Statistical Computing I - Stat 2021

Basics of SPSS and Stata

Awol S.
Department of Statistics
College of Computing & Informatics
Haramaya University
Dire Dawa, Ethiopia

Contents

Course Guide Book

Module Title (Code):	Statistical Computing (MStat2021)
Course Title (Code):	Statistical Computing I (Stat2021)
Credit:	5 EtCTS
Credit Hours:	3hrs (2hrs lecture +3hrs laboratory)
Course Type:	Core
Academic Year:	2016/17 (2009 E.C)
Study Year (Semester):	2 (I)
Student's College/Institute:	College of Computing and Informatics
Department/School:	Department of Statistics
Study Program:	Undergraduate
Enrollment:	Regular
Instructor's Name:	Awol S.

This course is mainly aimed to help second year statistics students to get started with SPSS and Stata statistical software. As a result, it has two parts. Part I covers basics of the SPSS software package and Part II deals with that of the Stata software package. In order to comply with the goal of the course, it starts with a general introduction to the software packages and introduce then chapter by chapter more complicated notions. Every student is supposed to understand the statistical analysis and models presented here, and know how and when to use them.

Part I

Basics of the SPSS Software Package

Chapter 1

Some Statistical Concepts

The following are some key concepts that will be used throughout this course. Most of you are familiar with them, but it is worth reviewing the terms for the sake of remembering.

1.1 Common Statistical Terms

- **Datum**: It is an observed value(s) 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 collection of observed values (observations or cases) of some objects.
- **Variable**: It is a characteristic or an attribute that can assume different values. For example:
  - height (1m, 1.5m, 2.6m, $\cdots$)
  - family size (1, 2, 4, 7, $\cdots$)
  - gender (male, female)
  - blood type (A, B, AB, O)
  - grade (A, B, C, D, F)

Based on the values that a variable assumes, a variable is classified into two: Qualitative and Quantitative.

- **Qualitative/categorical variable**: is a variable that does not assume numeric values. For example:
  - gender (male, female) - Nominal/Binary
  - blood type (A, B, AB, O) - Nominal/Multinomial
  - grade (A, B, C, D, F) - Ordinal/Multinomial
- **Quantitative variable**: is, on the other hand, a variable which assumes numeric values. This variable is numeric in nature. For example:
  - height (1m, 1.5m, 2.6m, $\cdots$) - Continuous
  - family size (1, 2, 4, 7, $\cdots$) - Discrete

The reason for distinguishing between the type of variables is that the method of data analysis is different depending on the type of the variable.

1.2 Choice of Statistical Analysis

Most statistical methods distinguish the role of variables as dependent (response) and independent (explanatory) variable. Depending on the characteristics of the data (continuous/categorical) and the role of the data (explanatory/response), the appropriate method of statistical analysis should be chosen. Some of the common statistical methods are presented in the following table.

Independent Variable	Dependent Variable	Method of Data Analysis
categorical (binary)	continuous	t test
categorical (multinomial)	continuous	ANOVA
categorical	categorical	χ^2 test
continuous or categorical or both	continuous	regression
continuous or categorical or both	categorical	logistic regression

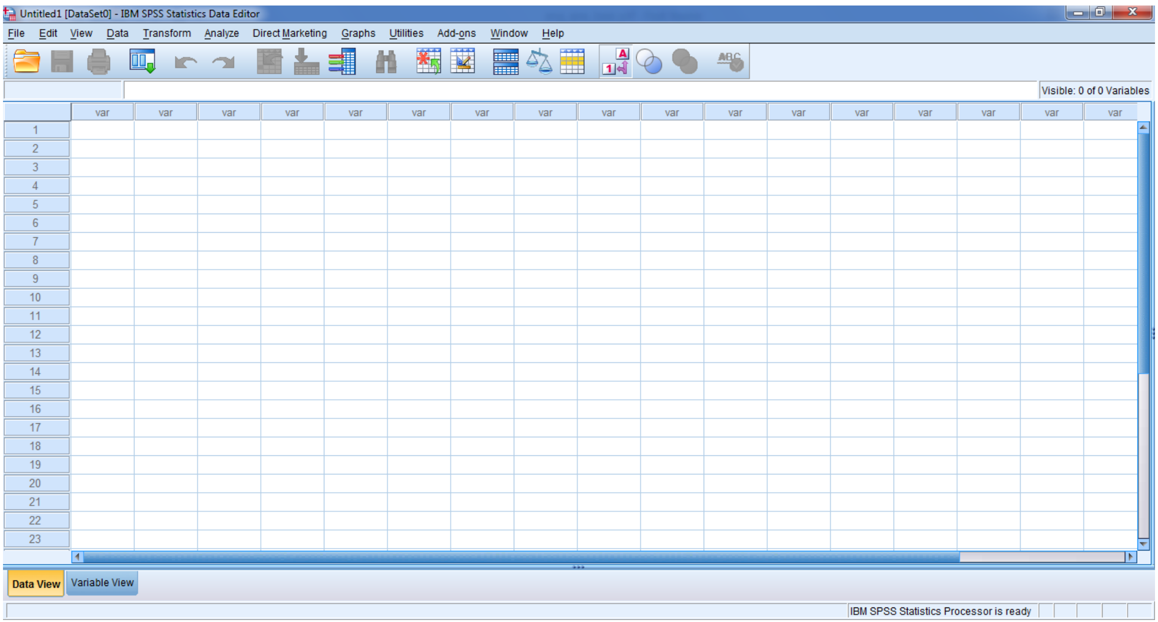
Chapter 2

An Introduction to SPSS

SPSS was an acronym of Statistical Package for the Social Sciences but now it stands for Statistical Product and Service Solutions. Originally developed as a programming language for conducting statistical analysis, it has grown into a complex and powerful application with both a graphical and a syntactical interface and provides dozens of functions for managing, presenting and analyzing data. Its statistical capabilities alone range from simple percentages to complex analyses of variance, multiple regressions, and generalized linear models.

2.1 Exploring SPSS Windows

To open SPSS, click on the **Start** → **IBM SPSS Statistics 20**. Then, the main SPSS interface looks the following.



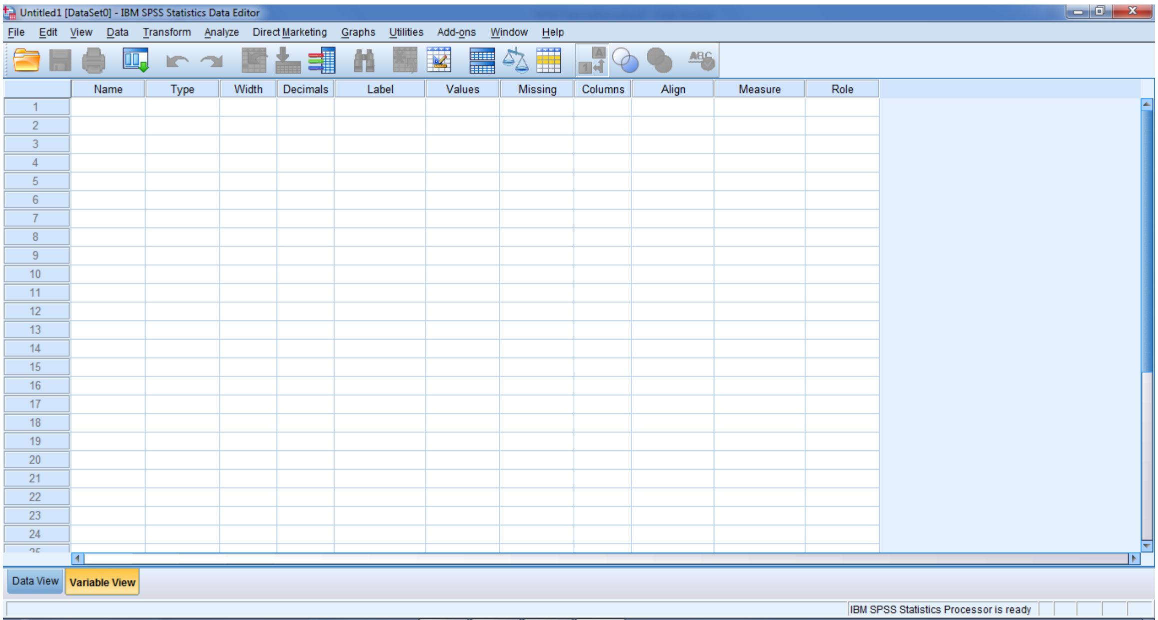
In the main interface, the **Title** bar at the top displays the name of the opened data file if any, or "Untitled1[DataSet0]" if empty or if the file has not yet been saved. Next, the *Menu* bar lists different pull down menus, grouping the available SPSS commands, which provides easy access to most SPSS features. Also, the *Status* bar at the bottom of each SPSS window tells what SPSS is currently doing. The *Status* bar runs along the bottom of a window and alerts the user to the status of the system. Typical messages one will see are "IBM SPSS Statistics Processor is ready", "Running procedure…".

Of the several windows that can be opened when using SPSS, the four common are the *Data Editor* Window, the *Output* Window, the *Syntax Editor* Window and the *Chart Editor* Window.

The Data Editor Window

The *Data Editor* window opens automatically when SPSS is started. It is a spreadsheet in which the variables are to be defined and the data are to be entered or edited. Each row corresponds to a case or an observation while each column represents a variable. Thus, the dimension of the data file is determined by the number of cases and variables. There is no limit to the number of variables and/or cases that can be used.

Notice the *Data Editor* window has two sheets since there are two tabs on the bottom labelled **Data View** and **Variable View**. The **Data View** sheet is used to enter data just as an Excel worksheet does. The **Variable View** sheet is used define each variable in the data set (defining variables in SPSS is described in detail in Section 2.2.1). Now to open the **Variable View** sheet, just click on the **Variable View** tab. Then, the sheet looks:



While the variables are listed as columns in the **Data View** sheet, they are listed as rows in the **Variable View**. In the **Variable View**, each row is a variable, each column is an attribute (characteristic) associated with that variable.

The Output (Viewer) Window

The output window displays the statistical results, tables, and charts from the analysis performed. An output window opens automatically when a procedure that generates output is run. In the output window, the results can be edited, moved, deleted and copied in a Microsoft Explorer-like environment. This window is not accessible until output has been generated.

A file with an extension of *.spo* is assumed to be a Viewer file containing statistical results and graphs.

The Chart Editor Window

The chart editor window is only displayed after SPSS has been requested to produce a plot (chart). In this window, the plots can be edited, i.e., the colors can be changed, different type fonts or sizes can be selected, axes can be rotated (switch the horizontal and vertical axes), the chart type can be changed and the like.

The Syntax Editor Window

Most SPSS commands are accessible from the SPSS menus and dialog boxes. However, some commands and options are available only by using the SPSS command language. In this case, the Syntax Window is used.

A file with an extension of *.sps* is assumed to be a Syntax file containing spss syntax and commands.

2.2 Entering and Saving Data

2.2.1 Defining Variables in SPSS

The **Variable View** sheet is used create variable names and define the attributes of each variable. The entries of this sheet are:

- **Name** - Variable names can be up to 64 characters, always beginning with a letter and not end with a period. They can contain numbers (also @, #, _ and $ characters) but no funny characters like spaces/blanks/hyphens or special characters. It does not matter if a variable is called WEIGHT, weight, or WEiGhT since variable names are not case sensitive. But they must be unique.
- **Type** - It indicates what type the variable is. There are more than eight variable types in SPSS. The most common are described below.
  - **Numeric**: A variable whose values are numbers. The *Data Editor* accepts numeric values both in standard format and scientific notation.
  - **Comma**: A numeric variable whose values are displayed with commas delimiting every three places, and with the period as a decimal delimiter, example $76,721.05$. The *Data Editor* accepts numeric values for comma variables with or without commas; or in scientific notation.
  - **Dot**: A numeric variable whose values are displayed with periods delimiting every three places, and with the comma as a decimal delimiter, example $76.721,05$. The *Data Editor* accepts numeric values for dot variables with or without dots; or in scientific notation.

- **Scientific notation**: A numeric variable whose values are displayed with an embedded `E` and a signed power-of-ten exponent. The *Data Editor* accepts numeric values for such variables with or without an exponent. The exponent is preceded by `E`, for example, 123, 1.23E2 or 1.23E+2.
- **Date**: A numeric variable whose values are displayed in one of several calendar date or clock-time formats. Dates can be entered with slashes, hyphens, periods, commas, or blank spaces as delimiters. The century range for 2-digit year values is determined by **Edit** menu and the **Options** submenu. Then, click on **Data** tab.
- **Custom currency**: A numeric variable whose values are displayed in one of the custom currency formats that you have defined in the **Currency** tab of the **Options** submenu of the **Edit** menu. Defined custom currency characters cannot be used in data entry but are displayed in the **Data Editor**.
- **String**: Some numbers are not really numbers. That is, they are numbers but you cannot use them in mathematical calculations. Take a phone number, for example, or an account number or a zip code. You can sort them, but you cannot add or subtract or multiply them. Well, you could, but the result would be meaningless. In essence, these numbers are actually just text which happens to be numeric. A good example is an address, which contains both numbers and text. We refer to these variables as string (text) or alphanumeric. Uppercase and lowercase letters are considered distinct.

- **Width** - The numerical entry in this box gives how many spaces the entries in the **Data View** will be for this variable.

- **Decimals** - For numeric data, this entry gives how many decimal places will be shown for this variable in the **Data View**.

- **Label** - It stands for a descriptive title for the variable. This makes all output much more understandable.

- **Values** - This is used for variables which are categorical. You can specify a label for each numerical value of a categorical variable.

- **Missing** - This allows you to specify which values for a variable indicate missing data. By default, SPSS assigns a period for missing data. However, sometimes data sets you might receive use special numerical values to indicate missing data.

- **Columns** - The numerical value in this item gives how many spaces will be allocated for the variable in the **Data View**. This is different from width in that width limits the number of spaces for the actual number. **Columns** limits how many spaces will be visible in the **Data View**.

- **Align** - This entry either left aligns, centers, or right aligns the entries for the variable.

- **Measure** - This indicates what measurement scale of variable is. The available measures are scale, ordinal, and nominal. A scale in SPSS is a quantitative variable.

2.2.2 Data Entry and Coding

There are three steps that must be followed to create a new data set in SPSS.

Example 2.1. For illustration, lets see how to enter the following data: three variables *Weight*, *Sex* (M, F) 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

STEP 1: Defining the Variables

Now note that *Weight* is a **Numeric** variable in nature. And *Sex* and *Marital Status* are both categorical (nominal). *Sex* will be treated as **String**. But since the categories of *Marital Status* are coded, it will be treated as a **Numeric** variable.

Having the above note in mind, first each variable should be defined with its characteristics in the **Variable View** sheet. To start, open a new SPSS and go to the **Variable View**. Then, on the first row, define the *Weight* variable as follows:

1. Write the name *Wei* standing for the *Weight* variable in the **Name** column.

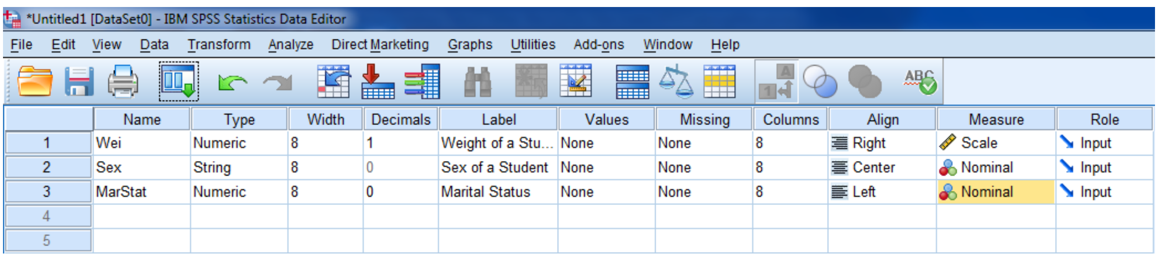
2. Change the **Type** column to **Numeric** which is the default.

3. Change the **Decimals** column to 1 since there is one decimal place in the values.

4. In the **Label** column, write "Weight of a Student".

5. Change the **Align** column to **Center**.

6. In the **Measure** column, change it to **Scale**.



	Name	Type	Width	Decimals	Label	Values	Missing	Columns	Align	Measure	Role
1	Wei	Numeric	8	1	Weight of a Stu...	None	None	8	Right	Scale	Input
2	Sex	String	8	0	Sex of a Student	None	None	8	Center	Nominal	Input
3	MarStat	Numeric	8	0	Marital Status	None	None	8	Left	Nominal	Input
4											
5											

On the second row of the **Variables View** sheet, define *Sex* as follows:

1. Write the name *Sex* standing for the *Sex* variable in the **Name** column.

2. Change the **Type** column to **String**.

3. In the **Label** column, write "Sex of a Student".

4. Change the **Align** column to **Center**.
5. In the **Measure** column, change it to **Nominal**.

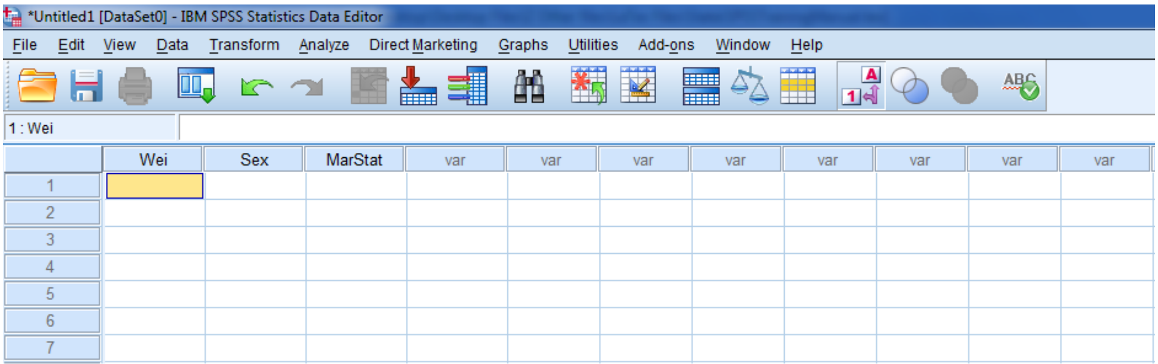
Similarly, define the *Sex* variable in the second row of the **Variables View** sheet:

1. Write the name *MarStat* standing for *Marital Status* in the **Name** column.
2. Change the **Type** column to **Numeric**.
3. Change the **Decimals** column to 0 since there is no decimal place in the values.
4. In the **Label** column, write "Marital Status".
5. Change the **Align** column to **Left**.
6. In the **Measure** column, change it to **Nominal**.

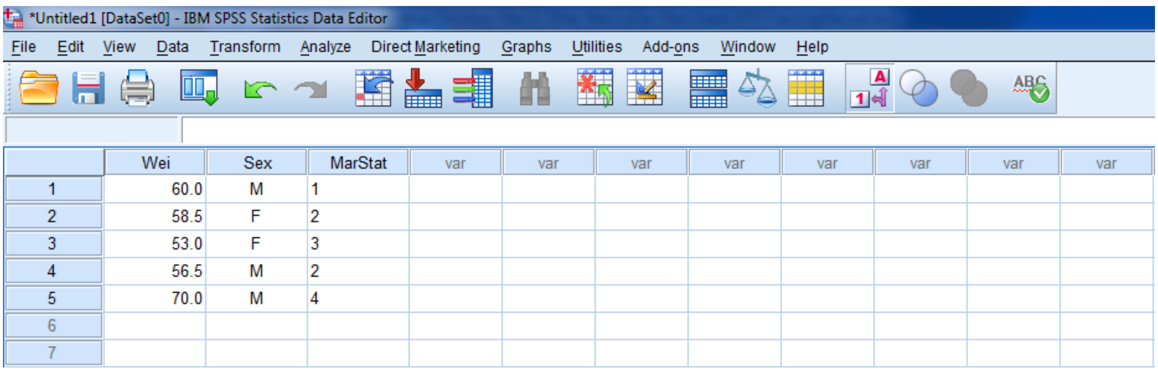
Note: Had you entered the data into the **Data View** prior to defining the variables, SPSS assigned the default variable names *VAR00001, VAR00002, VAR00003*. To change the variable names, click on the **Variable View** tab. Then change *VAR00001* to *Wei* and similarly the other two.

STEP 2: Entering the Data

Once all of the variables are defined, the data can be entered manually in the **Data View** sheet. Now go back to the **Data View** which shows the variable names as the column names. It looks like:



The data is then typed into one cell at a time. The information is entered into the cell when the active cell is changed. The mouse and the tab, enter, and cursor keys can be used to enter data.



