardized regression coefficients β^* which are defined as $\beta^* = \beta(\sigma_x/\sigma_y)$. Hence, the standardized

coefficients tell how increases in the independent variables affect relative position within the group. We can determine whether a 1 standard deviation change in one independent variable produces more of a change in relative position than a 1 standard deviation change in another independent variable.

By adding the option `beta` in the `regress` command,

```
. reg SonH FathH FathW ,beta
```

the standardized parameter estimates will be displayed as follows.

```
      Source |       SS       df       MS              Number of obs =       8
-------------+------------------------------           F(  2,     5) =  183.85
       Model |  82.8731177     2  41.4365588           Prob > F      =  0.0000
    Residual |  1.12688232     5  .225376464           R-squared     =  0.9866
-------------+------------------------------           Adj R-squared =  0.9812
       Total |          84     7          12           Root MSE      =  .47474

------------------------------------------------------------------------------
        SonH |      Coef.   Std. Err.      t    P>|t|                     Beta
-------------+----------------------------------------------------------------
       FathH |    1.14947   .1132307    10.15   0.000                 .7718526
       FathW |   .1007808   .0275995     3.65   0.015                 .2776376
       _cons |  -16.05862   6.363865    -2.52   0.053                        .
------------------------------------------------------------------------------
```

The results show that an increase in the height of fathers by 1 standard deviation increases the height of sons by 0.772 standard deviation. Similarly, an increase in the weight of fathers by 1 standard deviation increases the height of sons by 0.278 standard deviation. Clearly, fathers' height is more determining sons' height than fathers' weight.

8.4 Including Qualitative Explanatory Variables

Based on the number of categories that qualitative variables have, they are classified into two: *binary* (*dichotomous*) and *multinomial* (*polytomous*). The first kind, *binary* variables, have only two sets of mutually exclusive categories. For example, gender (male or female) and patient outcomes (dead or alive) are binary variables.

The second kind of variables, *multinomial* variables, are those qualitative variables with three or more categories. For example, blood type (A, B, AB or O), teaching method (lecturing, using slides, discussion or other), clinical stage of a disease (none, mild or severe) and academic qualifications (BSc, MSc or PhD) are multinomial variables.

8.4.1 Binary Explanatory Variables

The simplest way of including a binary predictor in a regression analysis is to used a 0/1 coding scheme. Then, it can be directly entered into the regression model like that of a continuous explanatory variable.

8.4.2 Multinomial Explanatory Variables: The `i.` Operator

If some of the explanatory variables are qualitative with more than two categories, then it is inappropriate to include them in the model as if they were quantitative variables. This is because the numbers used to represent the various categories are merely identifiers and have no numeric significance. In such case, a set of binary variables, called *dummy* (*design* or *indicator*) variables, should be created to represent the categories of the explanatory variable.

Suppose, for example, that one of the explanatory variable is marital status with three categories: "Single", "Married", "Separated". In this case, taking one of the categories as a reference (comparison group), two design variables (d_1 and d_2) are required to represent marital status in a regression model. For example, if the category "Single" is taken as a reference, the two design variables, d_1 and d_2 are set to 0; when the subject is "Married", d_1 is set to 1 while d_2 is still 0; when the marital status of the subject is "Separated", $d_1 = 0$ and $d_2 = 1$ are used. The following table shows this example of design variables for marital status:

	Design Variables	
Marital Status	Married (d_1)	Separated (d_2)
Single	0	0
Married	1	0
Separated	0	1

In general, if a nominal explanatory variable X has m categories, then $m-1$ design variables are needed. The $m-1$ design variables are denoted as d_u and the coefficients of those design variables are denoted as $\beta_u,\ u = 1, 2, \cdots, m-1$. Thus the linear regression model of the continuous response variable would be

$$y_i = \beta_0 + \beta_1 d_1 + \beta_2 d_2 + \cdots + \beta_{m-1} d_{m-1} = \beta_0 + \sum_{u=1}^{m-1} \beta_u d_u$$

Therefore, to include such a multinomial explanatory variable in a regression analysis, first, create the design variables using the `tabulate` command with the `generate` option as shown so far in section 4.9.14. Then, by taking one of the design variables as a reference, entering all the remaining $m-1$ design variables in the model is the first option, but probably not the most efficient one.

The other option is to use Stata's qualitative variable declaration `i.` operator. The prefix `i.` should be written to the variable to tell Stata that this variable is multinomial. As a result, Stata will internally generate the design variables and then fits the regression model on the $m-1$ of design variables.

For example, if `x2` is a multinomial variable, the `regress` command is written as

```
. regress y x1 i.x2 x3
```

By default, the smallest coded category will be taken as a reference. But, if there is a need to change the baseline, say for example second category, it can be done as

```
. regress y x1 ib2.x2 x3
```

Example 8.4. To study factors affecting students performance on a Basic Statistics course, a random sample of 30 second year students are selected from one faculty in a a certain university. The basic response is the score of out of 100% (ScoreStud). The other variables are sex of the student (GenderStud), academic rank of the instructor (AcadRank), years of service of the instructor since making the rank (ServiceYear) and last year GPA of the Student (LastGPA). Let's fit a regression model and interpret the results.

```
. reg ScoreStud GenderStud i.AcadRank ServiceYear LastGPA
```

```
      Source |       SS       df       MS              Number of obs =      30
-------------+------------------------------           F(  5,    24) =   12.70
       Model |  2451.10687     5  490.221375            Prob > F      =  0.0000
    Residual |  926.259793    24   38.594158            R-squared     =  0.7257
-------------+------------------------------           Adj R-squared =  0.6686
       Total |  3377.36667    29   116.46092            Root MSE      =  6.2124

------------------------------------------------------------------------------
   ScoreStud |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
  GenderStud |  -8.100036   3.159929    -2.56   0.017    -14.62181   -1.578264
             |
    AcadRank |
   Lecturer  |   .9784472   3.044773     0.32   0.751    -5.305656    7.26255
Assi. Prof.  |   9.895759   3.423986     2.89   0.008     2.828999   16.96252
             |
 ServiceYear |   .4587003   .4074585     1.13   0.271    -.3822527   1.299653
     LastGPA |   5.221274   2.167057     2.41   0.024     .7486891   9.693859
       _cons |    35.3388   7.102875     4.98   0.000     20.67918   49.99841
------------------------------------------------------------------------------
```