STEP 3: Saving the Data

To retain the current data set, it must be saved to a file.

1. From the *Menu* bar, click on **File** → **Save As**.

2. In the **Save Data As** dialogue box, in the **File name:** field, write a data file name, say, *Student*. Since, from the **Save as type:** drop-down list, the default extension is 'SPSS Statistics (*.sav*)', then SPSS by default saves the data file by adding the extension *.sav*, that is, *Student.sav*.

3. From the **Look in:** drop-down list, select the location path where the file will be saved.

4. Then, click on the **Save** tab.

Now the *Title* bar looks:

Note: A file with an extension of *.sav* is assumed to be a data file in SPSS for Windows format.

To open a data file,

1. From the *Menu* bar, click on **File** → **Open** → **Data**.

2. In the **Open Data** dialogue box, from the **Look in:** drop-down list, select the location path where the file is saved. SPSS automatically searches the extension *.sav* since from the **Files of type:** drop-down list, the default extension is 'SPSS Statistics (*.sav*)'.

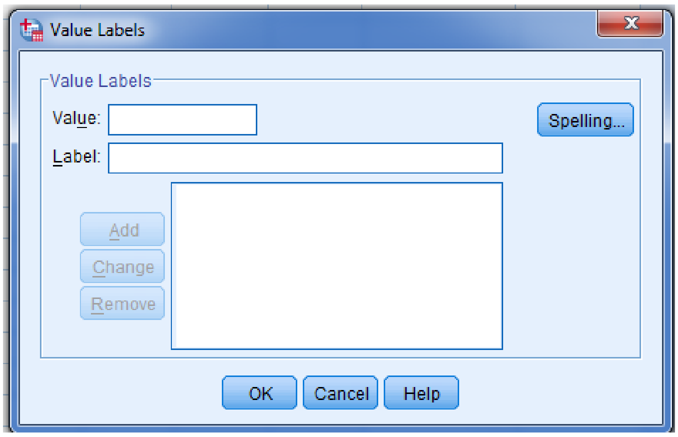
3. Of the list of data files, if any, click on the data file name to be opened, for example, *Student.sav*.

4. Then, click on the **Open** tab.

2.2.3 Creating Value Labels

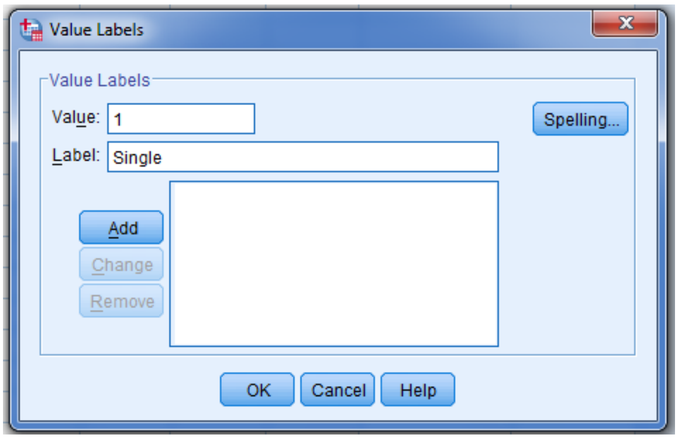
It is nice to have the values of a categorical variable labeled with their meaning. For example, *Marital Status* should have its categories as *Single*, *Married*, *Divorced* and *Other* rather than displayed as 1, 2, 3, and 4.

To create a label for the *MarStat* variable, click on the **Value** column of the *MarStat* variable in the **Variable View** sheet. Then the **Value Labels** dialogue box appears.

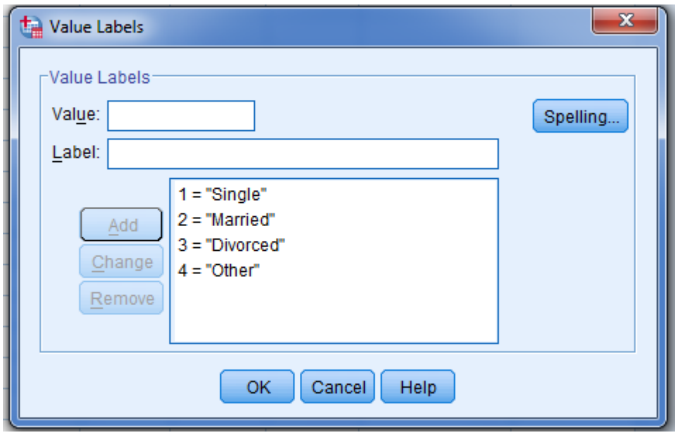


Then

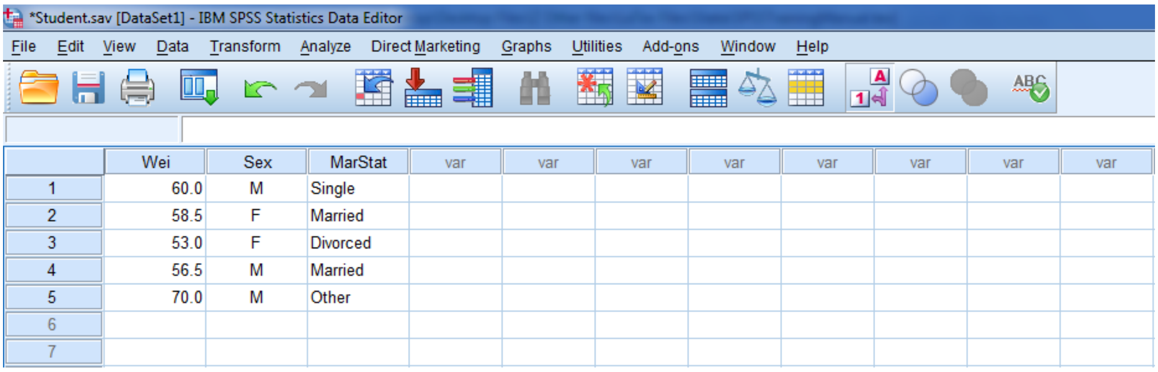
- Type 1 in the **Value** field.
- Write *Single* in the **Label** field.
- Click the **Add** tab to have this label added to the list.



In a similar way, continue labeling until all the values are labeled. Then click on **OK**. When all values are labeled, the **Value Labels** window becomes:



Now go back to the **Data View** and observe the difference.



	Wei	Sex	MarStat	var	var	var	var	var	var	var	var
1	60.0	M	Single								
2	58.5	F	Married								
3	53.0	F	Divorced								
4	56.5	M	Married								
5	70.0	M	Other								
6											
7											

Example 2.2. Enter the following data in SPSS 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 Type	Sex
1.55	1	0
1.6	0	1
1.72	1	0
1.5	2	1
1.85	3	0

2.3 Exporting-to and Importing-from Other Data Files

Data can be saved (exported) to and read (imported) from a number of different sources. Some of the common data files SPSS supports are: Excel data files: *.xls*, *.xlsx*; Text files: *.txt*, *.csv*, *.dat*; Stata data files: *.dta*.

Example 2.3. Saving into a different format: Let us save the *Student.sav* data in an excel file. The procedure is as follows.

1. From the *Menu* bar, click on **File** → **Save As**.
2. In the **Save Data As** dialogue box, from the **Save as type:** drop-down list, select the extension 'Excel 2007 through 2010 (*.xlsx*)'. If not changed, SPSS by default saves in 'SPSS Statistics (*.sav*)' format.
3. From the **Look in:** drop-down list, select your preferred location path where the file will be saved.
4. Then, in the **File name:** field, write a data file name, say, *Stud*.
5. Lastly, click on the **Save** tab.

Example 2.4. Opening a non-SPSS data file: Let us open the previously saved excel data file into SPSS.

1. From the *Menu* bar, click on **File** → **Open** → **Data**.

2. In the **Open Data** dialogue box, from the **Files of type:** drop-down list, select an extension, 'Excel (*.xls*, *.xlsx*, *.xlsm*)', (possibly 'All Files (*.*)'). If not changed, SPSS automatically searches the default extension 'SPSS Statistics (*.sav*)'.

3. From the **Look in:** drop-down list, select the location path where the file is saved.

4. Of the list of data files, if any, click on the data file name to be opened, that is, *Stud.xlxs*.

5. Click on the **Open** tab and then the **OK** tab.

Example 2.5. In the folder given to you, there is an excel data file named *JUSH_HAART.xlsx*. Import this data to SPSS. Then, give the variables definitions in the table below and save it by giving a similar file name as the excel file.

Variable Name	**Variable Label**	**Value Label**
CardNum	Patient's Card Number	
Age	Age	
Sex	Sex	
Wei	Weight	
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=Loss-to-follow
Defaulter	Dropped Out Patient	0=Active, 1=Defaulted
Days	Survival Time (Days)	
SurvTime	Survival Time (Months)	

Chapter 3

Basic Data Management

After creating a new data file or opening an existing data file, it is typically essential to examine the data and identify possible data processing problems (errors). Data processing errors[1] are errors that occur after the data have been collected. Examples of data processing errors include:

- Coding errors (e.g., groups of marital status gets improperly coded because of changes in the coding scheme)
- Routing errors (e.g., the interviewer asks the wrong question or asks questions in the wrong order)
- Consistency errors (contradictory responses, such as the reporting of a pregnancy after the respondent has identified himself as a male)
- Range errors (responses outside of the range of plausible answers, such as a reported age of 290)
- Duplicating errors (a single case might be entered accidentally more than once)
- Transpositions (e.g., 19 becomes 91 during data entry)
- Copying errors (e.g., 0 (zero) becomes O during data entry)

Data management takes place during all stages of a study which includes all aspects in planning the data needs of the study, data collection, data entry, data validation and checking, data manipulation, data files backup and data documentation. The objective is to create a reliable database containing high quality data, that is, without introducing data processing errors. Therefore, to prevent data processing errors, the stage at which the errors occurred must be identified and then the problem should be corrected. Some of the mechanisms are:

- Manual checking during data collection (e.g., checks for completeness, handwriting legibility)
- Range and consistency checking during data entry (e.g., preventing impossible results, such as ages greater than 110)

[1] This is distinct from measurement errors, which are differences between the true state of affairs and what appears on the data collection form.

- Double entry and validation following data entry
- Data analysis screening for outliers during data analysis
- Identifying cases containing duplicate information and deleting them from the data file

3.1 Preliminary Data Analysis

Before directly going to analyse data, it is essential to look at the details of the data at a glance. The question is "Does the data make sense?" out of range, missing, illogical/implausible values, consistency with other variables. In fact, it is not necessary to print out all the details of the data set. The basic rule is printing frequencies for categorical variables and descriptive statistics for continuous variables. These two printouts can be used as a primary references and give a picture of the overall shape of the data. That is,

- How much are missing?
- Which variables have missing data?
- Any variable has value(s) which seem unusual/implausible? Example: Age of 150.
- Assess internal consistency. Example: pregnancy and gender.

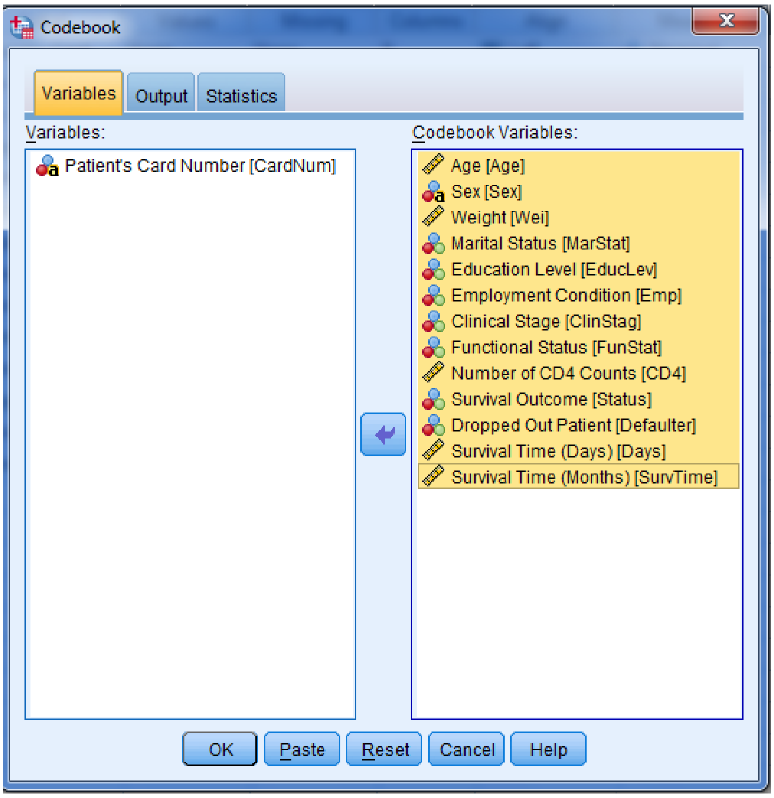
The Codebook Procedure

Codebook reports the dictionary information - such as variable names, variable labels, value labels, missing values - and summary statistics for the specified variables. For nominal and ordinal variables, summary statistics include counts and percents. For scale variables, summary statistics include mean, standard deviation, and quartiles.

From the *Menu* bar, click on **Analyze** → **Reports** → **Codebook**. In the **Codebook** dialogue box, enter at least one variable to the **Codebook Variables:** box.

Example 3.1. Observe the overall shape of the *JUSH_HAART.sav* data a using the **Codebook** procedure.

As shown below, all variables except **CardNum** are entered into the **Codebook Variables:** box. You can also specify the information you need under the **Output** and **Statistics** tabs.



Then, click on the **OK** tab and examine the results on the *Output* window.

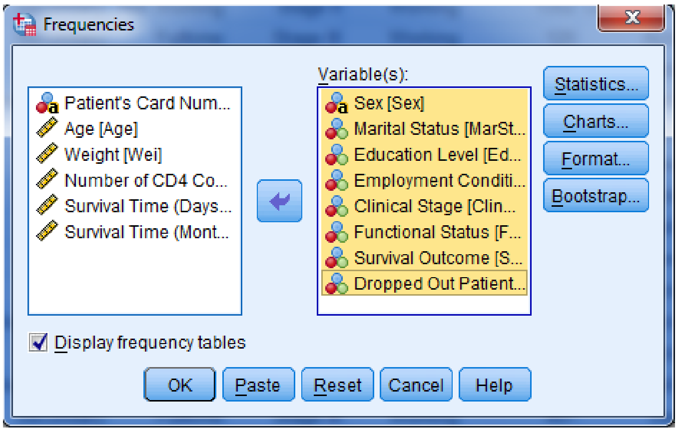
The Frequencies Procedure

The **Frequencies** procedure provides statistics and graphical displays that are useful for describing categorical variables. It is a good place to start looking at your data.

From the *Menu* bar, click on **Analyze** → **Descriptive Statistics** → **Frequencies**. In the **Frequencies** dialogue box, enter at least one categorical variable in the **Variable(s):** box.

Example 3.2. Examine all categorical variables in the *JUSH_HAART.sav* data a using the **Frequencies** procedure.

Here, all categorical and string variables are entered into the **Variable(s):** box as shown below.



Then, click on the **OK** tab and examine the results.

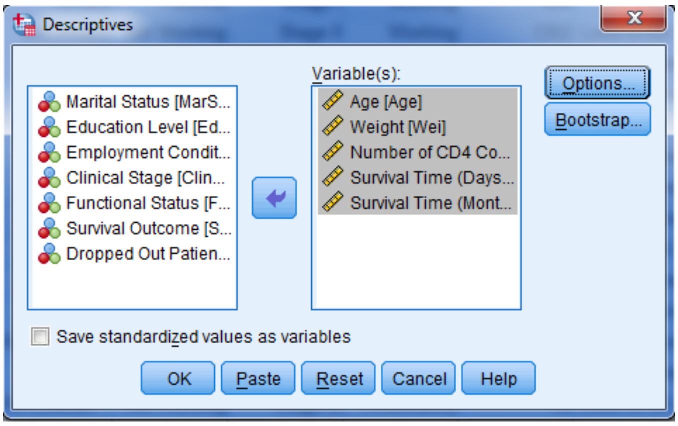
The Descriptives Procedure

The **Descriptives** procedure displays univariate summary statistics for continuous variables in a single table.

From the *Menu* bar, click on **Analyze** → **Descriptive Statistics** → **Descriptives**. In the **Descriptives** dialogue box, enter at least one quantitative variable in the **Variable(s):** box in the usual manner.

Example 3.3. Examine all quantitative variables in the *JUSH_HAART.sav* data a using the **Descriptives** procedure.

Now all quantitative variables are entered into the **Variable(s):** box.



Click on the **OK** tab and look at the output.

3.2 Manipulating Data

3.2.1 Protecting Original Data

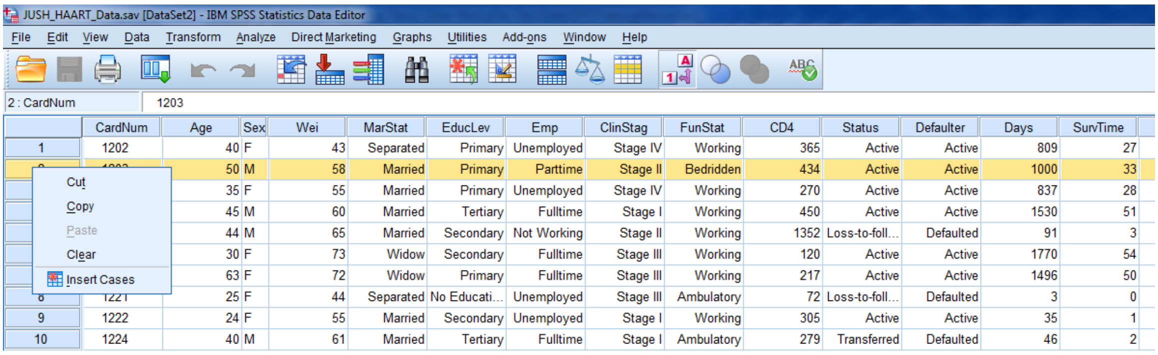
A data file can be marked as read-only so as to prevent the accidental modification of the original data. From the *Menu* bar, click on **File** → **Mark File Read Only**. If subsequent modifications are made to the data, the modified data can be saved only with a different filename; so the original data are not affected. The file permissions can be changed back to read/write by selecting **Mark File Read Write** from the **File** menu.

3.2.2 Inserting and Deleting Cases

To insert a new case, right click on the row number in the **Data View** sheet in which the new case is to be inserted and then click on **Insert Cases**. (Entering data in an empty row of the **Data View** automatically creates a new case.)

To delete a case, right click on the row number (case) to be deleted and click on **Clear**.

For example, to insert a new case on the second row or to delete the second case, just right click on row number 2 and then click on **Insert Cases** or **Clear**, respectively.

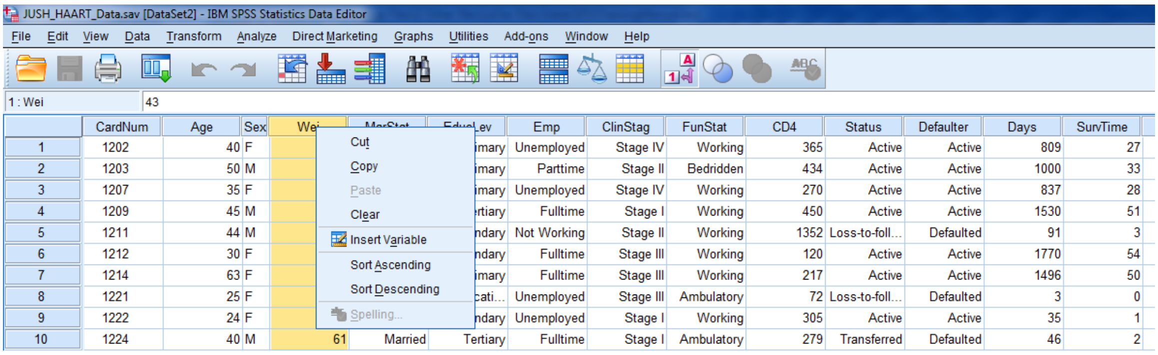


3.2.3 Inserting and Deleting Variables

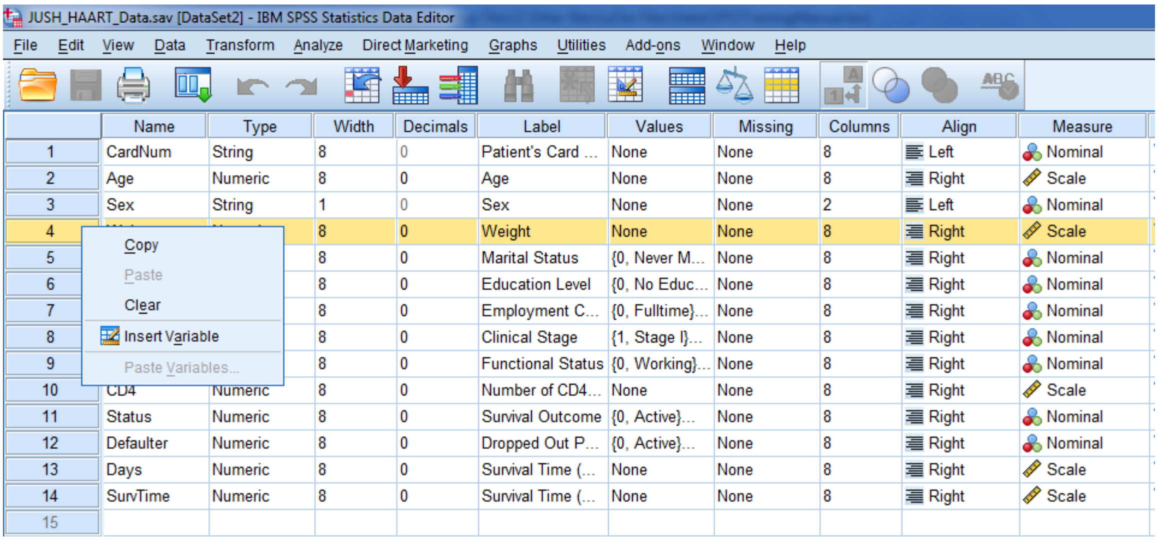
Inserting a new variable can be done using both sheets of the *Data Editor*. On the **Data View** sheet, click on the column that the new variable is to be inserted and then click on **Insert Variable**. Or in the **Variable View** sheet, right click on the row number that the new variable is to be inserted, and then click on **Insert Variable**. (Entering data in an empty column in the **Data View** or in an empty row in the **Variable View** automatically creates a new variable with a default name.)

To delete a variable, on the **Data View** sheet, right click on the variable name to be deleted and then click on **Clear**. Or in **Variable View** sheet, right click on the row number to be deleted, and then click on **Clear**.

For example, to insert a new variable between *Sex* and *Wei*, or to delete the *Wei* variable, just right click on *Wei* and then click on **Insert Variable** or **Clear**, respectively.



Or

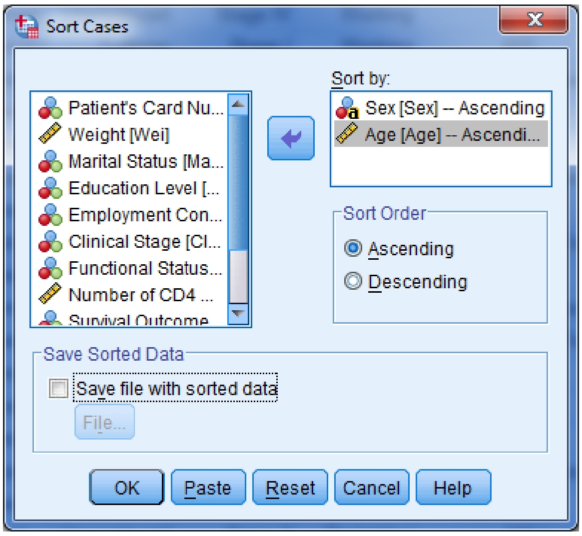


3.2.4 Sorting Cases

In order to sort the data, from the *Menu* bar, click on **Data** → **Sort Cases**. Then, in the **Sort Cases** dialogue box, enter the sorting variable(s) in the **Sort by:** box. If two or more variables are sorting variables, then the cases are sorted by each variable within categories of the preceding variable on the sort list.

Example 3.4. Sort *JUSH_HAART.sav* by *Sex* and *Age*.

Both *Sex* and *Age* should be entered in the **Sort by:** box of the **Sort Cases** dialogue box as follows.

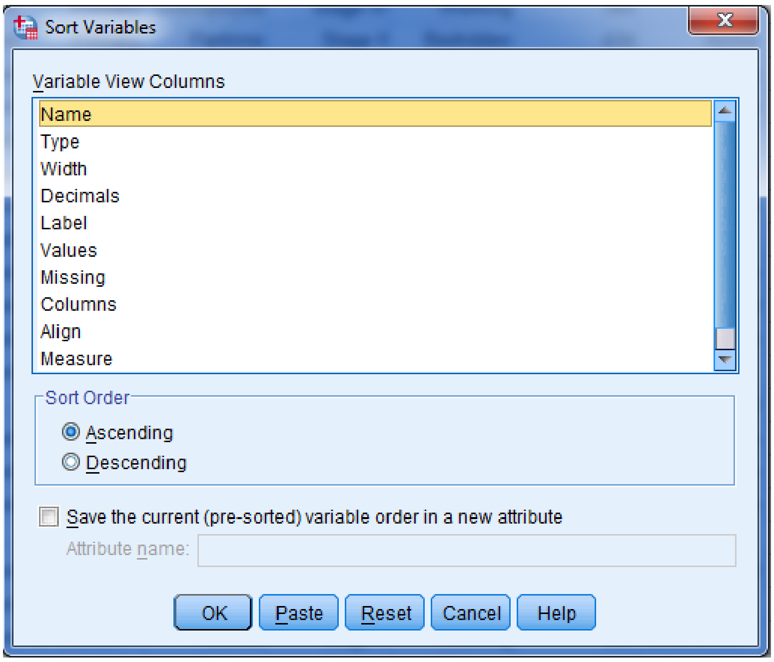


Here, *Sex* was the first variable entered, followed by *Age*; accordingly, the data will first be sorted by *Sex*, then, within each *Sex* category, the data will be sorted by *Age*.

Note: For string variables, uppercase letters precede their lowercase counterparts in sort order. For example, the string value "Yes" comes before "yes" in sort order.

3.2.5 Sorting Variables

In order to sort the variables, from the *Menu* bar, click on **Data** → **Sort Variables**.

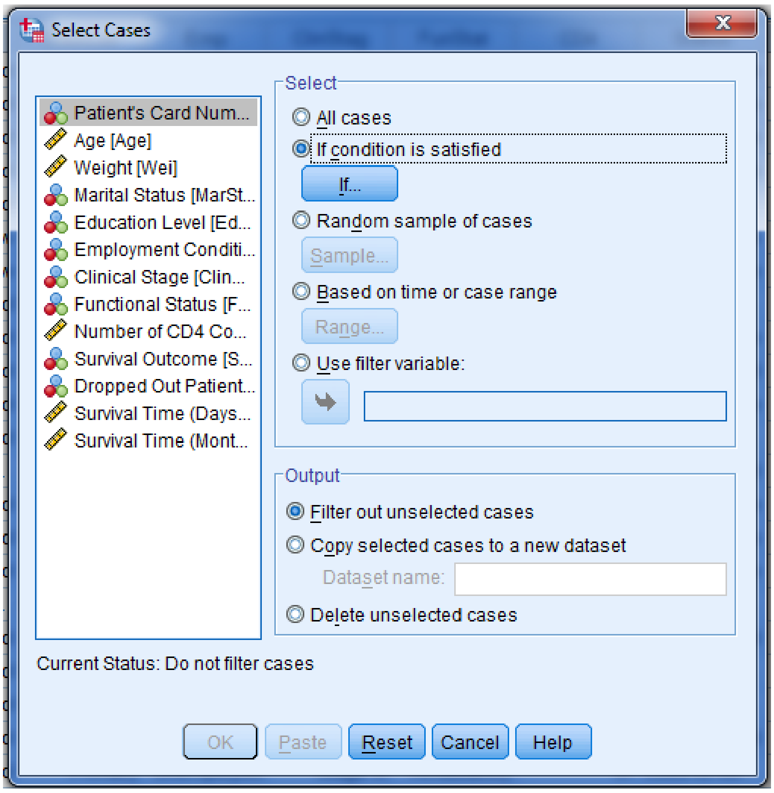


Here, the variables cannot only be sorted by their names but also by their **Variable View** characteristics.

3.2.6 Selecting Cases

Instead of just wanting to look at all possible values and observations for a particular variable, you can analyze a specific subset of the data by selecting only certain cases in which you are interested. How would you do that? You would use a conditional statement.

To select cases, from the *Menu* bar, click on **Data** → **Select Cases**. Then, the **Select Cases** dialogue box comes.



Under the **Output** options:

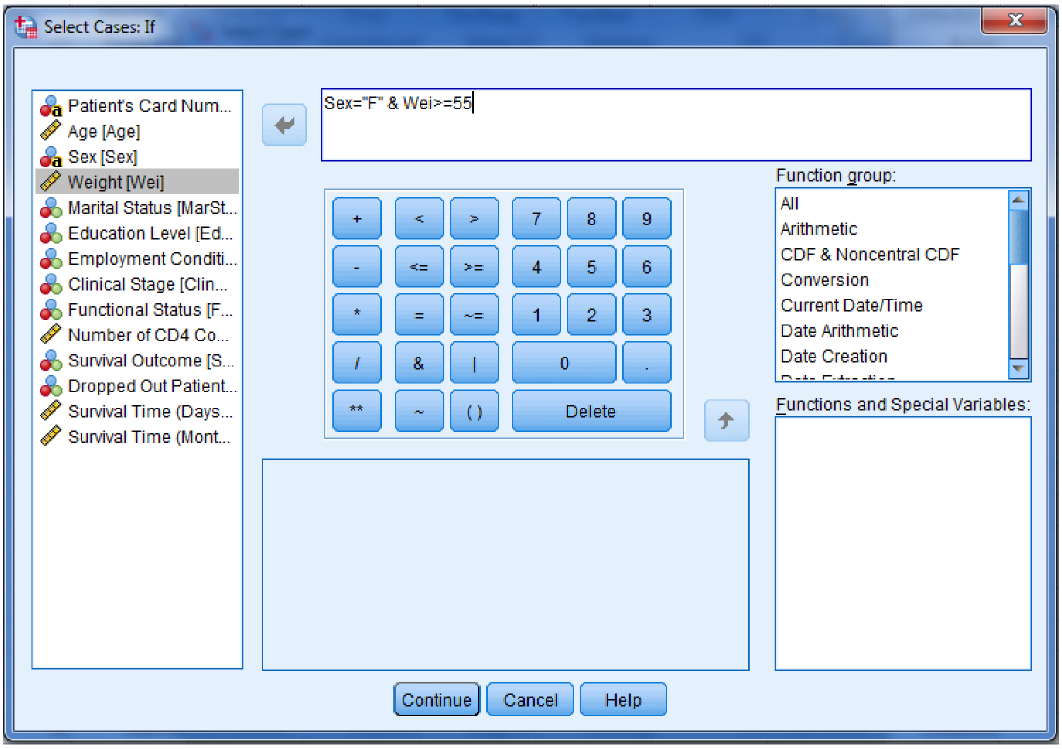
- **Filter**$\cdots$ indicates the unselected cases in the *Data Editor* by placing a slash over the row numbers and removes them from subsequent analyses until the **All Cases** under the **Select** option is reset, at which time all cases will again be active for analyses.

- **Copy**$\cdots$ saves the selected cases to a new data set.

- **Delete**$\cdots$ removes unselected cases from the working data set; be very careful with this option, because if the dataset is subsequently saved, these cases will be permanently deleted.

Example 3.5. Select (filter) female patients whose weight is greater than or equal to 55.

When the **If** button of the **Select Cases** dialogue box is clicked, the **Select Cases: If** dialogue box opens. Then, to select females whose weight is greater than or equal to 55, we do as follows.



Note that the string constants must be enclosed in quotation marks or apostrophes. Click on **Continue** and then **OK**. Then, the *Data Editor* window looks:

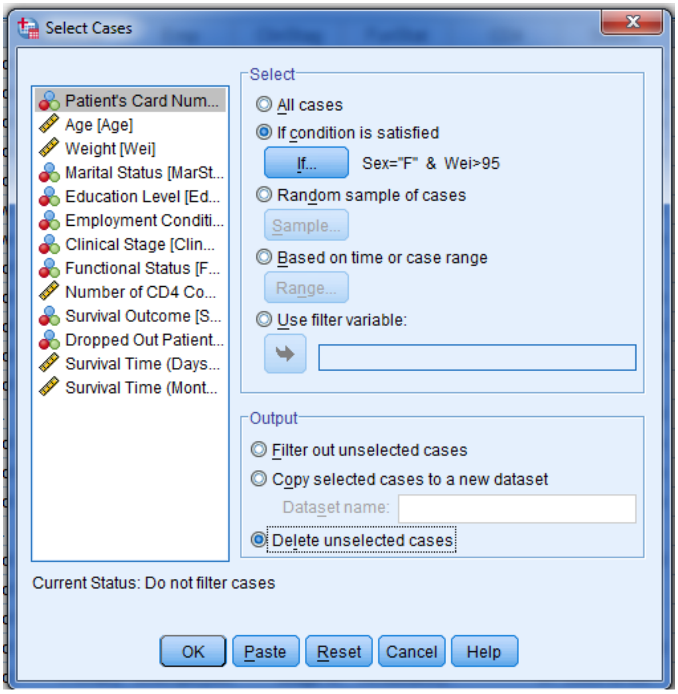
	CardNum	Age	Sex	Wei	MarStat	EducLev	Emp	ClinStag	FunStat	CD4	Status	Defaulter	Days	SurvTime	filter_$	var
~~1~~	1202	40	F	43	Separated	Primary	Unemployed	Stage IV	Working	365	Active	Active	809	27	Not Selected	
~~2~~	1203	50	M	58	Married	Primary	Parttime	Stage II	Bedridden	434	Active	Active	1000	33	Not Selected	
3	1207	35	F	55	Married	Primary	Unemployed	Stage IV	Working	270	Active	Active	837	28	Selected	
~~4~~	1209	45	M	60	Married	Tertiary	Fulltime	Stage I	Working	450	Active	Active	1530	51	Not Selected	
~~5~~	1211	44	M	65	Married	Secondary	Not Working	Stage II	Working	1352	Loss-to-foll...	Defaulted	91	3	Not Selected	
6	1212	30	F	73	Widow	Secondary	Fulltime	Stage III	Working	120	Active	Active	1770	54	Selected	
7	1214	63	F	72	Widow	Primary	Fulltime	Stage III	Working	217	Active	Active	1496	50	Selected	
~~8~~	1221	25	F	44	Separated	No Educati...	Unemployed	Stage III	Ambulatory	72	Loss-to-foll...	Defaulted	3	0	Not Selected	
9	1222	24	F	55	Married	Secondary	Unemployed	Stage I	Working	305	Active	Active	35	1	Selected	
~~10~~	1224	40	M	61	Married	Tertiary	Fulltime	Stage I	Ambulatory	279	Transferred	Defaulted	46	2	Not Selected	
~~11~~	1228	53	F	50	Separated	Secondary	Unemployed	Stage III	Ambulatory	436	Active	Active	1255	42	Not Selected	
~~12~~	1229	39	F	50	Married	Primary	Fulltime	Stage I	Ambulatory	213	Loss-to-foll...	Defaulted	33	1	Not Selected	
~~13~~	1230	33	M	62	Never Marri...	Secondary	Fulltime	Stage III	Working	68	Transferred	Defaulted	226	8	Not Selected	
14	1231	30	F	61	Married	Primary	Fulltime	Stage I	Working	1003	Active	Active	573	19	Selected	
~~15~~	1240	30	F	48	Married	No Educati...	Not Working	Stage I	Working	516	Active	Active	1022	34	Not Selected	
16	1242	27	F	63	Married	Secondary	Fulltime	Stage I	Working	368	Active	Active	986	33	Selected	
~~17~~	1244	43	F	46	Never Marri...	No Educati...	Not Working	Stage III	Ambulatory	38	Loss-to-foll...	Defaulted	1302	43	Not Selected	
~~18~~	1246	31	M	57	Married	Secondary	Fulltime	Stage III	Working	407	Dead	Defaulted	445	15	Not Selected	
~~19~~	1249	25	F	48	Separated	No Educati...	Unemployed	Stage II	Bedridden	122	Loss-to-foll...	Defaulted	6	0	Not Selected	
20	1250	30	F	56	Divorced	No Educati...	Fulltime	Stage II	Working	504	Loss-to-foll...	Defaulted	614	20	Selected	
21	1251	27	F	55	Married	Secondary	Fulltime	Stage II	Working	197	Transferred	Defaulted	921	31	Selected	
~~22~~	1252	30	F	36	Divorced	Primary	Unemployed	Stage III	Ambulatory	57	Loss-to-foll...	Defaulted	4	0	Not Selected	
23	1253	27	F	58	Married	No Educati...	Parttime	Stage IV	Working	474	Loss-to-foll...	Defaulted	1498	50	Selected	

Data View | Variable View

From now onwards, those cases having a slash over on the row numbers are deactivated, that means, they will not be included in the subsequent analysis. Hence, do not forget to reset **All Cases** under the **Select** option of **Select Cases** dialogue box which makes all cases active for analyses.

Example 3.6. Select female patients whose weight is greater than 95 and remove the unselected cases.

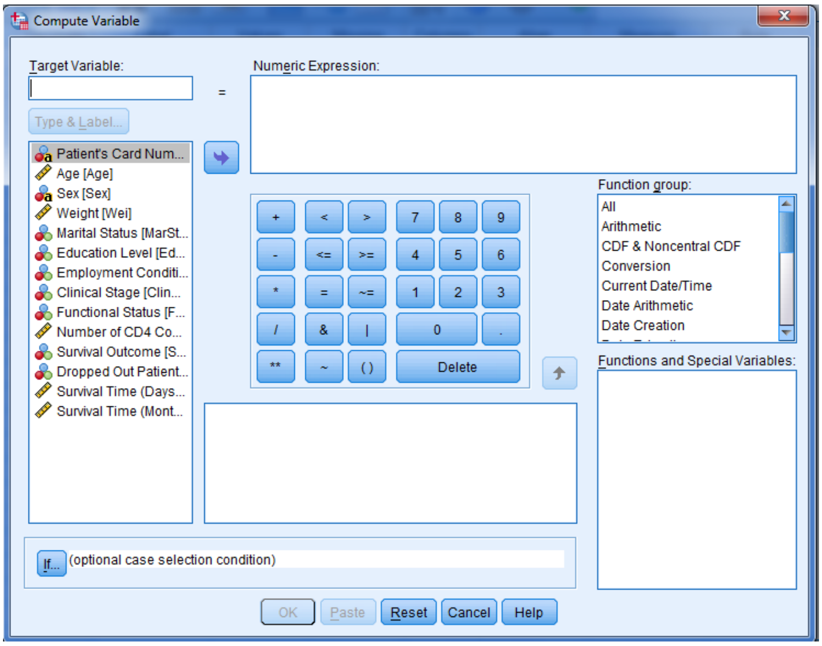
To select female patients whose weight is greater than 95 and remove the unselected cases, the **Select Cases** dialogue box looks:



After clicking the **OK** tab, you can easily observe that there are only 2 cases as shown in the **Data View** sheet of the *Data Editor*. All the unselected cases are removed from the working file.

3.2.7 Creating New Variables

Variable transformation is a way of creating new variables using existing continuous variables and formulae. To create a new variable, from the *Menu* bar, click on **Transform** → **Compute**. This opens the **Compute Variable** dialog box.



Example 3.7. Create new variable by taking the square root of the *CD4* variable.