As can be easily observed, ServiceYear is not statistically significant. The next procedure is to re-fit the model excluding this non-significant variable. That is,

```
. reg ScoreStud GenderStud i.AcadRank LastGPA
```

```
      Source |       SS       df       MS              Number of obs =      30
-------------+------------------------------           F(  4,    25) =   15.40
       Model |  2402.19516     4  600.548789            Prob > F      =  0.0000
    Residual |  975.171511    25  39.0068604            R-squared     =  0.7113
-------------+------------------------------           Adj R-squared =  0.6651
       Total |  3377.36667    29   116.46092            Root MSE      =  6.2455

------------------------------------------------------------------------------
   ScoreStud |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
  GenderStud |  -7.817445   3.166739    -2.47   0.021    -14.33947   -1.295424
             |
    AcadRank |
```

```
   Lecturer  |  -.1481101   2.890961    -0.05   0.960    -6.102155    5.805935
Assi. Prof.  |   9.553506   3.428649     2.79   0.010     2.492072    16.61494
             |
     LastGPA |   5.736507    2.12947     2.69   0.012     1.350781    10.12223
       _cons |   35.52797   7.138752     4.98   0.000     20.82543    50.23051
------------------------------------------------------------------------------
```

About 67% of the variation in the Basic Statistics score of students is explained by the gender of the student, academic rank of the instructor and last year GPA. The score of male students decreases by about 8 points as compared to female students. Those students who were educated by assistant professors scored about 10 points higher than those who were educated by graduate assistants. Also, an increase in the last year's GPA of the student by 1 point, increases the Basic Statistics course score by about 6 points.

8.5 Including Interaction Terms: The # Operator

Interaction terms are needed whenever there is reason to believe that the effect of one independent variable depends on the value of another independent variable. Instead of defining interaction terms in a new variable using the product of the two variables in case of having continuous variables or combinations for categorical data, the symbol # can be used. However simply writing

```
. reg y x z x#z
```

works only if x and z are categorical variables. In this case, all possible combinations are included as a dummy variable. To be clear about what Stata does, it would be always better to write like

```
. reg y x z i.x#i.z
```

which is not absolutely needed but recommendable. Similarly, to use an interaction between continuous variables, the `c.` operator should be used. For example, in the command

```
. reg y x z c.x#c.z
```

the `c.` operator tells Stata to create an interaction by multiplication of the two variables (not with all possible combination dummies).

8.6 Including Time-Series Operators

In time series or panel data analysis, there will be a need to use lagged or difference variables.

8.6.1 Lagged Variables: The `l.` Operator

A lagged variable can be easily generated with the prefix `l.` after declaring the time variable. The command

```
. gen LagY=l.y
```

creates the variable LagY which is y_{t-1}. Without generating the lagged variable, for instance, the command

```
. reg y l.y x1 x2
```

will fit a regression of y on the lagged value of y and two control variables x_1 and x_2. If more than one lag is needed, different choices are available.

Syntax	Variables	Description
`ll.`y	y_{t-2}	Double lagged variable used
`l(1/2).`y	y_{t-1}, y_{t-2}	Lagged and double lagged variables
`l(0/5).`y	y_t, y_{t-1}, $\cdots$, y_{t-5}	From non-lagged to 5 periods lagged variables

8.6.2 Difference: The `d.` Operator

Just like lagged variables, you can create difference variables. For instance, after setting the time variable, the command

```
. gen DiffY=d.y
```

creates $\delta_{y_t} = y_t - y_{t-1}$. Higher interval differences can be created with the same logic seen for the `l.` operator.

8.7 Regression Model Diagnostics

How good the model is will depend on how well it predicts the responses, the linearity of the model and the behavior of the residuals. Hence, after estimating a model, the next task is to check the entire regression assumptions.

A number of predicted values can be obtained after fitting the model. The most important are the predicted values for the dependent variable (usually called fitted values) and the predicted residuals. This can be done using the `predict` command immediately after running the regression. For example, the command

```
. predict Yhat
```

saves the predicted values for dependent variable under the variable name Yhat. To save the estimated residuals, the option `residuals` should be added. That is,

```
. predict Resid, r
```

saves the predicted residuals under the variable name Resid. Residuals are the difference between the observed values (y) and the predicted values ($\hat{y}$).

8.7.1 Checking the Normality Assumption

An important assumption of a regression model that impact the validity of all tests (p, t and F) is that residuals are normally distributed. If residuals do not follow a 'normal' pattern, then we should check for omitted variables, model specification, linearity, functional forms. In sum, we may need to reassess our model.

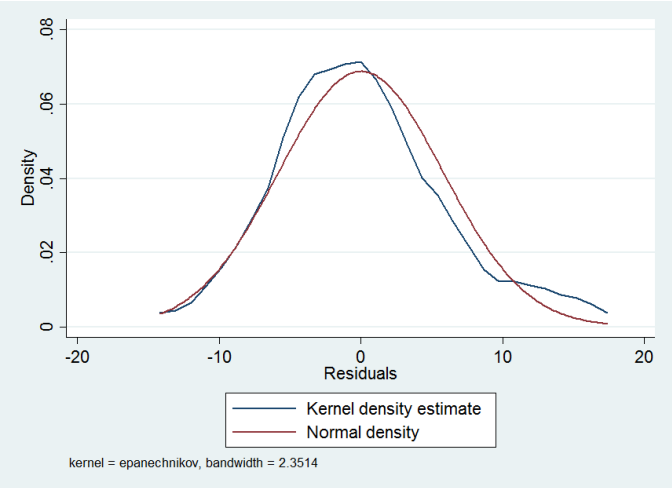


Figure 8.4: Plot of the Kernel Desity Estimate

Kernel Density Plot: The `kdensity` Command

The `kdensity` command produces the estimated kernel density plot, the option `normal` overlays a normal distribution to compare.

```
. kdensity Resid ,normal
```

This command displays figure 8.4. Here residuals deviate no more from a normal distribution.

P-P and Q-Q Plots: The `pnorm` and `qnorm` Commands

Standardize normal probability plot (P-P plot) (command `pnorm`) checks for non-normality in the middle range of residuals while quintile-normal plots (Q-Q plot) (command `qnorm`) check for non-normality in the extremes of the data.

The command

```
. pnorm Resid
```

produces the P-P plot shown in figure 8.5. Again, this plot indicates a slightly off the line in the middle.

Also, the command

```
. qnorm Resid
```

produces the Q-Q plot shown in figure 8.6. The extremes are a bit off from normality.