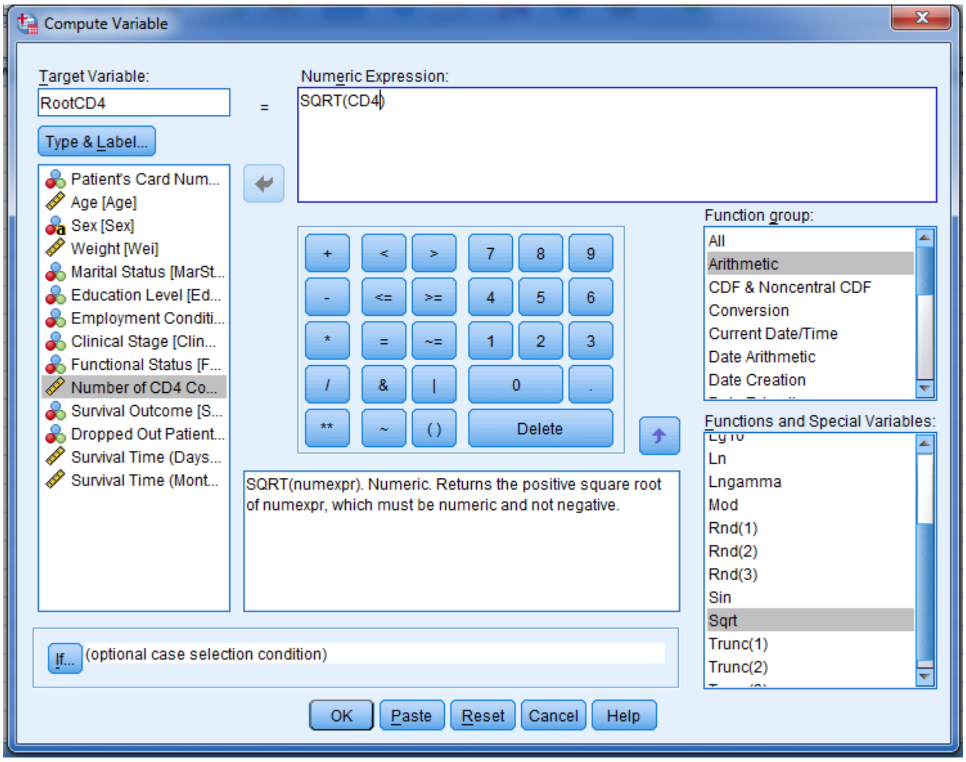
Now to compute the square root of the *CD4* variable,

1. First, write the new variable name, *RootCD4* in the **Target Variable:** field.

2. Next from the **Function group:** list select **Arithmetic**.

3. And then from **Functions**··· list select **Sqrt** and enter into the **Numeric Expression:** box (any formula can be written as function of existing variables and/or numbers).

4. Lastly, inside the brackets of the square root, enter the *CD4* variable.



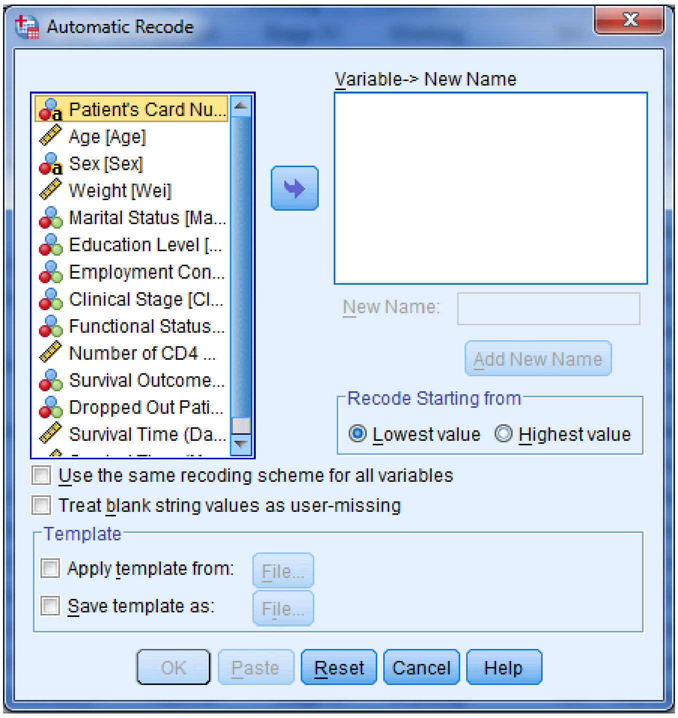
You can also specify the type and label for the new variable. The **If** button can be used to case selection condition.

Example 3.8. Generate a new variable *AgeSquare* by squaring the *Age*.

Example 3.9. Generate a new variable *LogCD4* by taking the logarithm of *CD4*.

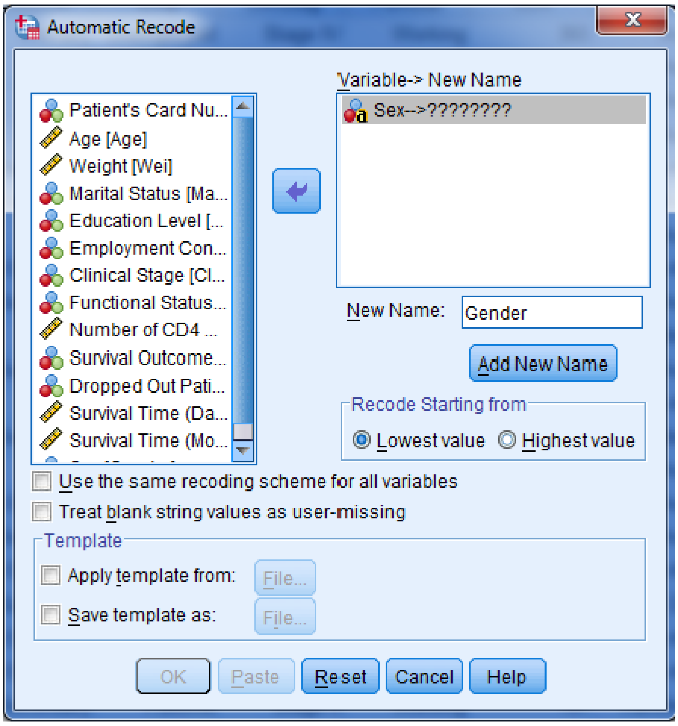
3.2.8 Recoding a String Variable

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. From the *Menu* bar, by clicking on **Transform** → **Automatic Recode**, the **Automatic Recode** dialogue box opens. In the **Variable->New Name** box, enter the string variable.



Example 3.10. Automatically recode the string variable *Sex*.

In the **Automatic Recode** dialogue box, enter *Sex* in the **Variable->New Name** box.



Next in the **New Name:** field write *Gender* and click on the **Add New Name** button to add the new name into **Variable->New Name** box. Click on **OK**. This automatically assigns the values 1 for 'F' and 2 for 'M' and labels to the categories of *Gender*.

3.2.9 Recoding a Categorical Variable

For *Gender*, the numeric values for 'Female' and 'Male' patients are 1 and 2. Suppose, that is not what we want. We want the typical 0='F', 1='M' setup. How might we do this? We will be looking how to recode this categorical variable. There are two options available for recoding variables. The first option is recoding values into the same variable which eliminates all record of the original values and the second one is creating a new variable containing the recoded values.

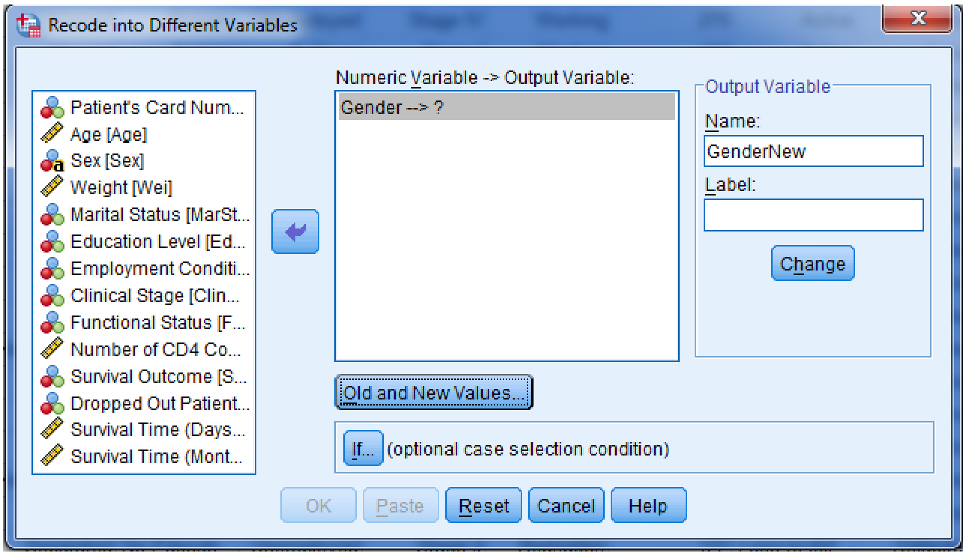
Note: NEVER ever, ever, EVER recode a variable into the same variable name. For one thing it deletes the existing data and for another it destroys the history of the data. Always create a new variable to contain the new codes.

From the *Menu* bar, by clicking on **Transform** → **Recode into Different Variables**, then a dialogue box opens.

Example 3.11. Recode *Gender* so that 0 stands for 'F' and 1 stands for 'M'.

In the **Recode into Different Variables** dialogue box,

1. In the **Numeric Variable->Output Variable** box, enter *Gender*.

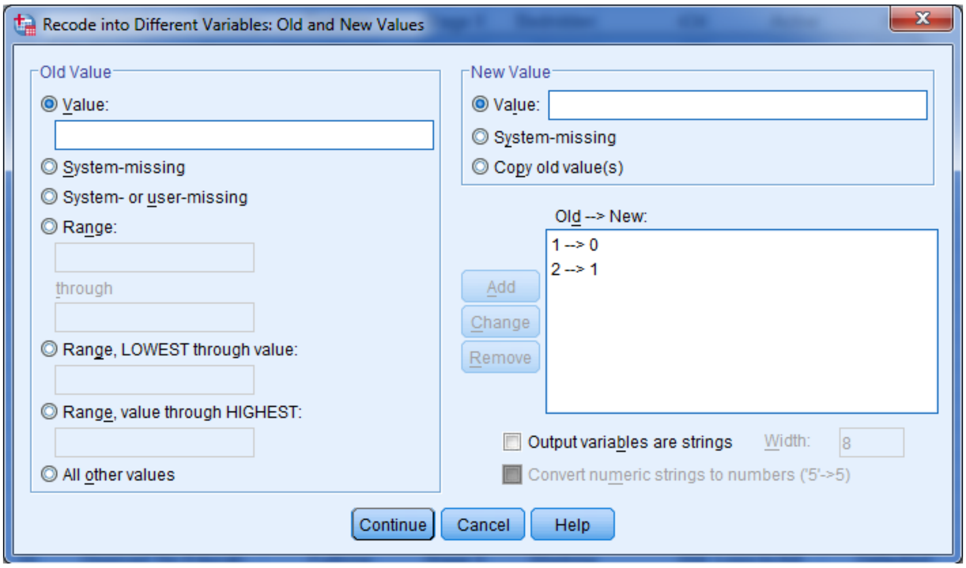


2. Next in the **Output Variable** option **Name:** field write a new variable name, say, *GenderNew* and click on the **Change** button to add the new name into **Numeric Variable->Output Variable** box.

3. When you click on the **Old and New Values** button, the following dialogue box comes.

   (a) In the **Old Value** option **Value:** field type 1 and in the **New Value** option **Value:** field type 0, then click on the **Add** tab.

   (b) Next in the **Old Value** option **Value:** field type 2 and in the **New Value** option **Value:** field type 1, then click on the **Add** tab.



(c) Click on the **Continue** tab and **OK**.

Example 3.12. Recall the variable Employment Condition (*Emp*) variable is coded as 0=*Full-time*, 1=*Part-time*, 2=*Not Working* and 3=*Unemployed*. Now recode this variable using 0=*Working* (Full-time and Part-time), 1=*Not Working* and 2=*Unemployed* setup.

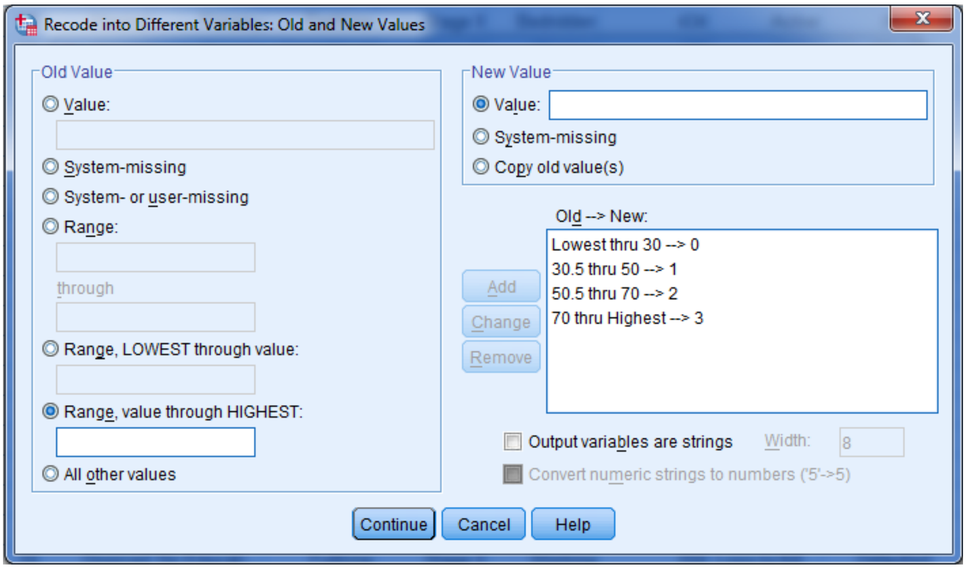
3.2.10 Recoding a Continuous Variable

Some times, it is also useful to collapse a continuous variable into categorical groups. From the *Menu* bar, by clicking on **Transform** → **Recode into Different Variables**, then the usual dialogue box opens.

Example 3.13. Categorize the *Wei* variable into four categories: 0=Min-30, 1=30.5-50, 2=50.5-70 and 3=70.5-Max. Then, create their value labels.

In the **Recode into Different Variables** dialog box,

1. In the **Numeric Variable->Output Variable** box, enter *Wei*.
2. Next in the **Output Variable** option **Name:** field write a new variable name, say, *WeiCat* and click on the **Change** button to add the new name into **Numeric Variable->Output Variable** box.
3. Click on the **Old and New Values** button.
   (a) In the **Old Value** option in the **Range, LOWEST through Value:** field type 30 and in the **New Value** option **Value:** field type 0, then click on the **Add** tab.
   (b) Next in the **Old Value** option **Range:** fields type 30.5 and 50, respectively, and in the **New Value** option **Value:** field type 1, then click on the **Add** tab.
   (c) Thirdly, in the **Old Value** option **Range:** fields type 50.5 and 70, respectively, and in the **New Value** option **Value:** field type 2, then click on the **Add** tab.
   (d) Lastly, in the **Old Value** option in the **Range, value through HIGHEST:** field type 70 and in the **New Value** option **Value:** field type 3, then click on the **Add** tab.



(e) Click on **Continue** and **OK**.

Example 3.14. Categorize *Age* into three categories: 0=Min-29, 1=30-39 and 2=40-Max. Then, create their value labels.

3.3 Combining Data Sets

It is often needed to merge several databases into one. There are two main ways of merging data sets. The first situation is when we have two databases with the *same variables but different cases* and the second situation is when we have two data sets with *same cases but different variables.*

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

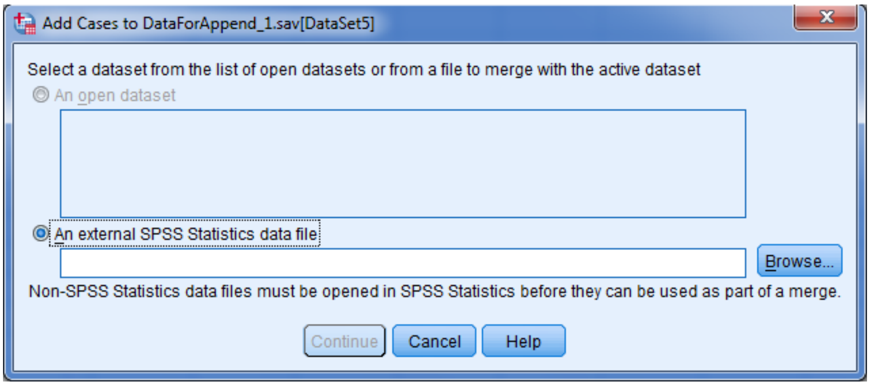
3.3.1 Adding Cases (Observations)

Here, the two data files are considered to have *different observations but same variables.* Hence, observations from the using file are added to the end of the working data file, that is, the files are stacked vertically. But, be sure that each variable should have *same name and same data type* in both data files.

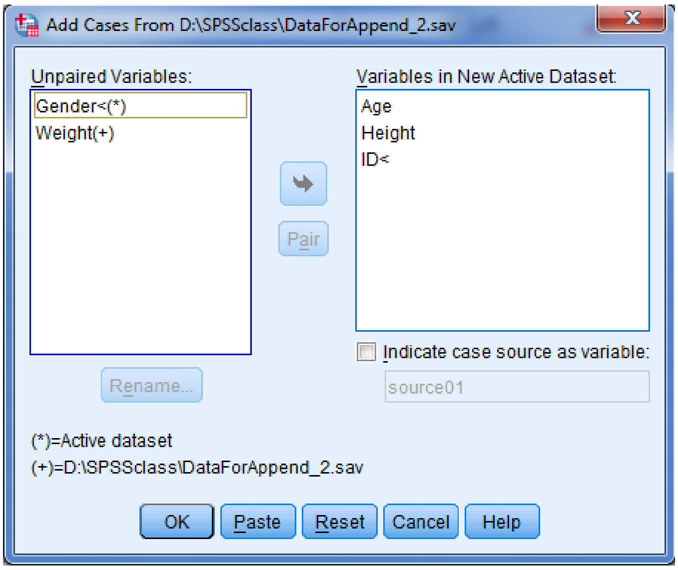
From the *Menu* bar, click on **Data** → **Merge Files** → **Add Cases**. This opens the **Add Cases to**$\cdots$ dialogue box having the **Browse** button that helps to locate where the using file is saved.

Example 3.15. In the folder given to you, there are two data files named *DataForAppend_1.sav* and *DataForAppend_2.sav* having some same variables but different cases. Add the cases from *DataForAppend_2.sav* to *DataForAppend_1.sav*.

First, open the *DataForAppend_1.sav* data and click on **Data** → **Merge Files** → **Add Cases**. In the **Add Cases to**$\cdots$ dialogue box, click on the **Browse** button to locate where the using file (*DataForAppend_2.sav*) is saved



After browsing and selecting the using file, click on the **Continue** tab. Then the **Add Cases From**··· dialogue box with the list of variables in two boxes appears as shown below. The **Unpaired Variables:** box contains variables to be excluded from the new appended data file (due to the difference in names and/or type) while the **Variables in New···:** box contains variables to be included in the new appended data file.



Click on the **OK** tab.

Note:

- If the same information is recorded under different variable names in the two files, you can create a pair from the **Unpaired Variables:** list. Select the two variables on the list and click on **Pair**.

- To include an unpaired variable from one file without pairing it with a variable from the other file, select the variable from the **Unpaired Variables:** list and add it to **Variables in New···:** list. Any unpaired variable included in the merged file will contain missing data for cases from the file that does not contain that variable.

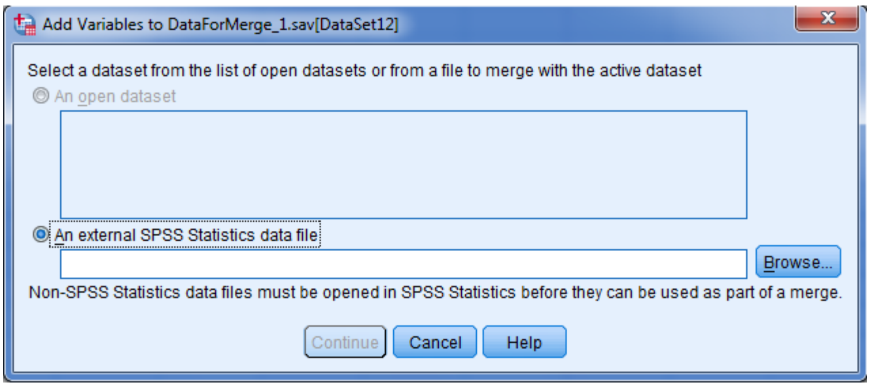
3.3.2 Adding Variables

Unlike the previous merging, the two data files are now considered to have *different variables but same observations*. The additional variables from the using file are added to the data file in memory (the files are stacked horizontally). A necessary condition is that both data sets should have a unique identifier of each observation which might consist of a single variable or a series of variables. In other words, both files should have a matching variable (or variables) that is (are) used to associate an observation from the master file with an observation in the using file. Before trying to merge, both data sets should be sorted by the matching variable. If two or more variables are used to match cases, the two data files must be sorted by ascending order of these key variables.

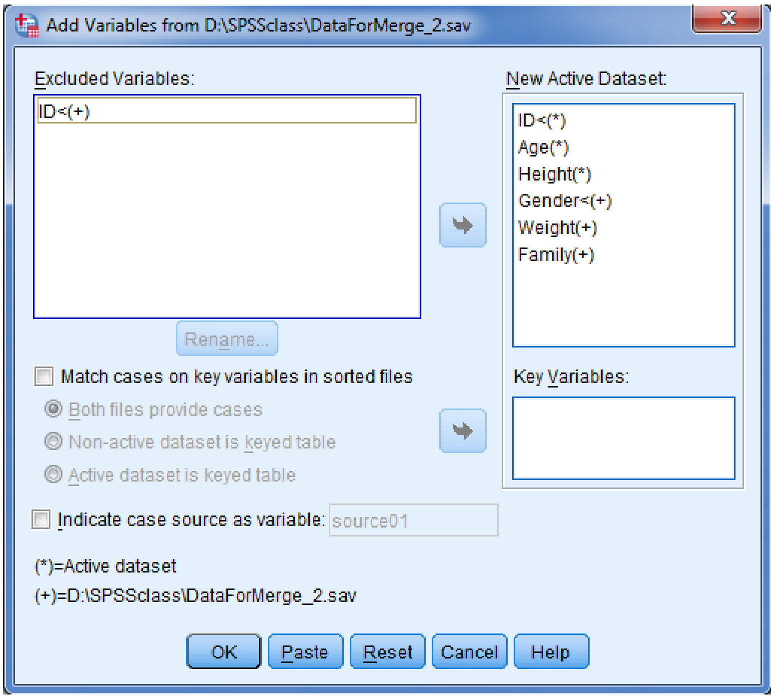
From the *Menu* bar, click on **Data** → **Merge Files** → **Add Variables**. This opens the **Add Variables to**··· dialogue box having the **Browse** button that helps to locate where the using file is saved.

Example 3.16. There are two data files named *DataForMerge_1.sav* and *DataForMerge_2.sav*, in the folder given to you. The **ID** variable is an identifier (matching variable) in both data files. Merge the variables from *DataForMerge_2.sav* to *DataForMerge_1.sav*.

First, open *DataForMerge_1.sav* data. In the **Add Variables to**··· dialogue box, click on the **Browse** button to locate where the using file (*DataForMerge_2.sav*) is saved.



After browsing and selecting the file to be merged, click on the **Continue** tab. Then the **Add Variables from**··· dialogue box with the list of variables appears.



The **Excluded Variables:** box contains variables to be excluded from the new merged data file. Variable names in the second data file that duplicate variable names in the working data file are excluded by default because it assumes that these variables contain duplicate information. To include an excluded variable with a duplicate name to the merged file, rename it and add it to the **New Active Dataset:** box.

In the case of two or more key variables, tick on the **Match cases on key**$\cdots$ and enter them to **Key Variables:** list. But the order of these variables on the **Key Variables:** list must be the same as their sort sequence.

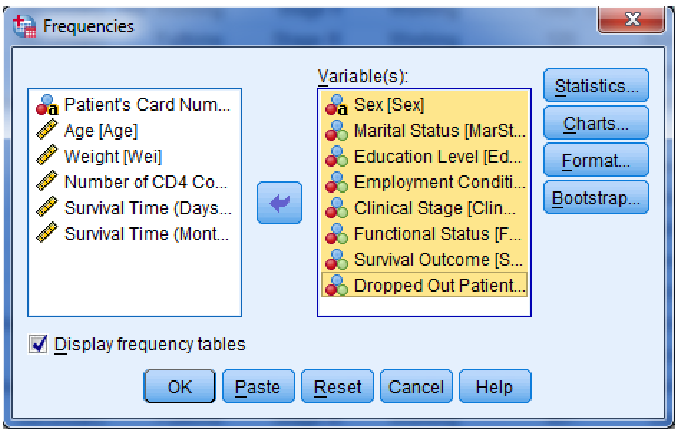
Chapter 4

Descriptive Statistics

4.1 One-Way Frequency Tables

Frequency tables are used to summarize categorical variables. To construct one-way frequency tables, from the *Menu* bar, click on **Analyze** → **Descriptive Statistics** → **Frequencies**. In the **Frequencies** dialogue box, enter at least one least categorical variable in the **Variable(s):** box.

Example 4.1. Obtain the one-way frequency tables for all categorical variables in the *JUSH_HAART.sav* data.



The first two frequency tables of the output are:

Sex

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	F	930	63.5	63.5	63.5
	M	534	36.5	36.5	100.0
	Total	1464	100.0	100.0	

Marital Status

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Never Married	293	20.0	20.1	20.1
	Married	739	50.5	50.6	70.7
	Divorced	134	9.2	9.2	79.9
	Separated	140	9.6	9.6	89.5
	Widowed	154	10.5	10.5	100.0
	Total	1460	99.7	100.0	
Missing	System	4	.3		
Total		1464	100.0		

The result indicates that 930 (63.5%) of the patients were females and the remaining 534 (36.5%) were males. Regarding the marital status of the patients, 293 (20%), 739 (50.5%), 134 (9.2%), 140 (9.6%) and 154 (10.5%) were never married, married, divorced, separated and widowed, respectively. But note that there are 4 (0.3%) missing observations.

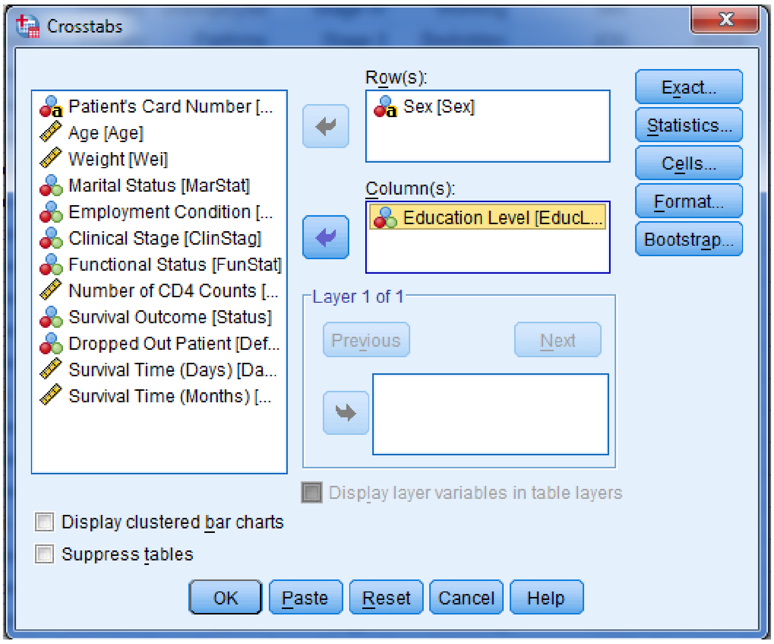
4.2 Contingency Tables (Cross-Tabulations)

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). The **Crosstabs** procedure forms two-way and multi-way contingency tables, and provides a variety of tests and measures of association for two-way contingency tables only.

For constructing cross-tabulations (contingency tables), from the *Menu* bar, click on **Analyze** → **Descriptive Statistics** → **Crosstabs**. In the **Crosstabs** dialogue box, enter at least one categorical variable in the **Row(s):** box and another one in the **Column(s):** box.

Example 4.2. Construct the cross-tabulation of *Sex* and *Education Level*, and examine the frequencies.

In the **Crosstabs** dialogue box, enter *Sex* in **Row(s):** box and *Education Level* in **Column(s):** box as show below and click the **OK** tab.



The crosstab result is:

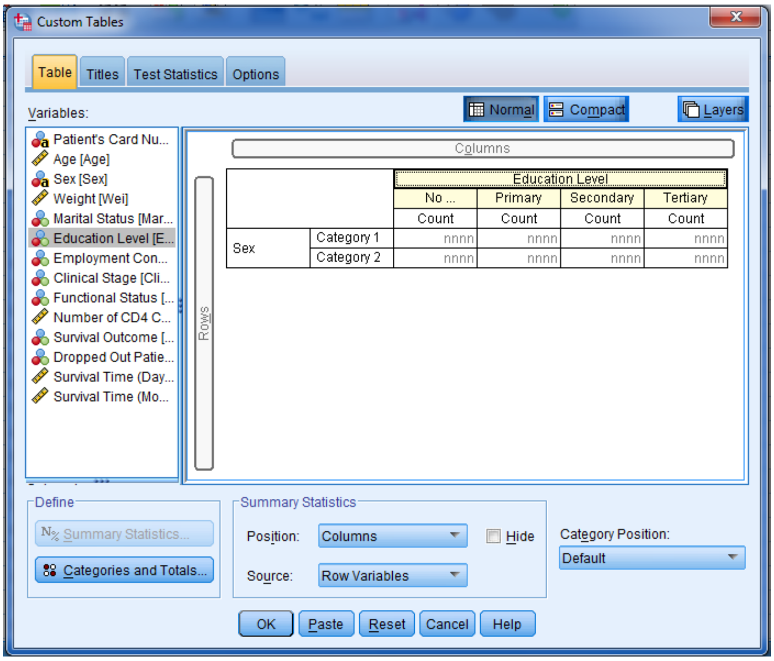
Sex * Education Level Crosstabulation

Count

		Education Level				
		No Education	Primary	Secondary	Tertiary	Total
Sex	F	211	325	316	73	925
	M	86	190	176	82	534
Total		297	515	492	155	1459

There were 211 female patients who did not have education, 325 females with primary education, $\cdots$. Of the total 534 males, 86 of them had no education, 190 of them were having primary education, $\cdots$.

SPSS can also do custom tables which describe the relationship between variables in a table of frequencies. These tables can either be simple two-way tables or multi-way tables. To do so, from the *Menu* bar, click on **Analyze** $\rightarrow$ **Tables** $\rightarrow$ **Custom Tables**. In the **Custom Tables** dialogue box, select and drag the variable(s) to be in the row and column and click the **OK** tab.



4.3 Measures of Central Tendency and Variation

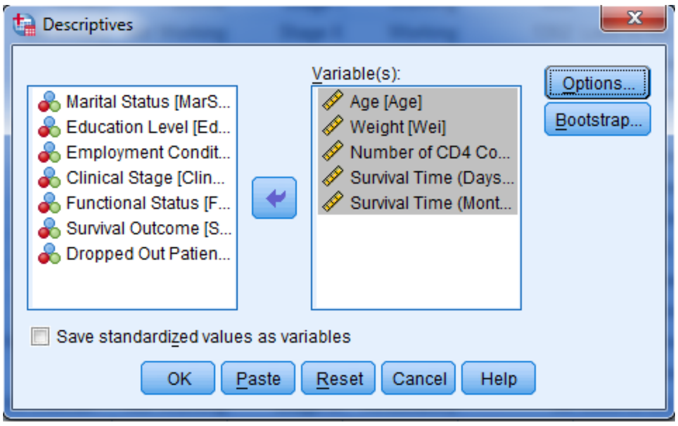
For quantitative variables, it is necessary to calculate certain indicators like measures of central tendency and measures of variation.

4.3.1 Basic Descriptive Statistics

The **Descriptives** procedure displays univariate summary statistics for scale variables in a single table. From the *Menu* bar, click on **Analyze** → **Descriptive Statistics** → **Descriptives**. In the **Descriptives** dialogue box, enter at least one quantitative variable in the **Variable(s):** box in the usual manner.

Example 4.3. Obtain descriptive statistics for all quantitative (scale) variables of the *JUSH_HAART.sav* data.

In the **Descriptives** dialogue box, all quantitative variables are entered in **Variable(s):** box as follows.



Here is the output:

Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation
Age	1464	18	85	34.01	9.160
Weight	1460	16	96	51.87	10.193
Number of CD4 Counts	1464	1	1914	198.19	171.240
Survival Time (Days)	1464	1	2141	761.66	511.852
Survival Time (Months)	1464	.03	54.00	25.1977	16.70695
Valid N (listwise)	1460				

The minimum and maximum ages of the 1464 patients are 18 and 85 years, respectively. The average age is 34.01 years with a standard deviation of 9.16 years. The average weight is 51.87 kilograms with a standard deviation of 10.19 kilograms (note that these values are calculated from 1460 patients which means there are 4 weight missing values) with a minimum weight of 16 and a maximum weight of 96 kilograms. The average number of CD4 counts is 198.19 with a standard deviation of 171.240. The same is true for the remaining variables.

Also, by clicking on the **Options** button of **Descriptives** dialogue box, we can select additional measures like sum, variance, range, standard error for the mean, kurtosis and skewness.

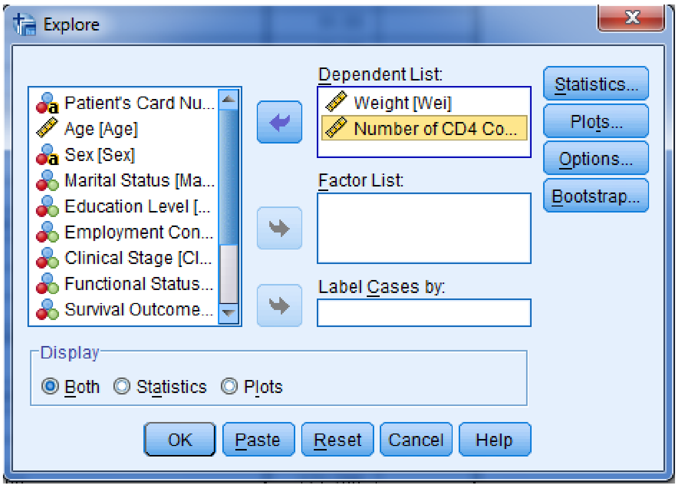
4.3.2 Exploring More Descriptive Statistics

The **Explore** procedure produces summary statistics and graphical displays, either for all cases or separately for groups of cases. It is used for data screening, outlier identification, normality assumption checking, and characterizing differences among groups of cases. Data screening may help to identify the presence of unusual values, extreme values, gaps in the data or other peculiarities.

From the *Menu* bar, click on **Analyze** → **Descriptive Statistics** → **Explore**. In the **Explore** dialogue box, enter at least one quantitative variable in the **Dependent List:** box. Sometimes, it is also necessary to explore the data for a group of cases, that is, exploring the scale variable(s) within each category of a categorical variable.

Example 4.4. Obtain descriptive statistics using the **Explore** procedure for the *Wei* and *CD4* variables.

In the **Explore** dialogue box, enter both *Wei* and *CD4* in the **Dependent List:** box. It is also possible to select an identification variable to label cases and enter to **Label Cases By:** box.



Also, the five largest and five smallest values with case labels can be displayed if the **Outliers** option is selected under the **Statistics** tab.

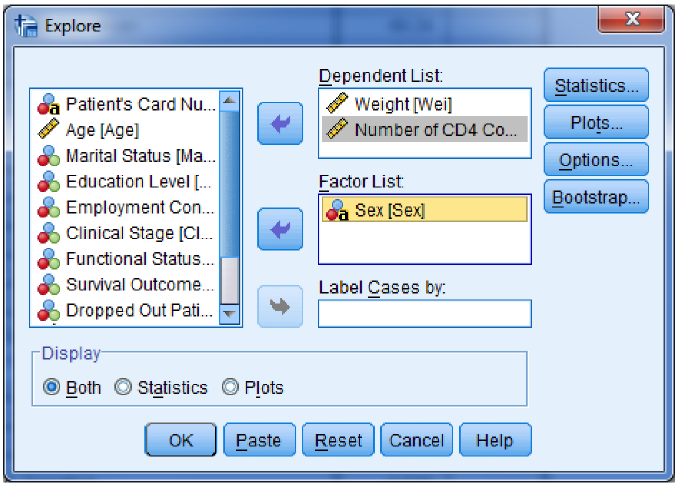
The partial result looks:

Descriptives

			Statistic	Std. Error
Weight	Mean		51.87	.267
	95% Confidence Interval for Mean	Lower Bound	51.34	
		Upper Bound	52.39	
	5% Trimmed Mean		51.55	
	Median		51.00	
	Variance		103.905	
	Std. Deviation		10.193	
	Minimum		16	
	Maximum		96	
	Range		80	
	Interquartile Range		13	
	Skewness		.519	.064
	Kurtosis		.909	.128
Number of CD4 Counts	Mean		198.69	4.481
	95% Confidence Interval for Mean	Lower Bound	189.90	
		Upper Bound	207.48	
	5% Trimmed Mean		180.60	
	Median		159.00	
	Variance		29311.335	
	Std. Deviation		171.206	
	Minimum		1	
	Maximum		1914	
	Range		1913	
	Interquartile Range		180	
	Skewness		2.877	.064
	Kurtosis		16.429	.128

Example 4.5. Now, explore the *Wei* and *CD4* variables within each category of *Sex*.

Now in the **Explore** dialogue box, enter both *Wei* and *CD4* in the **Dependent List:** box and enter *Sex* in the **Factor List:** box.



After clicking on the **Statistics** tab, the partial outputs are:

Descriptives

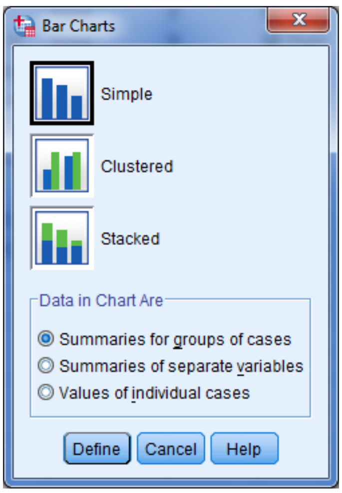
	Sex			Statistic	Std. Error
Weight	F	Mean		49.68	.326
		95% Confidence Interval for Mean	Lower Bound	49.04	
			Upper Bound	50.32	
		5% Trimmed Mean		49.24	
		Median		49.00	
		Variance		98.344	
		Std. Deviation		9.917	
		Minimum		16	
		Maximum		96	
		Range		80	
		Interquartile Range		12	
		Skewness		.741	.080
		Kurtosis		1.615	.160
	M	Mean		55.67	.413
		95% Confidence Interval for Mean	Lower Bound	54.86	
			Upper Bound	56.48	
		5% Trimmed Mean		55.49	
		Median		55.00	
		Variance		90.992	
		Std. Deviation		9.539	
		Minimum		29	
		Maximum		92	
		Range		63	
		Interquartile Range		11	
		Skewness		.347	.106
		Kurtosis		.828	.211

Number of CD4 Counts	F	Mean		209.71	5.842
		95% Confidence Interval for Mean	Lower Bound	198.25	
			Upper Bound	221.18	
		5% Trimmed Mean		191.43	
		Median		173.00	
		Variance		31739.389	
		Std. Deviation		178.156	
		Minimum		2	
		Maximum		1914	
		Range		1912	
		Interquartile Range		192	
		Skewness		2.929	.080
		Kurtosis		17.435	.160
	M	Mean		178.13	6.778
		95% Confidence Interval for Mean	Lower Bound	164.81	
			Upper Bound	191.44	
		5% Trimmed Mean		161.08	
		Median		140.00	
		Variance		24532.012	
		Std. Deviation		156.627	
		Minimum		1	
		Maximum		1352	
		Range		1351	
		Interquartile Range		172	
		Skewness		2.692	.106
		Kurtosis		12.675	.211

4.4 Diagrams and Graphs

4.4.1 Bar Charts

From the *Menu* bar, click on **Graphs** → **Legacy Dialogs** → **Bar**. Then, the **Bar Charts** dialogue box appears.