Shapiro-Wilk and Shapiro-Francia Tests: The `swilk` and `sfrancia` Commands

Common statistical tests for normality are the Shapiro-Wilk, Shapiro-Francia and Skewness-Kurtosis tests. All test the hypothesis that the distribution is normal, in this case the null hypothesis

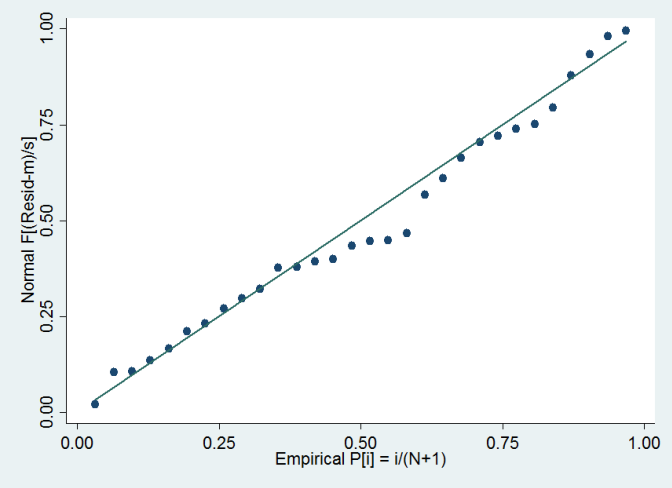


Figure 8.5: P-P Plot of Residuals

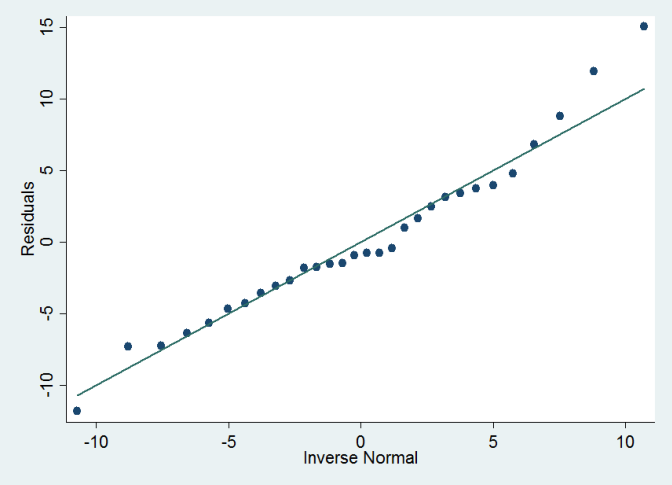


Figure 8.6: Q-Q Plot of Residuals

is that the distribution of the residuals is normal.

The `swilk` command performs the Shapiro-Wilk W test for normality and `sfrancia` performs the Shapiro-Francia W' test for normality. The first command can be used with $4 - 2000$ observations, and the second can be used with $10 - 5000$ observations. The structures of the commands are

```
. swilk Residuals
. sfrancia Residuals
```

For our previous example, the command is

```
. swilk Resid

                  Shapiro-Wilk W test for normal data

    Variable |    Obs       W           V         z       Prob>z
-------------+--------------------------------------------------
       Resid |     30    0.96960      0.966    -0.071    0.52837

. sfrancia Resid

                Shapiro-Francia W' test for normal data

    Variable |    Obs       W'          V'        z       Prob>z
-------------+--------------------------------------------------
       Resid |     30    0.96402      1.269     0.436    0.33132
```

Both test the null hypothesis which states that the data is normal. The results suggest not to reject the assumption of normality.

Skewness and Kurtosis Test: The `sktest` Command

The `sktest` presents a test for normality based on skewness and another based on kurtosis and then combines the two tests into an overall test statistic. It requires a minimum of 8 observations to make its calculations.

```
. sktest Residuals
```

For our example:

```
. sktest Resid

                  Skewness/Kurtosis tests for Normality
                                                      ------- joint ------
    Variable |    Obs   Pr(Skewness)   Pr(Kurtosis)  adj chi2(2)    Prob>chi2
-------------+-------------------------------------------------------------
       Resid |     30      0.1464         0.2944         3.52         0.1718
```

The larger the $p-$values confirms that the data is normal like that of the `swilk` and `sfrancia` results.

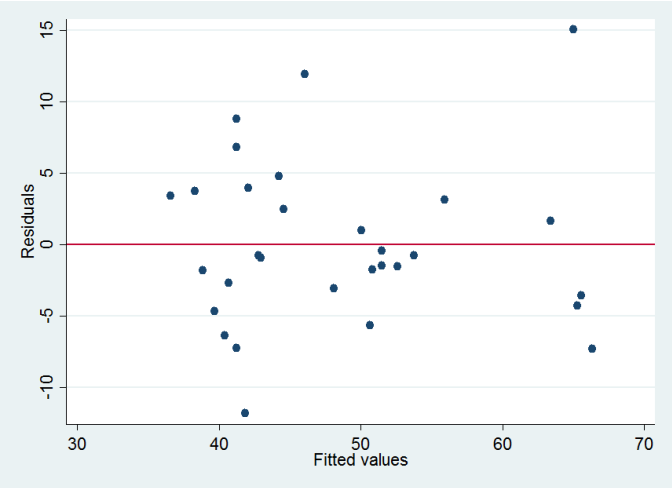


Figure 8.7: Plot of Residuals versus Fitted Values

8.7.2 Checking Homoscedasticity

An important assumption is that the variance in the residuals has to be homoscedastic or constant. Residuals should not be varied for lower or higher values of a certain explanatory variable (i.e. fitted values of y since $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$).

Plot of Residuals Vs Fitted Values: The `rvfplot` Command

The `rvfplot` command, right after running a regression, in Stata plots the residuals against the fitted values, figure 8.7.

```
. rvfplot ,yline(0)
```

There won't be any pattern if the residuals are homoscedastic. If it shows some shaped pattern, try to determine which of the predictor variable(s) is the cause. One way do doing this is to use the `rvpplot` command which plots residuals versus a predictor.

```
. rvpplot LastGPA
```

The Heteroskedasticity Test: The `estat hettest` Command

A statistical test to detect heteroscedasticity is the Breusch-Pagan & Cook-Weisberg test. The null hypothesis is that residuals are homoscedastic. For our example, we do as follows.

```
. estat hettest
```

The `estat` stands for **e**stimation **stat**istics.

```
Breusch-Pagan / Cook-Weisberg test for heteroskedasticity
         Ho: Constant variance
         Variables: fitted values of ScoreStud

         chi2(1)      =     0.63
         Prob > chi2  =   0.4278
```

The large $p-$value leads the constant variance assumption not to be rejected at the 99% confidence level. Therefore, the residuals are homogeneous.

The problem of heteroscedasticity is that the standard errors for the coefficients might be wrongly estimated and also their t-values. There are two ways to deal with this problem, one is using heteroscedasticity-robust standard errors, the other one is using weighted least squares. In practice it is recommended to use heteroscedasticity-robust standard errors to deal with heteroscedasticity.

By default, Stata assumes homoscedastic standard errors, so to adjust a model to account for heteroscedasticity, the option `robust` is used in the `regress` command. Had the assumption of homoscedasticity been rejected, the command could be written as follows.

```
. reg ScoreStud GenderStud i.AcadRank LastGPA ,r
```

8.7.3 Detection of Multicollinearity: The `estat vif` Command

An important assumption for the multiple regression model is that one predictor should not be a linear function of another. When multicollinearity is present, standard errors may be inflated. The Stata command is

```
. estat vif
```

If the VIF is less than 10, no indication of multicollinearity. For our case, there is no problem of multicollinearity as shown below.

```
    Variable |       VIF       1/VIF
-------------+----------------------
  GenderStud |      1.71    0.583455
    AcadRank |
           2 |      1.26    0.795546
           3 |      1.45    0.691280
     LastGPA |      1.37    0.729247
-------------+----------------------
    Mean VIF |      1.45
```

8.7.4 Omitted Variables Test: The `estat ovtest` Command

Omitted variables are variables that significantly influence the response variable and but not included in the fitted model. Testing for omitted variable bias is important since it is related to the assumption that the error term and the independent variables in the model are not correlated ($E(\varepsilon_i|x) = 0$). If the errors are correlated with the included predictors, then the regression coefficients are inconsistent.

In Stata, the `estat ovtest` command is used for this purpose.

```
. estat ovtest
```

```
Ramsey RESET test using powers of the fitted values of ScoreStud
       Ho:  model has no omitted variables
                  F(3, 22) =      0.24
                  Prob > F =      0.8683
```

The null hypothesis is that the model does not have omitted variables bias, the p-value is higher than the usual threshold of 0.05 (95% confidence level), so the null hypothesis cannot be rejected and conclude that the model has no variables omitted.

8.8 Estimation Results

8.8.1 Storing Estimation Results: The estimates store Command

We can store results of regressions and use previously stored results for further analysis like comparing estimation results from different models. The `estimates store` command saves the current (active) estimation results under a given name. For example, the command

```
. est store ModelA
```

will save the current regression results under the name ModelA.

Let's recall and save the previous three models considered before.

```
. reg ScoreStud GenderStud i.AcadRank ServiceYear LastGPA
. est store Model1
. reg ScoreStud GenderStud i.AcadRank LastGPA
. est store Model2
. reg ScoreStud GenderStud i.AcadRank LastGPA ,r
. est store Model3
```

8.8.2 Restoring Estimation Results: The estimates restore Command

If we want to activate a stored result, we can use the `estimates restore` command which loads the results saved under the specified name into the current (active) estimation results.

```
. est restore Model1
```

8.8.3 Estimates and Model Summaries: The estimates table Command

Making regression results table in Stata is easy using the `estimates table` command. To display a table of three models, say Model1 Model2 Model3, the command

```
. est table Model1 Model2 Model3 ,star stats(N r2 r2_a rmse)
```

adds information on number of observations, R^2, R^2_{adj} and root mean squared error (estimated standard deviation of the errors) in addition to the default parameters estimates. The option `star` is used to flag significant parameters.

Now let's make the estimation table using the above three stored models, Model1 Model2 Model3.

```
.  est table Model1 Model2 Model3 ,star stats(N r2 r2_a rmse)

-------------------------------------------------------------
    Variable |    Model1          Model2          Model3
-------------+-----------------------------------------------
  GenderStud |  -8.100036*     -7.8174454*     -7.8174454**
             |
    AcadRank |
   Lecturer  |  .97844717       -.1481101       -.1481101
Assi. Prof.  |  9.8957591**     9.5535062*      9.5535062**
             |
 ServiceYear |  .45870033
     LastGPA |   5.221274*      5.7365071*      5.7365071**
       _cons |  35.338797***     35.52797***     35.52797***
-------------+-----------------------------------------------
           N |         30              30              30
          r2 |  .72574497       .71126277       .71126277
        r2_a |  .66860851       .66506481       .66506481
        rmse |  6.2124197       6.2455472       6.2455472
-------------------------------------------------------------
                      legend: * p<0.05; ** p<0.01; *** p<0.001
```

If the standard errors are needed, the option `se` is to be added but, the option `star` should be removed.

To display numbers to be closer to the appropriate format, the command can be written as

```
. est table Model1 Model2 Model3 ,b(%7.3f) se(%7.3f) stfmt(%7.5g)
                                    stats(N r2 r2_a rmse)
```

The coefficients and standard errors `7.3f` format tells Stata to use a fixed width of (at least) 7 characters to display the number, with 3 digits after the decimal point. The `7.5g` tells Stata to use a general format where it tries to choose the best way to display a number, trying to fit everything within at most 7 characters, with at most 5 characters after the decimal point.

8.9 Comparing Regression Models

8.9.1 Likelihood-Ratio Test for Nested Models: The `lrtest` Command

Sometimes, determining the contribution of a group of variables may be an interest. Two models; one with all explanatory variables (full model) and the other without the explanatory variables to be tested (reduced model) are to be fitted. Thus, the reduced model is a special case of the full model.

Let ℓ_M denote the maximized value of the likelihood function for the model of interest M with $p_M = k+1$ parameters and let ℓ_R denote the maximized value of the likelihood function for the reduced model R with $p_R = k + 1 - q$ parameters. Note that model R is nested under model M. Thus, the null hypothesis $H_0 : \beta_1 = \beta_2 = \cdots = \beta_q = 0$ of no contribution

of all the q predictors in model M (according to the alternative, at least one of the extra parameters in the full model is nonzero) is tested using $G^2 = -2\log(\ell_R/\ell_M) \sim \chi^2(q)$ where $\ell_M = \ell(\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \cdots, \hat{\beta}_k)$ and $\ell_R = \ell(\hat{\beta}_0, \hat{\beta}_{q+1}, \hat{\beta}_{q+2}, \cdots, \hat{\beta}_k)$.