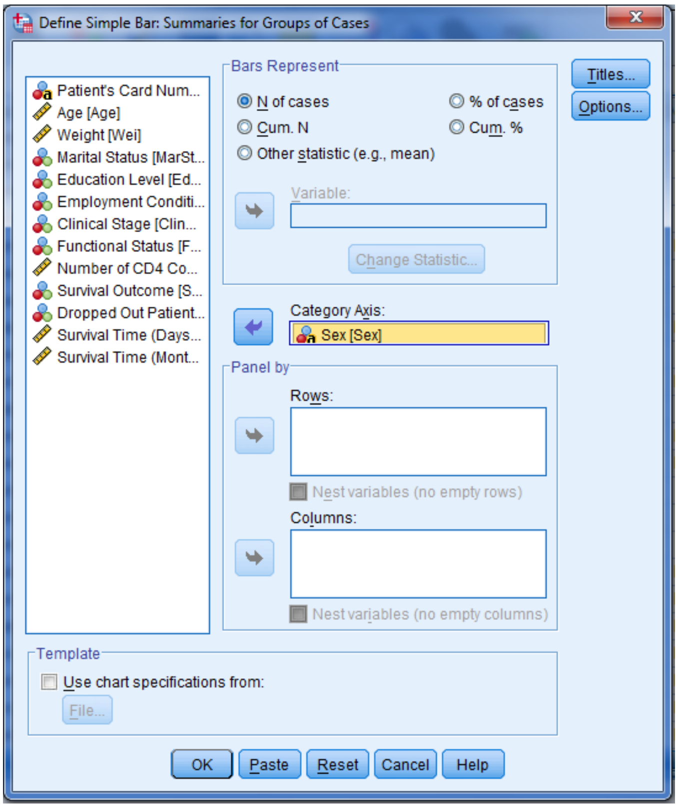


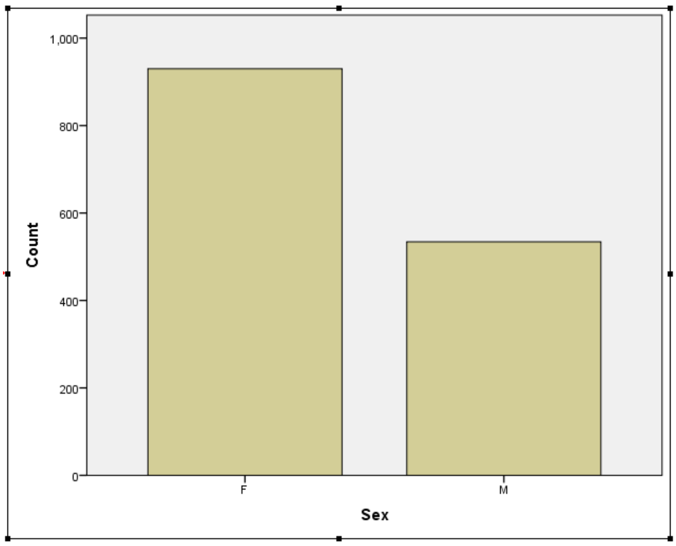
Simple Bar Diagram

A simple bar diagram is a diagram in which categories of a variable are marked on the X axis and the frequencies of the categories are marked on the Y axis.

Example 4.6. Construct simple bar diagram for *Sex*.

Of the three types of bar charts in the **Bar Charts** dialogue box, select the first one, that is, the **Simple** option. Then click on the **Define** button. In the **Category Axis:** box enter *Sex* and then **OK**.



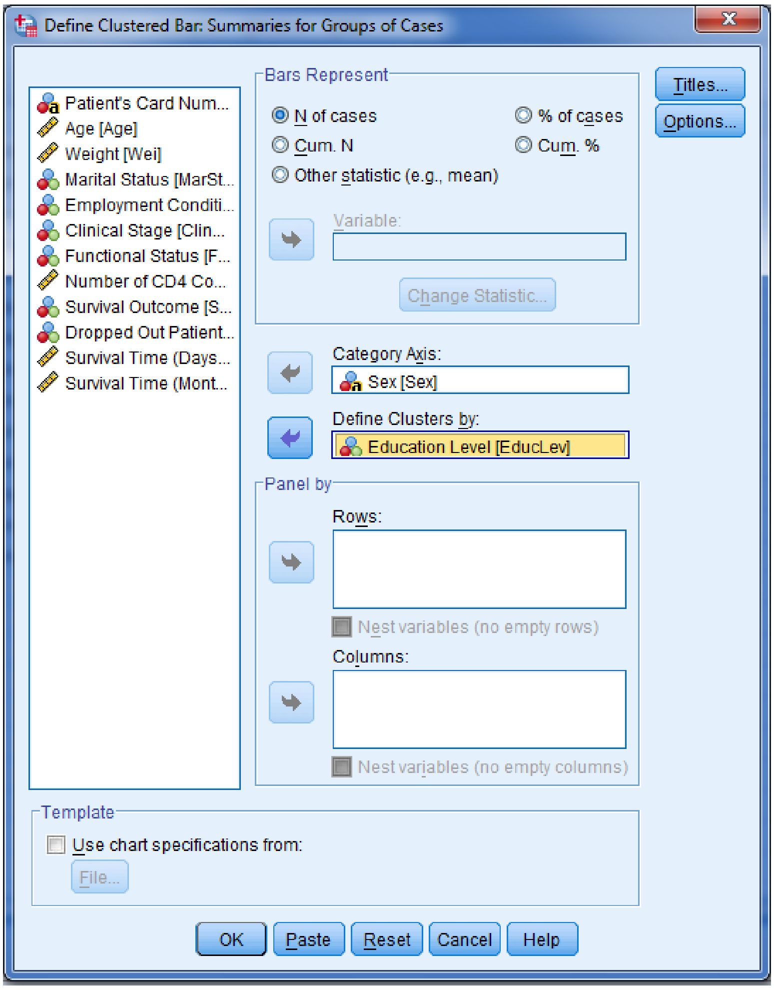


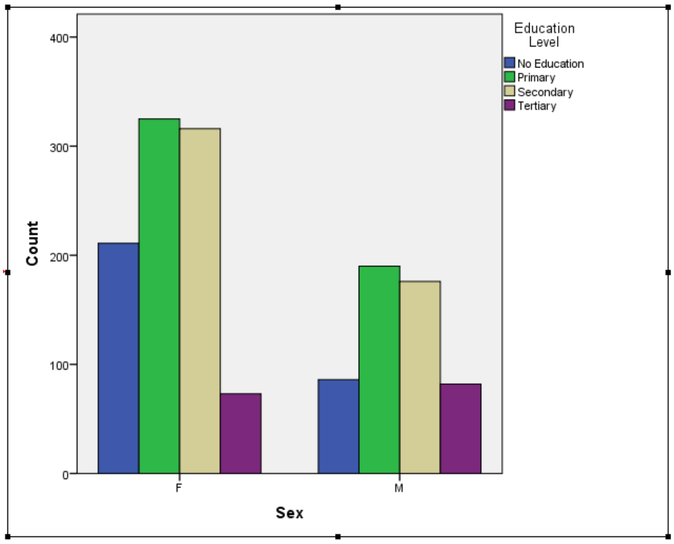
Multiple (Clustered) Bar Diagram

Multiple (clustered) bar diagram is used to display data on more than one variable. In the multiple bars diagram two or more sets of inter-related data are interpreted.

Example 4.7. Construct multiple bar diagram for *Sex* and *Education Level*.

Again, of the three types of bar charts in the **Bar Charts** dialogue box, select the second one, that is, the **Clustered** option. Then click on the **Define** button. In the **Category Axis:** box enter *Sex* and in the **Define Clusters by:** enter *Education Level*. Then **OK**.



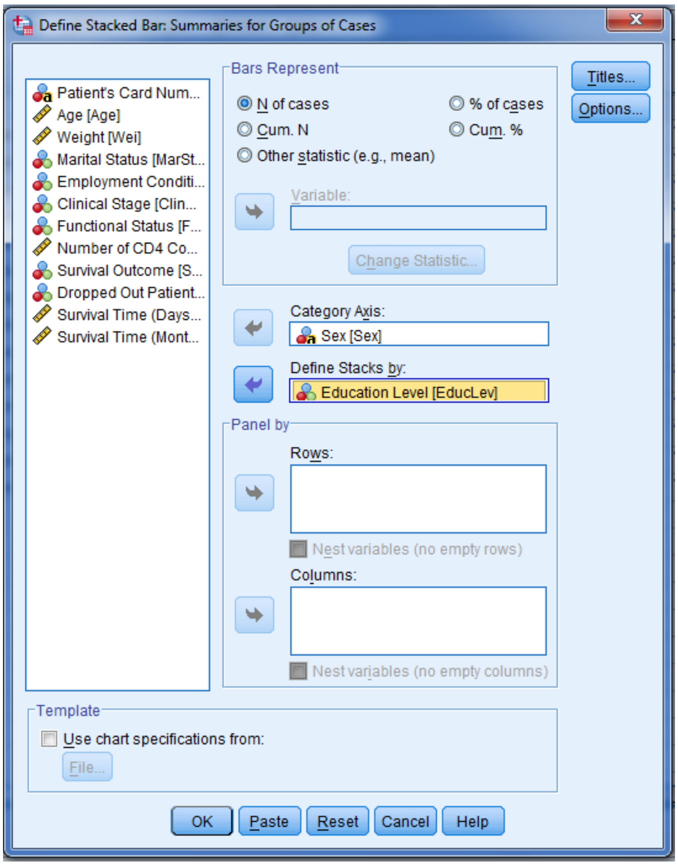


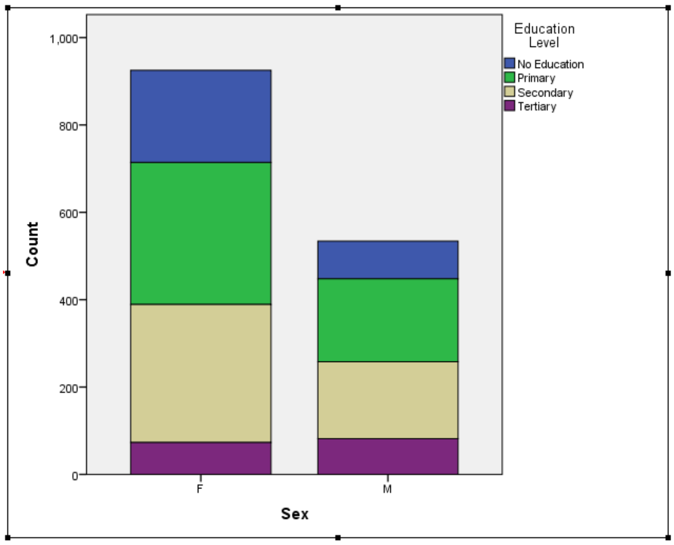
Component (Stacked) Bar Diagram

Component (stacked) bar diagram is used when there is a desire to show a total or aggregate is divided into its component parts.

Example 4.8. Construct multiple bar diagram for *Sex* and *Education Level*, and compare it with the clustered bar chart above.

Lastly, of the three types of bar charts in the **Bar Charts** dialogue box, select the third one, that is, the **Stacked** option. Then click on the **Define** button. In the **Category Axis:** box enter *Sex* and in the **Define Stacks by:** enter *Education Level*. Then click on **OK**.



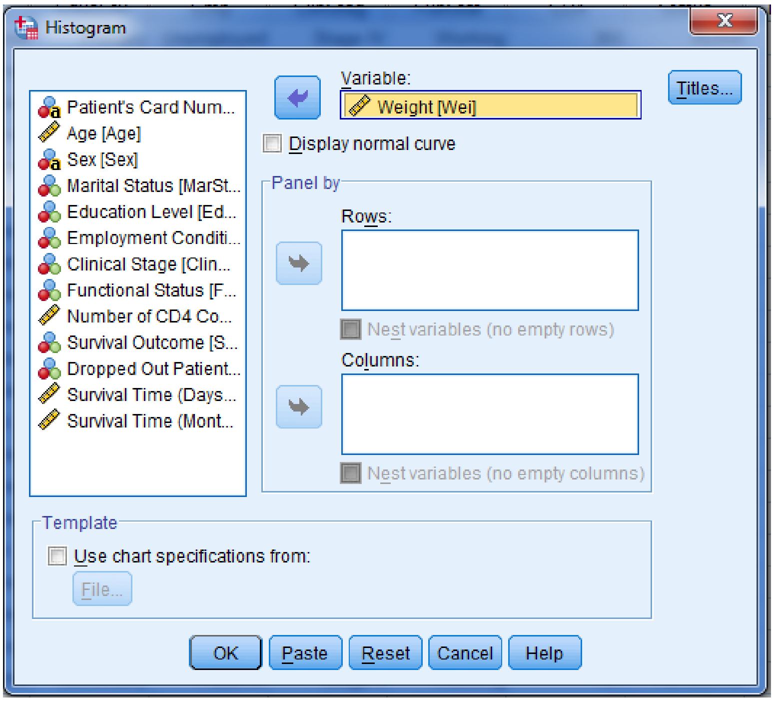


4.4.2 Histogram

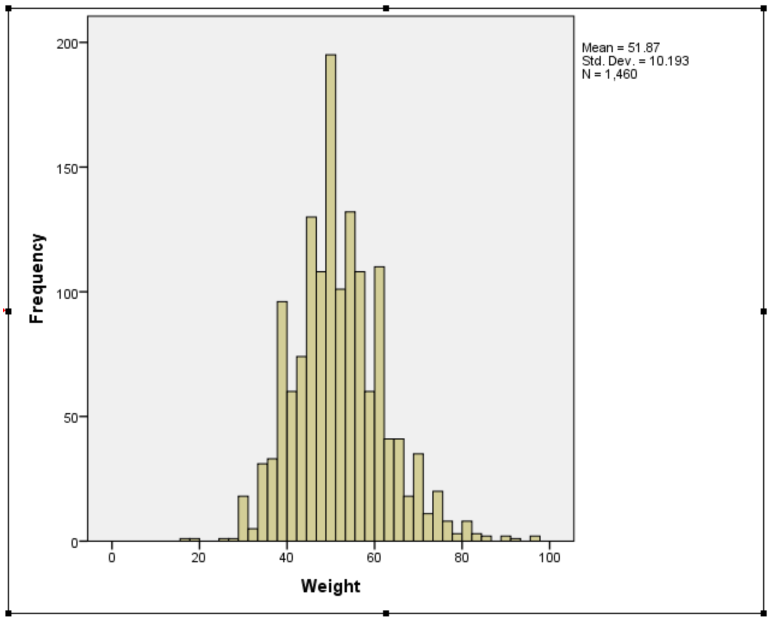
From the *Menu* bar, click on **Graphs** → **Legacy Dialogs** → **Histogram**.

Example 4.9. Construct histogram for *Wei* and say something about the distribution.

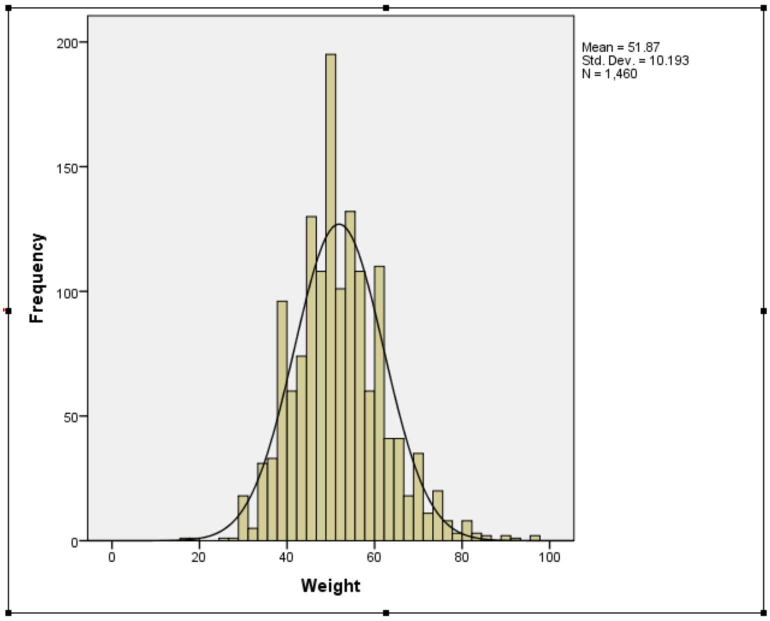
In the **Variable:** box of the **Histogram** dialogue box enter *Wei*. Then click on **OK**.



The histogram of weight of the patients is



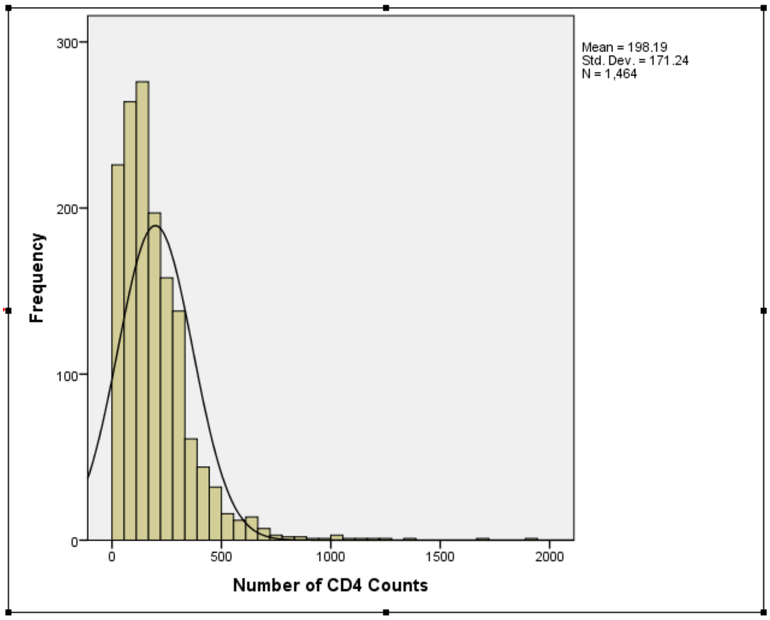
or by selecting **Display normal curve**



From this histogram, it seems the weight of the patients approximately follows a normal distribution.

Example 4.10. Construct histogram for *CD4*.

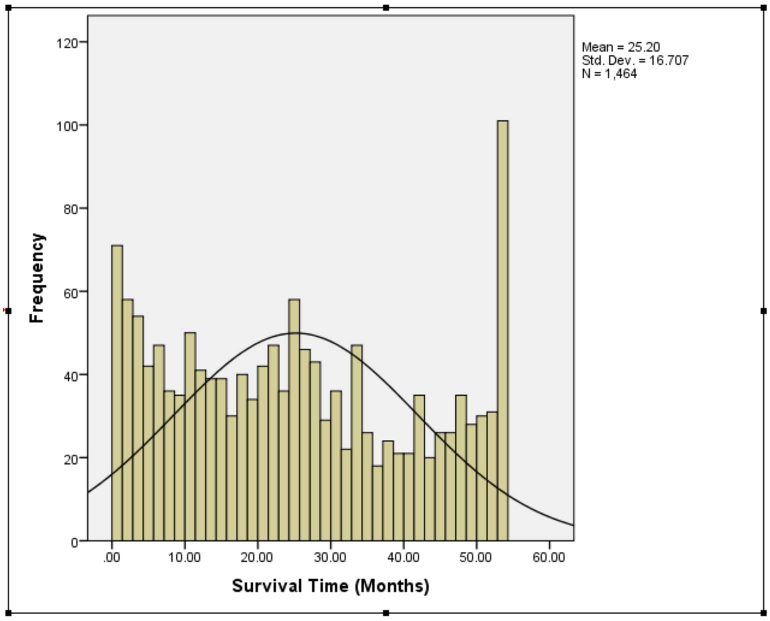
Using similar procedure as above, the histogram of the *CD4* is as follows.



This plot shows *CD4* is not normally distributed (positively skewed).

Example 4.11. Check the normality of *CD4* using histogram.

The histogram of *SurvTime* shown below indicates *SurvTime* does not follow a normal distribution.



Mean = 25.20
Std. Dev. = 16.707
N = 1,464
Frequency
Survival Time (Months)
0
20
40
60
80
100
120
.00
10.00
20.00
30.00
40.00
50.00
60.00

Chapter 5

Hypothesis Testing

5.1 Testing about a Single Population Mean

A one-sample t-test helps determine whether the population mean (μ) is equal to a hypothesized value (μ_0). If the difference between the sample mean and the test mean is large relative to the variability of the sample mean, then μ is unlikely to be equal to the assumed value.

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.

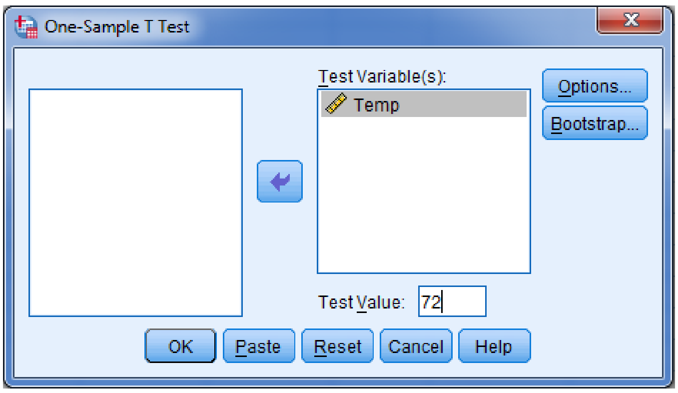
For testing a single population mean using SPSS, from the *Menu* bar, click on **Analyze** $\rightarrow$ **Compare Means** $\rightarrow$ **One-Sample T Test**.