In Stata, such a test is conducted using the `lrtest` command. The structure of the command is

```
. lrtest ModelA ModelB
```

where ModelA is nested under ModelB.

For our case, if we consider Model1 and Model2 stored above,

```
. lrtest Model1 Model2
```

then Stata delivers

```
Likelihood-ratio test                                 LR chi2(1)  =      1.54
(Assumption: Model2 nested in Model1)                 Prob > chi2 =    0.2141
```

Clearly, there is no advantage of including ServiceYear in the model. Hence, Model2 is better.

Note that in Model2, one of the design variable of AcadRank is significant but the other not. Therefore, to determine whether or not AcadRank is significant in general, we need to fit another model excluding this variable and store it say Model4, that is,

```
. reg ScoreStud GenderStud LastGPA
. est store Model4
```

Hence, the interest is to compare Model2 and Model4. Notice that Model4 is nested under Model2.

```
. lrtest Model4 Model2
```

```
Likelihood-ratio test                                 LR chi2(2)  =      8.17
(Assumption: Model4 nested in Model2)                 Prob > chi2 =    0.0168
```

Even if one of the design variable of AcadRank is not significant, AcadRank is a significantly determining factor of Basic Statistics score in general.

8.9.2 Akaike and Bayesian Information Criteria: The `estat ic` Command

Comparing alternative specifications for model using Akaike's Information Criterion and Schwartz's Bayesian Information Criterion using **e**stimation **stat**istics `ic`.

Run the regression of interest, then type the following command:

```
. estat ic
```

```
Akaike's information criterion and Bayesian information criterion

------------------------------------------------------------------------------
       Model |    Obs    ll(null)   ll(model)     df          AIC         BIC
-------------+----------------------------------------------------------------
           A |     30    -113.423    -94.7894      5     199.5788    206.5848
------------------------------------------------------------------------------
               Note:  N=Obs used in calculating BIC; see [R] BIC note
```

In the estimates table commad, the option `stats(aic bic)` can be added to display the AIC and BIC values.

Chapter 9

Logistic Regression Models

9.1 Binary Logistic Regression: The `logit` Command

Suppose there are k explanatory variables (categorical, continuous or both) to be considered simultaneously. Consequently, the the logit model is

$$\text{logit } \pi(\boldsymbol{x}_i) = \log\left[\frac{\pi(\boldsymbol{x}_i)}{1-\pi(\boldsymbol{x}_i)}\right] = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ik}.$$

Here, $\exp(\beta_j)$ represents the odds ratio associated with an exposure if X_j is binary (exposed $x_{ij} = 1$ versus unexposed $x_{ij} = 0$); or it is the odds ratio due to a unit increase if X_j is continuous ($x_{ij} = x_{ij} + 1$ versus $x_{ij} = x_{ij}$).

Polytomous Explanatory Variables

If there is a categorical explanatory variable with more than two categories, then it is inappropriate to include it in the model as if it was quantitative. This is because the codes used to represent the various categories are merely identifiers and have no numeric significance. In such case, a set of binary variables, called design (dummy, indicator) variables, should be created to represent such a polytomuous variable.

Suppose, for example, that one of the explanatory variable is marital status with three categories: "Single", "Married", "Separated". In this case, taking one of the categories as a reference (comparison group), two design variables (d_1 and d_2) are required to represent marital status in a regression model. For example, if the category "Single" is taken as a reference, the two design variables, d_1 and d_2 are set to 0; when the subject is "Married", d_1 is set to 1 while d_2 is still 0; when the marital status of the subject is "Separated", $d_1 = 0$ and $d_2 = 1$ are used. The following table shows this example of design variables for marital status:

	Design Variables	
Marital Status	Married (d_1)	Separated (d_2)
Single	0	0
Married	1	0
Separated	0	1

In general, if a polytomuous variable X has m categories, then $m-1$ design variables are needed. The $m-1$ design variables are denoted as d_u and the coefficients of those design variables are denoted as $\beta_u,\ u = 1, 2, \cdots, m-1$. Thus, the logit model would be:

$$\text{logit } \pi(x_i) = \beta_0 + \beta_1 d_{i1} + \beta_2 d_{i2} + \cdots + \beta_{m-1} d_{i,m-1}.$$

A logistic regression model can be fitted using the `logit` command which outputs model parameter estimates or the `logistic` command which outputs odds ratios. The structure of the commands is:

```
. logit DepVar IndepVars
```

or

```
. logistic DepVar IndepVars
```

where the DepVar is the response variable and IndepVars is list of the explanatory variables. Also, by adding the option `or` in the `logit` command, the odds ratio estimates can be obtained.

Example 9.1. Using the JUSH_HAART data, suppose, we are interested in identifying the risk factors associated with HAART treatment defaulter patients. Let's consider four explanatory variables: age in years (Age), sex of the patients (Gender: 0=Female, 1=Male), functional status (FunStat: 0=Working, 1=Ambulatory, 2=Bedridden) and number of baseline CD4 counts (CD4). Let us fit the logit model and interpret.

The response variable takes the value $y_i = 1$ if the patient was defaulted and $y_i = 0$ otherwise (if the patient was on the treatment).

The design variables for Functional Status are:

	Design Variables	
Functional Status	Ambulatory (d_{31})	Bedridden (d_{32})
Working	0	0
Ambulatory	1	0
Bedridden	0	1

Now the model can be written as

$$\text{logit } \pi(\boldsymbol{x}_i) = \beta_0 + \beta_1 \text{ Age}_i + \beta_2 \text{ Gender}_i + \beta_{31} \text{ Ambulatory}_i + \beta_{32} \text{ Bedridden}_i + \beta_4 \text{ CD4}_i$$

Thus, the command

```
. logit Defaulter Age Gender i.FunStat CD4
```

provides:

```
Iteration 0:   log likelihood = -782.52567
Iteration 1:   log likelihood = -754.41416
Iteration 2:   log likelihood = -753.55606
Iteration 3:   log likelihood = -753.55555
Iteration 4:   log likelihood = -753.55555
```

```
Logistic regression                                     Number of obs   =       1464
                                                        LR chi2(5)      =      57.94
                                                        Prob > chi2     =     0.0000
Log likelihood = -753.55555                             Pseudo R2       =     0.0370

------------------------------------------------------------------------------
   Defaulter |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         Age |  -.0287807   .0079524    -3.62   0.000    -.0443671   -.0131942
      Gender |    .508136   .1379462     3.68   0.000     .2377664    .7785056
             |
     FunStat |
 Ambulatory  |   .5221699    .140292     3.72   0.000     .2472026    .7971371
  Bedridden  |   1.289291   .2857969     4.51   0.000     .7291397    1.849443
             |
         CD4 |  -.0006967   .0004349    -1.60   0.109    -.0015491    .0001557
       _cons |   -.552179   .2756388    -2.00   0.045    -1.092421   -.0119368
------------------------------------------------------------------------------
```

If we need the odds ratios, the command can be either

```
. logistic Defaulter Age Gender i.FunStat CD4
```

or

```
. logit Defaulter Age Gender i.FunStat CD4 ,or
```

which provides the result:

```
Iteration 0:   log likelihood = -782.52567
Iteration 1:   log likelihood = -754.41416
Iteration 2:   log likelihood = -753.55606
Iteration 3:   log likelihood = -753.55555
Iteration 4:   log likelihood = -753.55555

Logistic regression                                     Number of obs   =       1464
                                                        LR chi2(5)      =      57.94
                                                        Prob > chi2     =     0.0000
Log likelihood = -753.55555                             Pseudo R2       =     0.0370

------------------------------------------------------------------------------
   Defaulter | Odds Ratio   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         Age |   .9716295   .0077268    -3.62   0.000     .9566027    .9868924
      Gender |    1.66219   .2292928     3.68   0.000     1.268413    2.178215
             |
     FunStat |
 Ambulatory  |   1.685681   .2364876     3.72   0.000     1.280439    2.219179
```

```
  Bedridden   |   3.630213   1.037504     4.51   0.000     2.073296    6.356278
              |
         CD4  |   .9993036   .0004346    -1.60   0.109     .9984521    1.000156
       _cons  |    .575694   .1586836    -2.00   0.045     .3354034    .9881341
------------------------------------------------------------------------------
```

As it can be seen from this result, Age, Gender and Functional Status (since both of the design variables are significant) are significant at 5% level of significance. When the age of a patient increases by one year, the odds of being defaulted decreases by a factor of 0.9716 assuming all other variables are same. Also, males are 1.6623 times more likely to default than females, that is, the odds of being defaulted is 66.23% higher for males than for females, assuming the other variables constant. Again, assuming all other variables constant, ambulatory and bedridden patients are 1.6857 and 3.6302 times more likely to be defaulted than working patients, respectively.

Model Checking

Once the variable selection process is addressed, then the selected model should be explored for assessing whether the assumptions of the probability model are satisfied. The diagnostic methods for logistic regression, like that of linear regression, mostly rely residuals which compare observed and predicted values. Goodness-of-fit statistics are often computed as an objective measures of the overall fit of a model. A model checked and if it is found lacking the fit, a new model is proposed - fitted and then checked. And this process is repeated until a satisfactory model is found.

Model Specification Error: The `linktest` Command

The specification link test for single-equation models can be done using the Stata command, `linktest`. For our example before,

```
. linktest

Iteration 0:   log likelihood = -782.52567
Iteration 1:   log likelihood = -754.80947
Iteration 2:   log likelihood = -753.52451
Iteration 3:   log likelihood = -753.52172
Iteration 4:   log likelihood = -753.52172

Logistic regression                               Number of obs   =       1464
                                                  LR chi2(2)      =      58.01
                                                  Prob > chi2     =     0.0000
Log likelihood = -753.52172                       Pseudo R2       =     0.0371

------------------------------------------------------------------------------
   Defaulter |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _hat  |   1.099447    .405066     2.71   0.007     .3055325    1.893362
      _hatsq |   .0461809   .1769876     0.26   0.794    -.3007084    .3930701
```

```
       _cons |    .042611   .2369357     0.18   0.857    -.4217745    .5069966
------------------------------------------------------------------------------
```

The insignificant `_hatsq` ($p-$value=0.26) indicates the link function is correctly specified (no specification error). Had `_hatsq` been significant, it means either we have omitted relevant variables or the link function is not correctly specified.

The Pearson Chi-squared Goodness-of-fit Statistic: The `lfit` Command

The Pearson residual is the standardized difference between the observed and expected number of successes. Thus, the Pearson chi-squared statistic is the sum of the square of standardized residuals. When this statistic is close to zero, it indicates a good model fit to the data. When it is large, it is an indication of lack of fit.

The Pearson residuals are obtained in Stata using the `predict` command after running the model. And, the `lfit` command provides the significance of the Pearson chi-squared statistic. For our case,

```
. lfit

Logistic model for Defaulter, goodness-of-fit test

       number of observations =      1464
 number of covariate patterns =      1410
           Pearson chi2(1404) =      1445.03
                  Prob > chi2 =         0.2178
```

Since the p-value is 0.2178 which is large, the null hypothesis of good fit cannot be rejected. Therefore, the model fits the data.

The Hosmer-Lemeshow Test Statistic

The Pearson chi-squared goodness-of-fit test cannot be readily applied if there are only one or a few observations for each possible value (combination of values) of the explanatory variable(s). Consequently, the Hosmer-Lemeshow statistic, the best goodness-of-fit test with continuous explanatory variables, was developed to address this problem. The idea is to aggregate similar observations into (mostly 10 - decile) groups that have large enough samples so that a Pearson statistic is computed on the observed and predicted counts from the groups. That is,

$$HL = \sum_{i=1}^{m} \frac{[n_{1i} - n_i\hat{\pi}(\boldsymbol{x}_i)]^2}{n_i\hat{\pi}(\boldsymbol{x}_i)[1-\hat{\pi}(\boldsymbol{x}_i)]} \sim \chi^2(m-2).$$

This test statistic can be obtained by adding the option `group(10)` in the `lfit` command. That is,

```
. lfit ,group(10)

Logistic model for Defaulter, goodness-of-fit test
```

```
(Table collapsed on quantiles of estimated probabilities)

      number of observations =      1464
            number of groups =        10
     Hosmer-Lemeshow chi2(8) =         5.05
                Prob > chi2 =         0.7522
```

Similar to the Pearson goodness-of-fit, the non-significant of the Hosmer-Lemeshow test statistics suggests that the model fits the data reasonably well.

9.2 Multinomial Logistic Regression: The mlogit Command

Let Y be a categorical response with J categories. Multinomial (also called polytomous) logit models for nominal response variables simultaneously describe log odds for all $\binom{J}{2}$ pairs of categories. Given a certain choice of $J-1$ of these, the rest are redundant.