Example 5.1. The thermostat in your classroom is set at 72°F, but you think the thermostat is not working well. On seven randomly selected days, you measure the temperature at your seat. Your measurements (in degrees Fahrenheit) are 71, 73, 69, 68, 69, 70, and 71. Test whether the mean temperature at your seat is different from 72°F. Conduct the analysis using SPSS.

Here, the hypothesis to be tested is H_0: $\mu = 72$ vs H_1: $\mu \neq 72$. Enter the data into SPSS under the variable name *Temp* as follows.

	Temp	var	var	var	var
1	71				
2	73				
3	69				
4	68				
5	69				
6	70				
7	71				
8					
9					
10					

In the **One-Sample T Test** dialogue box, enter *Temp* in the **Test Variable(s):** box. Then, in the **Test Value:** box, enter 72 which is the value assumed under H_0.



If you want to change the confidence level (default is 95%), click on the **Options** button, and then specify in the **Confidence Interval Percentage:** box. Then click on the **Continue** tab and **OK**.

The output provides two table results. The first table (**One-Sample Statistics**) provides the number of observations, mean, standard deviation and standard error of the sample mean. The sample mean and standard deviation of the 7 observations is 70.1429°F and 1.67616°F, respectively.

One-Sample Statistics

	N	Mean	Std. Deviation	Std. Error Mean
Temp	7	70.1429	1.67616	.63353

One-Sample Test

	Test Value = 72					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Temp	-2.931	6	.026	-1.85714	-3.4073	-.3070

The second table (**One-Sample Test**) provides the test statistic value, the degrees of freedom, the $p-$value, the difference of the sample mean from the assumed value and the confidence interval for the population mean. Thus, since the $p-$value is 0.026 which is smaller than the (default) level of significance $\alpha = 0.05$, the null hypothesis that the mean temperature at your seat is 72°F should be rejected. Therefore, the mean temperature in the classroom is significantly different from 72°F.

5.2 Comparing Paired Samples

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.

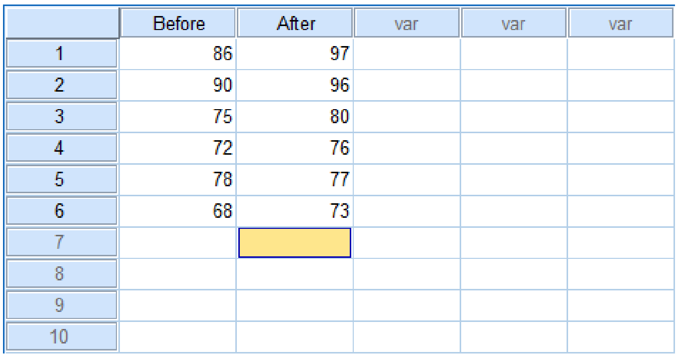
In SPSS, the procedure for paired test is: **Analyze** → **Compare Means** → **Paired-Samples T Test**.

Example 5.2. A researcher is interested in investigating whether alcohol has a positive or negative effect on 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

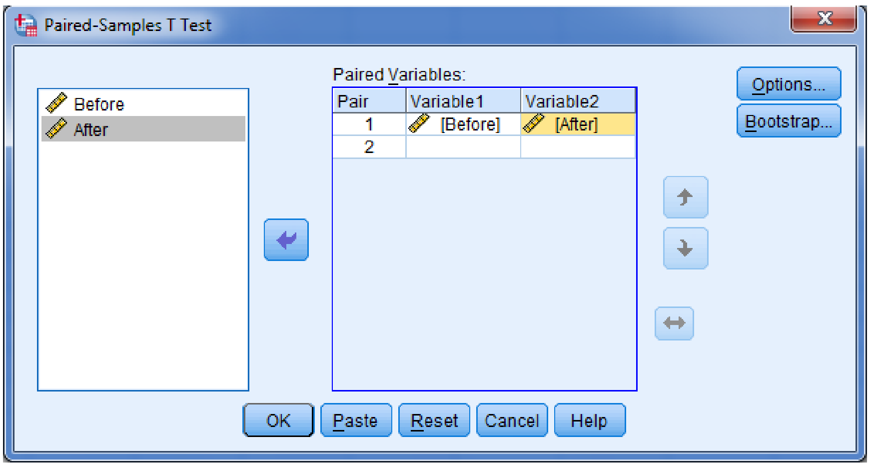
Test the hypothesis using SPSS.

Enter these data, naming the first variable of the pair *Before* and the second *After*, as follows.



	Before	After	var	var	var
1	86	97			
2	90	96			
3	75	80			
4	72	76			
5	78	77			
6	68	73			
7					
8					
9					
10					

Then, in the **Paired-Samples T Test** dialogue box, enter *Before* under **Variable1** and *After* under **Variable2** of the **Paired Variables:** box.



The **Paired-Samples T Test** procedure provides three table of results. The first table contains descriptive statistics for each of the two variables. The second table displays the correlation between the two variables as 0.943 and its corresponding $p-$value as 0.005. Since $p-$value is 0.005 which less than $\alpha = 0.05$, it can be concluded that there is a strong positive relationship between the before and after measurements of the heart beat of individuals.

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Before	78.1667	6	8.40040	3.42945
	After	83.1667	6	10.57198	4.31599

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	Before & After	6	.943	.005

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	Before - After	-5.00000	3.84708	1.57056	-9.03726	-.96274	-3.184	5	.024

As can be seen from the third table (**Paired Samples Test**) result, since $p-$value is 0.024 which smaller than $\alpha = 0.05$, we can conclude that alcohol has an increasing effect in the heart beat of individuals.

5.3 Comparing Independent Samples

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 have the same variance. If the two population variances are assumed to be different, then they must be estimated separately and the test statistic is a little bit modified as

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

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, first the equality of variances should be checked. That is,

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

$$H_0 : \sigma_1^2 = \sigma_2^2$$
$$H_1 : \sigma_1^2 \neq \sigma_2^2$$

2. Choose a level of significance (α).

3. The test statistic is: $F = \dfrac{s_1^2}{s_2^2}$ where $s_1^2 = \dfrac{1}{n_1 - 1}\sum_{i=1}^{n}(y_{1i} - \bar{y}_1)^2$ 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. 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.

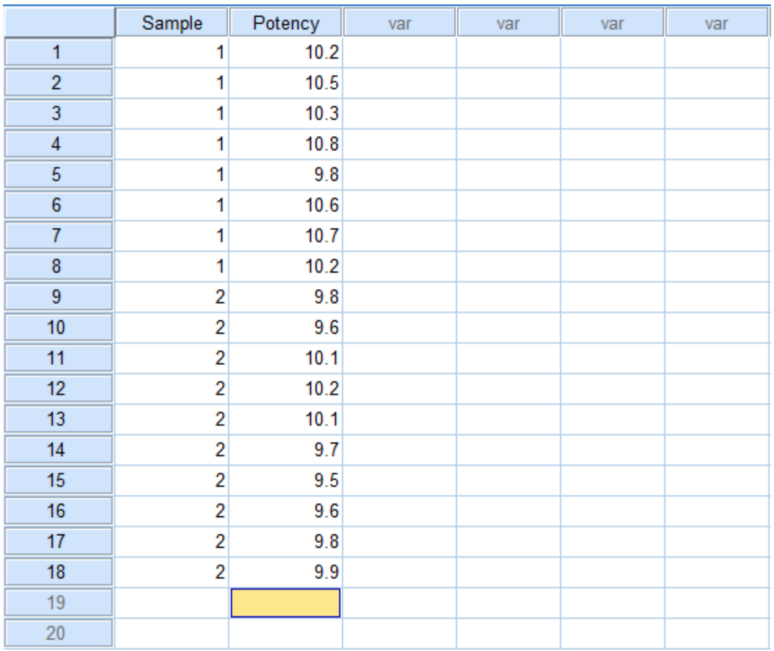
In the case of two independent populations, the procedure in SPSS is: **Analyze** $\rightarrow$ **Compare Means** $\rightarrow$ **Independent-Samples T Test**. Here, the response variable should be stacked in one column and a groping variable should be in another column.

Example 5.3. 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

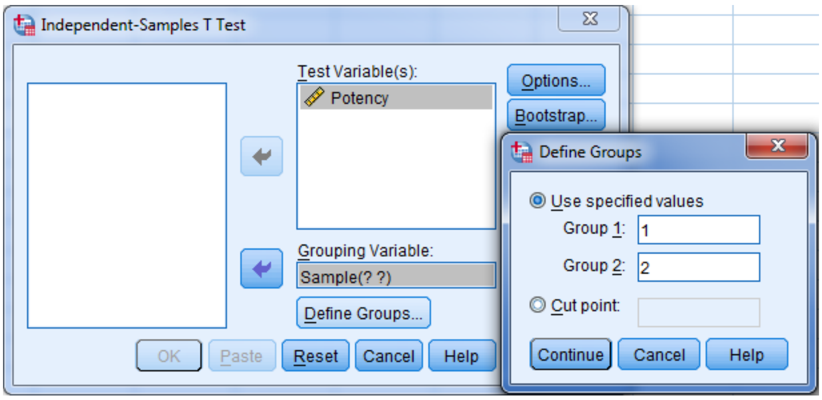
Using SPSS, test the null hypothesis that the drug product retains its potency. Also, construct the 95% confidence interval for the difference of the population means.

To enter the above data in SPSS, enter the grouping variable by naming **Sample** in one column and the stacked response in another column naming as **Potency**.



	Sample	Potency	var	var	var	var
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				
19						
20						

Then, in the **Independent-Samples T Test** dialogue box, enter *Potency* in the **Test Variable(s):** box and *Sample* in the **Grouping Variable:** box.



Next, click on the **Define Groups** button, and then type 1 in the **Group 1:** field and 2 in the **Group 2:** field. Click on the **Continue** tab and then **OK**.

This procedure results two tables; one a table of descriptive statistics for both variables and the other the table of test results.

Group Statistics

	Sample	N	Mean	Std. Deviation	Std. Error Mean
Potency	1	8	10.388	.3271	.1156
	2	10	9.830	.2406	.0761

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Potency	Equal variances assumed	.941	.347	4.172	16	.001	.5575	.1336	.2742	.8408
	Equal variances not assumed			4.028	12.545	.002	.5575	.1384	.2574	.8576

As shown in the above result in the **Independent Samples Test** table, SPSS by default conducts Levene's test for the equality of variances in the two groups. And then the test statistic is calculated under both assumptions (equal variance assumption and different variances).

Thus, the equal variance assumption is not rejected as its $p-$value (0.347) is larger than $\alpha = 0.05$. Then, the corresponding independent samples t test statistic value has a p-value of 0.001 implying the rejection of the null hypothesis of no difference in the mean potency of the two samples. Therefore, there is a significant difference in the mean potency of the two samples.

Example 5.4. 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

5.4 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 data set 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 and the alternative hypothesis is at least one of the means is significantly different from the other. That is, if there are g groups, then $H_0 : \mu_1 = \mu_2 = \cdots = \mu_g$ vs H_1 : not H_0.

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, an appropriate non-parametric alternative for one-way ANOVA which is called the Kruskal - Wallis test should be used.

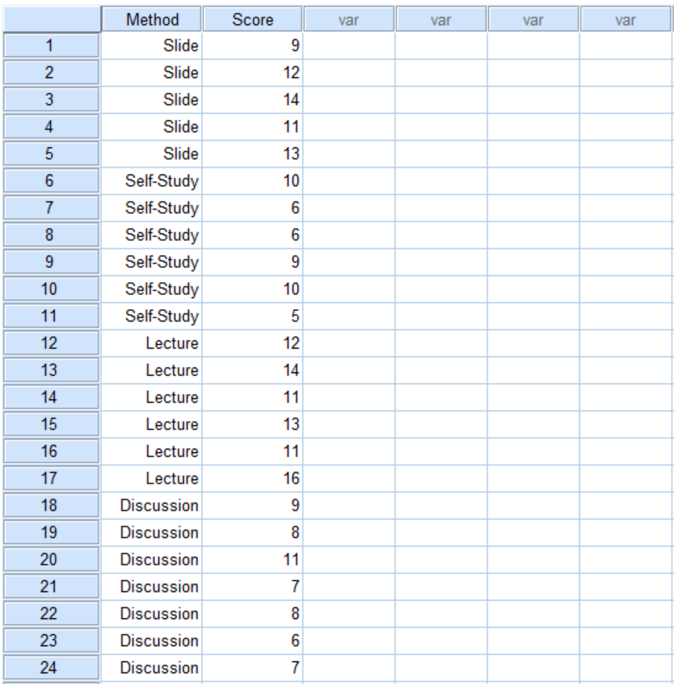
The procedure in SPSS for one-way ANOVA is: **Analyze → Compare Means → One-Way ANOVA**. Similar to the independent t test, the response should be stacked in one column and the grouping variable should be in another column.

Example 5.5. 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

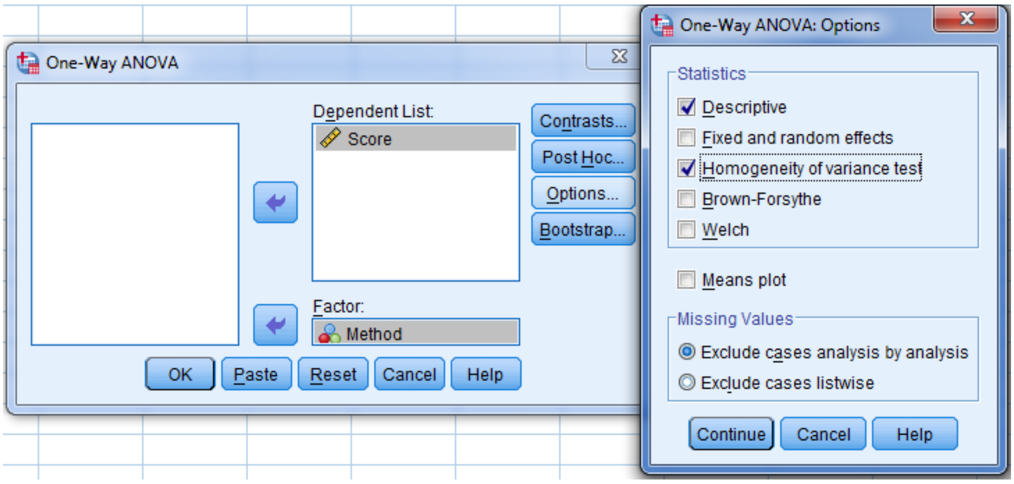
Is there any difference among the teaching methods?

Now enter the variable *Method* in one column and the *Score* in other column of the *Data Editor*.



	Method	Score	var	var	var	var
1	Slide	9				
2	Slide	12				
3	Slide	14				
4	Slide	11				
5	Slide	13				
6	Self-Study	10				
7	Self-Study	6				
8	Self-Study	6				
9	Self-Study	9				
10	Self-Study	10				
11	Self-Study	5				
12	Lecture	12				
13	Lecture	14				
14	Lecture	11				
15	Lecture	13				
16	Lecture	11				
17	Lecture	16				
18	Discussion	9				
19	Discussion	8				
20	Discussion	11				
21	Discussion	7				
22	Discussion	8				
23	Discussion	6				
24	Discussion	7				

Then, in the **One-Way ANOVA** dialogue box, enter *Score* in the **Dependent List:** box and enter *Method* in the **Factor:** box.



This procedure, by default, does not provide descriptive statistics per group. To obtain descriptives for each group, click on the **Options** button and select the **Descriptives** as above. Also, one of the assumptions of ANOVA is that the variances are the same across groups. To examine it, select the **Homogeneity of variance test**.

The larger the $p-$value (that is, a p-value of 0.461) for the Levene statistic indicates that the common variance assumption holds. Hence, the ANOVA test is appropriate.

Descriptives

Score

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
Slide	5	11.80	1.924	.860	9.41	14.19	9	14
Self-Study	6	7.67	2.251	.919	5.30	10.03	5	10
Lecture	6	12.83	1.941	.792	10.80	14.87	11	16
Discussion	7	8.00	1.633	.617	6.49	9.51	6	11
Total	24	9.92	2.948	.602	8.67	11.16	5	16

Test of Homogeneity of Variances

Score

Levene Statistic	df1	df2	Sig.
.894	3	20	.461

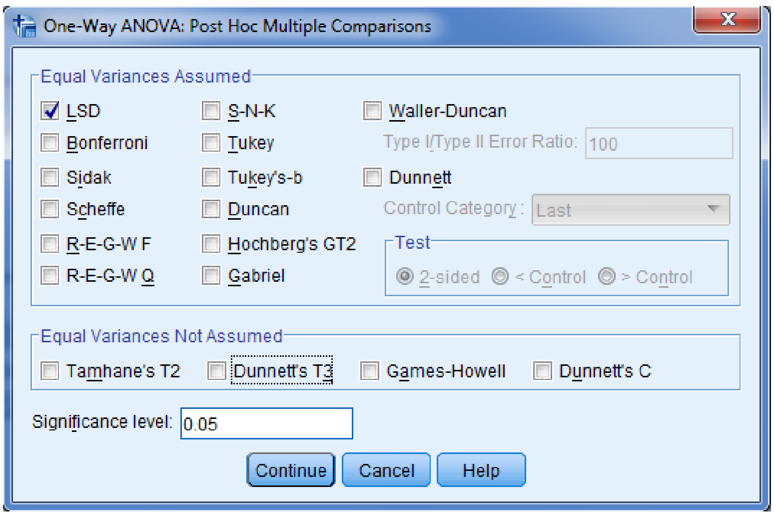
ANOVA

Score

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	124.867	3	41.622	11.104	.000
Within Groups	74.967	20	3.748		
Total	199.833	23			

In the ANOVA test, the significant F statistic (a p-value of < 0.0001) tells us that the means are not all equal, that means, at least one of the teaching methods differs from the other.

However, the one-way ANOVA does not tell us where the differences are. To examine the differences in the teaching methods, mean separation method should be used. To do so, click on the **Post Hoc** button of the **One-Way ANOVA** dialogue box and then select at least one comparison method like LSD, Bonferroni or Scheffe.



The output from the LSD mean separation method is as follows.

Multiple Comparisons

Dependent Variable: Score
LSD

(I) Method	(J) Method	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval Lower Bound	95% Confidence Interval Upper Bound
Slide	Self-Study	4.133*	1.172	.002	1.69	6.58
	Lecture	-1.033	1.172	.389	-3.48	1.41
	Discussion	3.800*	1.134	.003	1.44	6.16
Self-Study	Slide	-4.133*	1.172	.002	-6.58	-1.69
	Lecture	-5.167*	1.118	.000	-7.50	-2.84
	Discussion	-.333	1.077	.760	-2.58	1.91
Lecture	Slide	1.033	1.172	.389	-1.41	3.48
	Self-Study	5.167*	1.118	.000	2.84	7.50
	Discussion	4.833*	1.077	.000	2.59	7.08
Discussion	Slide	-3.800*	1.134	.003	-6.16	-1.44
	Self-Study	.333	1.077	.760	-1.91	2.58
	Lecture	-4.833*	1.077	.000	-7.08	-2.59

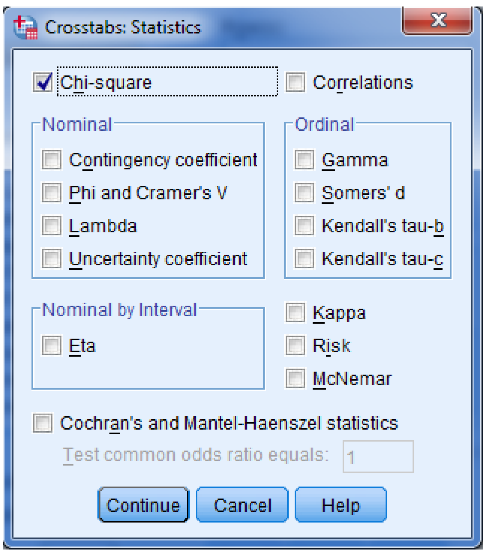
*. The mean difference is significant at the 0.05 level.

The significant pairs are Slide > Self-Study, Slide > Discussion, Self-Study < Lecture and Lecture > Discussion.

5.5 Chi-Square Test of Association

The χ^2 test is used for testing the independence of two categorical variables. The null hypothesis is H_0 : There is no statistical association between the two categorical variables.

The Pearson χ^2 test in SPSS is found as an option in **Statistics** tab of the **Crosstabs** dialogue box. As usual, if the $p-$value is less than the specified level of significance α, H_0 will be rejected.