Let $\pi_j(x) = P(Y = j|x)$ at a fixed setting x for explanatory variables with $\sum_{j=1}^{J} \pi_j(x) = 1$. Thus, Y has a multinomial distribution with probabilities $\{\pi_1(x), \pi_2(x), \cdots, \pi_J(x)\}$.

Logit models pair each response category with a baseline category, often the last category or the most common one. The model

$$\log\left[\frac{\pi_j(\boldsymbol{x}_i)}{\pi_J(\boldsymbol{x}_i)}\right] = \beta_{j0} + \beta_{j1}x_{i1} + \beta_{j2}x_{i2} + \cdots + \beta_{jk}x_{ik};\ j = 1, 2, \cdots, J-1$$

simultaneously describes the effects of the explanatory variables on these $J-1$ logit models. The effects vary according to the response paired with the baseline. The $J-1$ equations determine parameters for logit models with other pairs of response categories, since

$$\log\left[\frac{\pi_1(x)}{\pi_2(x)}\right] = \log\left[\frac{\pi_1(x)/\pi_J(x)}{\pi_2(x)/\pi_J(x)}\right] = \log\left[\frac{\pi_1(x)}{\pi_J(x)}\right] - \log\left[\frac{\pi_2(x)}{\pi_J(x)}\right]$$

The structure of the commands is:

```
. mlogit DepVar IndepVars
```

Example 9.2. Based on the survival outcome of HAART treatment, HIV/AIDS patients were classified into four categories (0= Active, 1= Dead, 2= Transferred to other hospital, 3= Lost-to-follow). To identify factors associated with these survival outcomes, let us fit a multinomial logit model with three explanatory variables: Age, Gender (0= Female, 1= Male) and Functional Status (0= Working, 1= Ambulatory, 2= Bedridden).

```
. mlogit Status Age Gender i.FunStat
```

Results

```
Iteration 0:   log likelihood = -1106.8354
Iteration 1:   log likelihood = -1083.8531
Iteration 2:   log likelihood = -1073.6673
```

```
Iteration 3:   log likelihood = -1073.3997
Iteration 4:   log likelihood = -1073.3983
Iteration 5:   log likelihood = -1073.3983

Multinomial logistic regression                 Number of obs   =       1464
                                                LR chi2(12)     =      66.87
                                                Prob > chi2     =     0.0000
Log likelihood = -1073.3983                     Pseudo R2       =     0.0302

------------------------------------------------------------------------------
      Status |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
Active       |  (base outcome)
-------------+----------------------------------------------------------------
Dead         |
         Age |  -.0195464   .0181194    -1.08   0.281    -.0550597    .0159669
      Gender |   .5637572   .3253754     1.73   0.083    -.0739669    1.201481
             |
     FunStat |
 Ambulatory  |   .9397352   .3325129     2.83   0.005     .2880219    1.591448
  Bedridden  |    2.28018   .4789242     4.76   0.000     1.341505    3.218854
             |
       _cons |  -3.271366   .6237115    -5.24   0.000    -4.493818   -2.048914
-------------+----------------------------------------------------------------
Transferred  |
         Age |  -.0302947   .0124265    -2.44   0.015    -.0546502   -.0059392
      Gender |   .6348197   .2111411     3.01   0.003     .2209907    1.048649
             |
     FunStat |
 Ambulatory  |    .833139   .2089071     3.99   0.000     .4236886    1.242589
  Bedridden  |    1.58372   .3925449     4.03   0.000     .8143462    2.353094
             |
       _cons |  -1.882046   .4133479    -4.55   0.000    -2.692193   -1.071899
-------------+----------------------------------------------------------------
Lost_to_follow |
         Age |  -.0317221   .0104999    -3.02   0.003    -.0523015   -.0111426
      Gender |   .4545194    .178057     2.55   0.011      .105534    .8035047
             |
     FunStat |
 Ambulatory  |   .2920603   .1825624     1.60   0.110    -.0657554     .649876
  Bedridden  |   .8279059   .3953184     2.09   0.036      .053096    1.602716
             |
       _cons |  -1.116181   .3433219    -3.25   0.001    -1.789079   -.4432823
------------------------------------------------------------------------------
```

The odds that male patients being dead (instead of active) is $\exp(0.565) = 1.759$ times that of females, or the odds of being dead (instead of active) is 75.9% higher for males than for

females. In other words, male patients are 1.759 times more likely to be dead (instead of active) than female patients. Also, those ambulatory patients are exp(0.941) = 2.563 times more likely to be dead (instead of active) than those working patients. Similarly, those bedridden patients are exp(2.280) = 9.777 times more likely to be dead (instead of active) than those working patients. In addition, the functional status effects indicate that the odds of being dead (instead of active) are relatively higher for those bedridden patients than those ambulatory patients.

9.3 Ordinal Logistic Regression: The ologit Command

Models with terms that reflect ordinal characteristics such as monotone trend have improved model parsimony and power. Let Y is an ordinal response with J categories. Then there are $J-1$ ways to dichotomize these outcomes. These are $Y \leq 1(Y=1)$ versus $Y > 2$, $Y \leq 2$ versus $Y > 2, \cdots, Y \leq J-1$ versus $Y > J-1(Y=J)$.

A model that simultaneously uses all cumulative logits is

$$\text{logit } P(Y_i \leq j|\boldsymbol{x}_i) = \beta_{j0} + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik};\ j = 1, 2, \cdots, J-1.$$

Each cumulative logit has its own intercept and same effect associated with the explanatory variables. Each intercept increases in j since logit $[P(Y \leq j|x)]$ increases in j for a fixed x, and the logit is an increasing function of this probability.

An odds ratio of cumulative probabilities is called a cumulative odds ratio. An ordinal logit model has a proportionality assumption which means the distance between each category is equivalent (proportional odds assumption). This assumption often is violated in practice. There is a need to test if this assumption holds (can use a Brant test in Stata).

```
. ologit DepVar IndepVar
```

Example 9.3. To determine the effect of Age and Gender (0= Female, 1=Male) on the Clinical Stage of HIV/AIDS patients (1= Stage I, 2= Stage II, 3= Stage III and 4= Stage IV), the following parameter estimates of ordinal logistic regression are obtained.

```
. ologit ClinStag Age Gender
```

Results

```
Iteration 0:   log likelihood = -1854.3173
Iteration 1:   log likelihood = -1852.1355
Iteration 2:   log likelihood = -1852.1351
Iteration 3:   log likelihood = -1852.1351

Ordered logistic regression                     Number of obs   =       1464
                                                LR chi2(2)      =       4.36
                                                Prob > chi2     =     0.1128
Log likelihood = -1852.1351                     Pseudo R2       =     0.0012
```

```
------------------------------------------------------------------------------
    ClinStag |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         Age |   .0033714   .0054717     0.62   0.538    -.0073529    .0140957
      Gender |   .1789047   .1028413     1.74   0.082    -.0226605    .3804699
-------------+----------------------------------------------------------------
       /cut1 |  -.9905059   .1884011                     -1.359765   -.6212465
       /cut2 |   .5382869   .1869514                      .1718689    .9047049
       /cut3 |   2.724598   .2066293                      2.319612    3.129584
------------------------------------------------------------------------------
```

The cumulative estimates $\hat{\beta}_1 = 0.0034$ suggests that the cumulative probability starting at the clinical stage IV end of the scale increases as the age of the patient increases (an increase in the age of the patient leads to be in higher clinical stages) given the gender. Also, the estimate $\hat{\beta}_2 = 0.1789$ indicates the estimated odds of being in the clinical stage below any fixed level for males are $\exp(0.179) = 1.1960$ times the estimated odds for female patients (males are more likely to be in higher clinical stages as compared to females) given the age of the patient. (NOTE: Since both of the slopes are non-significant, their estimates should not be interpreted. Here, I have interpreted them just for illustration purposes.)

Chapter 10

Poisson and Negative Binomial Models

10.1 The Poisson Regression: The `poisson` Command

If there are k explanatory variables, the multiple poisson regression model is written as:

$$\log \mu(\boldsymbol{x}_i) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik} = \sum_{j=0}^{k} \beta_j x_{ij} \tag{10.1}$$

where $x_{i0} = 1$ for all $i = 1, 2, \cdots, n$. Here, $\mu(\boldsymbol{x}_i)$ is the conditional mean of Y_i given $\boldsymbol{x}_i$ where $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \cdots, x_{ik})$.

Example 10.1. Suppose a study is conducted in identifying factors associated with CD4 counts of HIV/AIDS patients at the start of HAART treatment. Here the response variable is CD4 count of a patient and the explanatory variables were Age in years (Age), Gender (0=Female, 1=Male) and Functional Status (0=Working, 1=Ambulatory, 2=Bedridden). Let us find the parameter estimates and their corresponding standard errors of the poisson regression model.

```
. poisson CD4 Age Gender i.FunStat
```

Results

```
Iteration 0:   log likelihood = -85956.463
Iteration 1:   log likelihood = -85956.397
Iteration 2:   log likelihood = -85956.397

Poisson regression                                Number of obs   =       1464
                                                  LR chi2(4)      =   12209.83
                                                  Prob > chi2     =     0.0000
Log likelihood = -85956.397                       Pseudo R2       =     0.0663

------------------------------------------------------------------------------
         CD4 |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
```

```
         Age |   .0059927   .0002057    29.14   0.000     .0055896    .0063958
      Gender |  -.1982254    .004114   -48.18   0.000    -.2062887   -.1901622
             |
     FunStat |
  Ambulatory |  -.3782717   .0045673   -82.82   0.000    -.3872233     -.36932
   Bedridden |  -.6295727   .0123055   -51.16   0.000     -.653691   -.6054543
             |
       _cons |   5.264264   .0071412   737.17   0.000     5.250267     5.27826
------------------------------------------------------------------------------
```

As the age of the patient increases by one year, the mean CD4 count increases by 0.60% [$\exp(0.0060) - 1 = 0.60\%$]. The mean CD4 count of male patients decreases by 17.98% [$1 - \exp(-0.1982) = 17.98\%$] than female patients. Similarly the mean CD4 counts of ambulatory and bedridden patients decreases by 31.50% and 46.72% than working patients, respectively.

10.2 Negative Binomial Regression: The `nbreg` Command

For a poisson distribution, the variance and the mean are equal. Often count data vary more than the expected. The phenomenon of the data having greater variability than expected is called *over-dispersion*. But, over-dispersion is not an issue in ordinary regression models assuming normally distributed response, because the normal distribution has a separate parameter to describe the variability.

In the presence of over-dispersion, a negative binomial model is should be applied. Like a poisson model, a negative binomial model expresses the log mean response in terms of the explanatory variables. But a negative binomial model has an additional parameter called a *dispersion parameter*. That is, because, the negative binomial distribution has mean $E(Y) = \mu$ and variance $\text{Var}(Y) = \mu + \alpha\mu^2$ where $\alpha > 0$. The index α is a dispersion parameter. As α approaches 0, $Var(Y)$ goes to μ and the negative binomial distribution converges to the poisson distribution. The farther α falls above 0, the greater the over-dispersion relative to poisson variability.

Example 10.2. Consider example 10.1. Let us again fit a negative-binomial regression model and compare the parameter estimates and their corresponding standard errors of both models.

```
. nbreg CD4 Age Gender i.FunStat
```

Results

```
Negative binomial regression                    Number of obs   =       1464
                                                LR chi2(4)      =     103.15
Dispersion     = mean                           Prob > chi2     =     0.0000
Log likelihood = -9083.7296                     Pseudo R2       =     0.0056

------------------------------------------------------------------------------
         CD4 |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         Age |   .0066501   .0022993     2.89   0.004     .0021436    .0111566
```

```
      Gender |  -.1841342   .0442593    -4.16   0.000    -.2708809   -.0973875
             |
    FunStat  |
 Ambulatory  |  -.3742912   .0460146    -8.13   0.000    -.4644781   -.2841042
  Bedridden  |  -.6331784   .1065753    -5.94   0.000    -.8420621   -.4242946
             |
       _cons |   5.236089   .0785945    66.62   0.000     5.082047    5.390132
-------------+----------------------------------------------------------------
    /lnalpha |  -.5071609   .0345145                      -.574808   -.4395138
-------------+----------------------------------------------------------------
       alpha |   .6022029   .0207847                      .5628129    .6443496
------------------------------------------------------------------------------
Likelihood-ratio test of alpha=0:  chibar2(01) = 1.5e+05 Prob>=chibar2 = 0.000
```

As the dispersion parameter α is significantly larger than 0, it assures that the negative binomial regression model is appropriate than the poisson regression model.