Example 5.6. Is there a statistical association between the *Sex* of a patients and *Education Level* of the *JUSH_HAART.sav* data.

The result is as follows. The expected frequencies are obtained by selecting **Expected** under the **Counts** option of the **Cells** tab in the **Crosstabs** dialogue box.

Sex * Education Level Crosstabulation

			Education Level				Total
			No Education	Primary	Secondary	Tertiary	
Sex	F	Count	211	325	316	73	925
		Expected Count	188.3	326.5	311.9	98.3	925.0
	M	Count	86	190	176	82	534
		Expected Count	108.7	188.5	180.1	56.7	534.0
Total		Count	297	515	492	155	1459
		Expected Count	297.0	515.0	492.0	155.0	1459.0

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	25.397[a]	3	.000
Likelihood Ratio	24.929	3	.000
N of Valid Cases	1459		

a. 0 cells (0.0%) have expected count less than 5. The minimum expected count is 56.73.

The significance of the Pearson chi-square test (the smaller the p-value) reveals that there is an association between the sex of the patient and education level.

Chapter 6

Correlation and Linear Regression

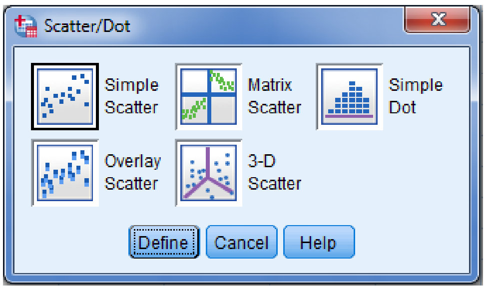
6.1 Correlation Analysis

Correlation is a statistical tool desired towards measuring the degree of the 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.

6.1.1 Scatter Plot

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.

From the *Menu* bar, click on **Graphs** $\rightarrow$ **Legacy Dialogs** $\rightarrow$ **Scatter/Dot**. Then, click on the **Simple Scatter** option of the **Scatter/Dot** dialogue box.



Then, click on the **Define** button.

Example 6.1. 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.

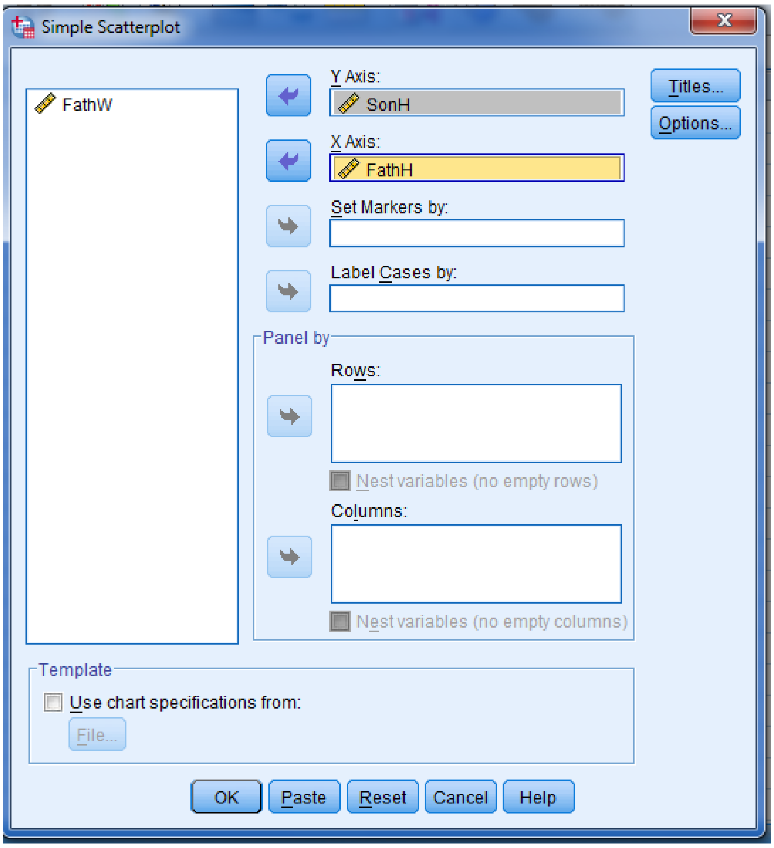
Son Height (Y)	66	68	65	67	69	70	71	60
Father Height (X_1)	65	67	66	67	68	69	69	62
Father Weight (X_2)	67	66	52	66	69	64	80	50

Obtain the scatter plot of son's height and father's height, son's height and father's weight, and father's height and father's weight.

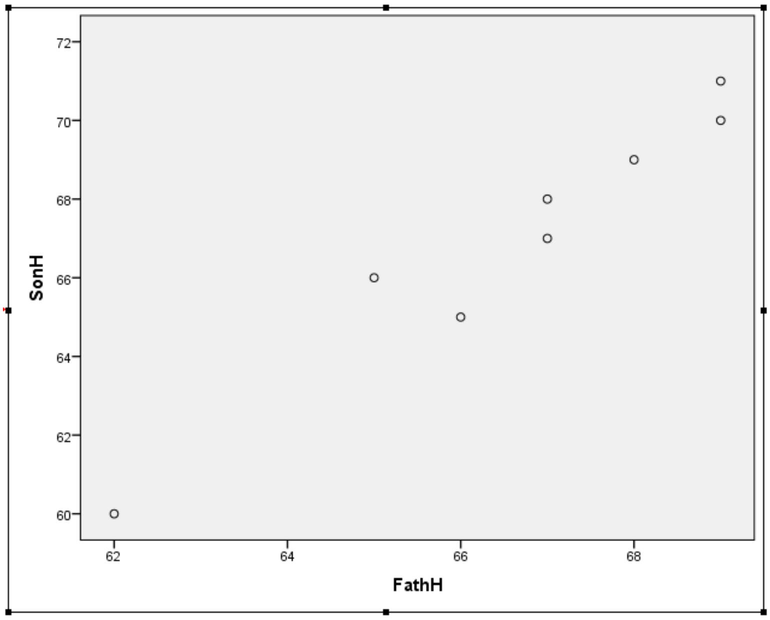
First enter the data as follows.

	SonH	FathH	FathW	var	var	var	var
1	66	65	67				
2	68	67	66				
3	65	66	52				
4	67	67	66				
5	69	68	69				
6	70	69	64				
7	71	69	80				
8	60	62	50				
9							
10							

In the **Simple Scatterplot** dialogue box, enter *SonH* in the **Y Axis:** box and enter *FathH* in the **X Axis:** box.

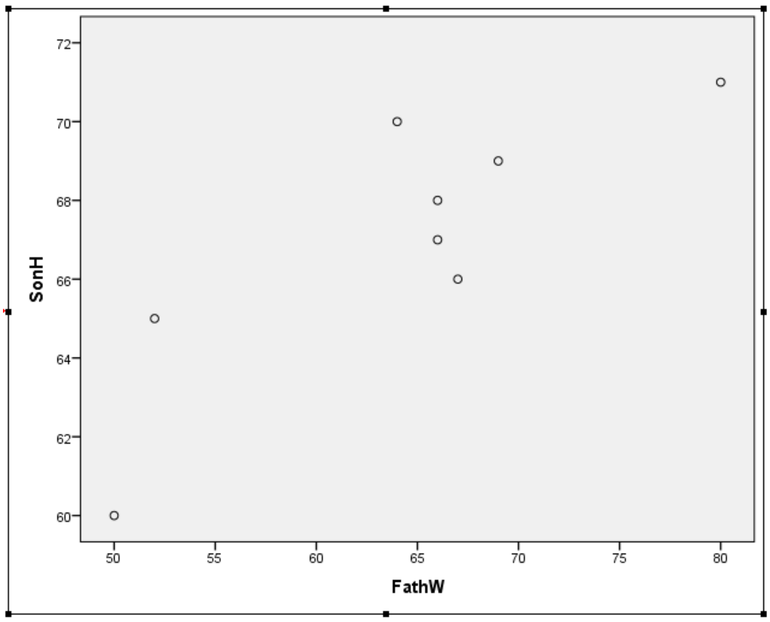


The scatter plot of son's height and father's height is:



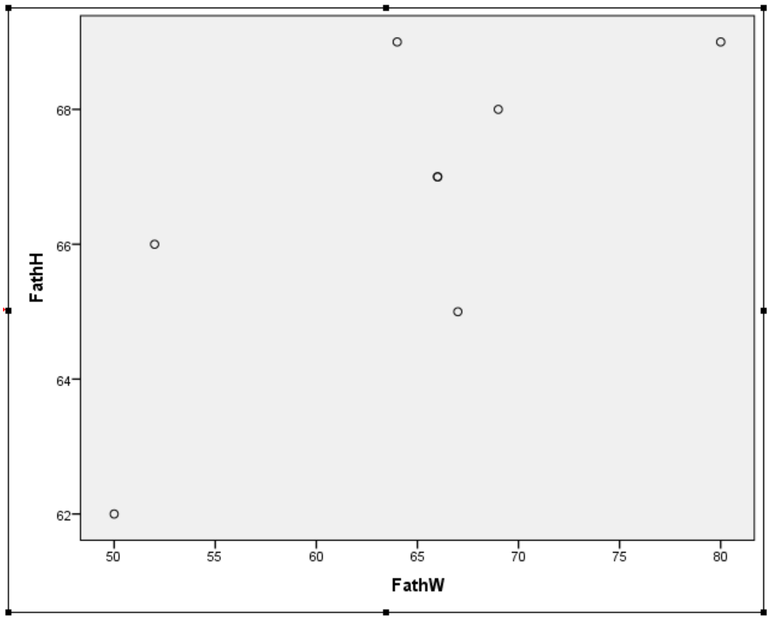
As can be seen from the plot, it is clear that there is a linear relationship between son's height and father's height.

Similarly, in the **Simple Scatterplot** dialogue box, by entering *SonH* in the **Y Axis:** box and *FathW* in the **X Axis:** box, the scatter plot of son's height and father's weight is shown below.



From this scatter plot, it seems there is a linear relationship between son's height and father's weight.

Again, in the **Simple Scatterplot** dialogue box, by entering *FathH* in the **Y Axis:** box and *FathW* in the **X Axis:** box, the scatter plot of father's height and father's weight is as follows.



There seams a linear relationship between father's height and father's weight.

6.1.2 Covariance and Correlation Coefficient

Covariance

It 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:

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

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.

Pearson's Correlation Coefficient

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.

$$r = \frac{S_{xy}}{S_x S_y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}\sqrt{n\sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y_i)^2}}$$

Interpretations of r: The value of the correlation coefficient can be positive or negative, depending on the sign of the covariance. But, it 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 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.

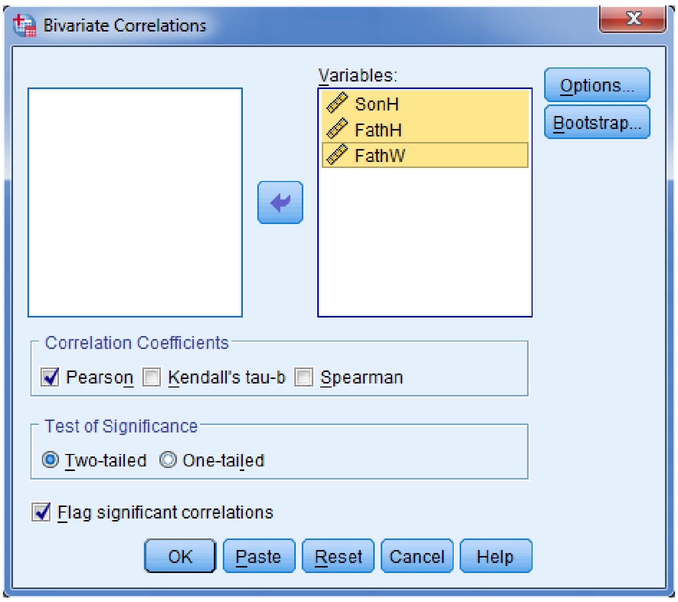
Limitations of r:

1. If x and y are statistically independent, the correlation coefficient between them is zero; but the converse is not always true. In other words, *zero correlation does not necessarily imply independence* because correlation has no meaning for describing nonlinear relations. Thus, for example, even if $y = x^2$ is an exact relationship, yet r is zero. (Why?)
2. Although, it is a measure of the linear association between variables, it does not necessarily imply any cause and effect relationship.

Using SPSS, to determine the correlation between quantitative variables, from the *Menu* bar, click **Analyze** $\rightarrow$ **Correlate** $\rightarrow$ **Bivariate**. Then, in the **Bivariate Correlations** dialogue box, enter at least two quantitative variables in the **Variable(s):** box.

Example 6.2. Using the data given on example 12.1, perform correlation analysis of among son's height, father's height and father's weight.

Enter the three quantitative variables in the **Variable(s):** box of the **Bivariate Correlations** dialogue box.



Then, click on **OK**.

The output of the correlation matrix is:

Correlations

		SonH	FathH	FathW
SonH	Pearson Correlation	1	.975**	.843**
	Sig. (2-tailed)		.000	.009
	N	8	8	8
FathH	Pearson Correlation	.975**	1	.732*
	Sig. (2-tailed)	.000		.039
	N	8	8	8
FathW	Pearson Correlation	.843**	.732*	1
	Sig. (2-tailed)	.009	.039	
	N	8	8	8

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

The result shows all the pairs of correlations are significant. Hence, it can be concluded there is a strong positive correlation between son's height and father's height, son's height and father's weight, and also father's height and father's weight.

6.2 Regression Analysis

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 variable which are used in determining the value of the dependent variable are called *explanatory (factor)* variables.

A *regression line* is a line 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 increment in the value of the dependent variable when the value of the j^{th} independent variable increases by 1 unit assuming all others the same.

Assumptions:

- Normal distribution: the response variable and the errors are normally distributed.
- Homoscedasticity: the variance of the response variable is constant for all values of the explanatory variable.
- Errors are independent and have a zero mean.
- No multicollinearity between the explanatory variables.

The objective in the above model is to estimate the regression parameters, β_0 and β_j; $j = 1, 2, \cdots, k$ using sample data. 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.

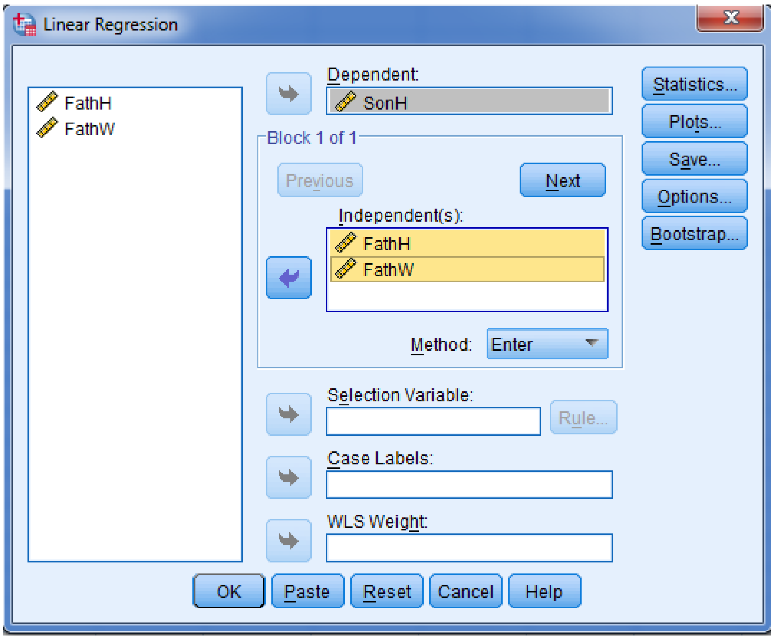
$\hat{\beta}_0$ is the estimated intercept.

$\hat{\beta}_j$ is the estimated (partial) slope of the j^{th} explanatory variable.

The procedure in SPSS to do regression analysis is: **Analyze** → **Regression** → **Linear**. Then, in the **Linear Regression** dialogue box, enter the dependent (response) variable in the **Dependent:** box and enter all the independent (explanatory) variables in the **Independent(s):** box.

Example 6.3. Recall example 12.1. Perform regression analysis of son's height on father's height and father's weight.

Enter the *SonH* in the **Dependent:** box, and enter both *FathH* and *FathW* in the **Independent(s):** box of the **Linear Regression** dialogue box.



Click **OK**.

The **Linear Regression** procedure of SPSS, delivers the main results in three tables as shown below. The **Model Summary** provides the coefficient of determination and the adjusted coefficient of determination as 0.987 and 0.981. This means, 98.7% of the variation in the height of sons is explained by both the father's height and father's weight.

The next table is the **ANOVA** which is used to determine the overall significance of the model. Since the $p-$value is very small, it is clear that the model is significant (that means, at least one of the explanatory variable is significant).

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.993[a]	.987	.981	.475

a. Predictors: (Constant), FathW, FathH

ANOVA[a]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	82.873	2	41.437	183.855	.000[b]
	Residual	1.127	5	.225		
	Total	84.000	7			

a. Dependent Variable: SonH

b. Predictors: (Constant), FathW, FathH

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	-16.059	6.364		-2.523	.053
	FathH	1.149	.113	.772	10.152	.000
	FathW	.101	.028	.278	3.652	.015

a. Dependent Variable: SonH

Lastly, the **Coefficients** table, contains parameter estimates of of the model together with their standard errors. The estimated model is $\widehat{SonH}_i = -16.059 + 1.149 FathH_i + 0.101 FathW_i$; $i = 1, 2, \cdots, 8$. Looking at the $p-$value of each parameter estimate, *FathH* is significant but not *FathW* (father's height does not determine son's height). Therefore, son's height is positively associated with father's height (taller fathers have taller sons). In particular, a one inch increment in the height of fathers leads to a 1.149 inches in height of sons.

Part II

Basics of the Stata Software Package

Chapter 7

Introduction to Stata

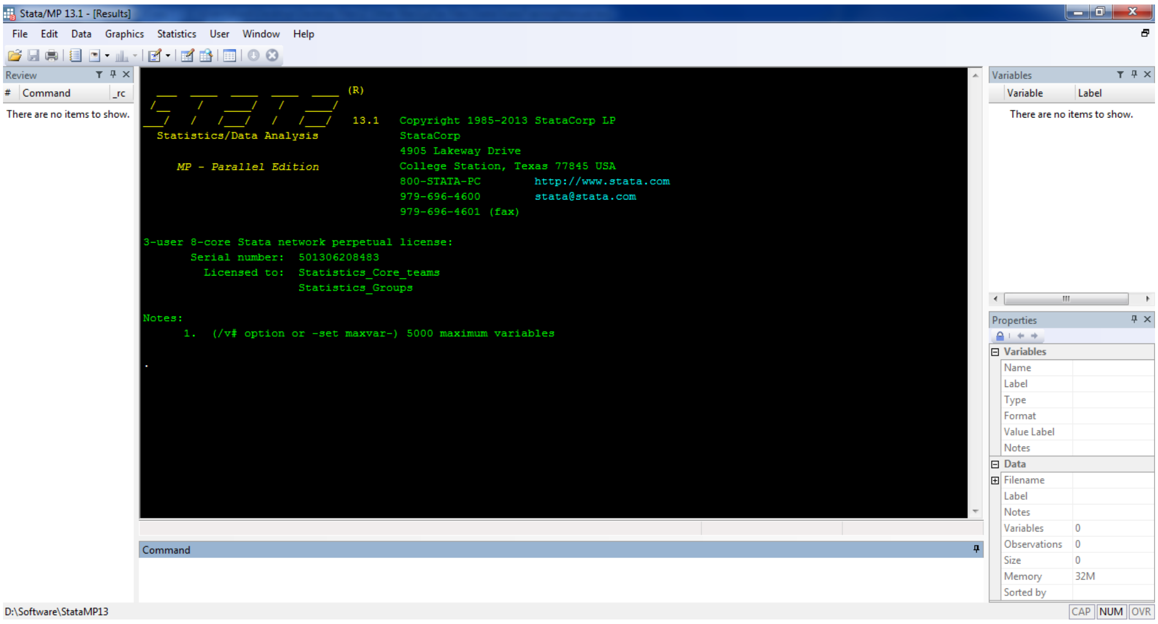
Stata is an integrated statistical analysis package designed for research professionals. It has got started in California in the mid-1980s and it was written in the C programming language. At one time, the name S was considered, later the name was changed to "Stata" (Statistics Data). In some early documentation, it was shouted out, all capitals, as "STATA", but the presently used form emerged quickly. Stata's main strengths are handling and manipulating large data sets.

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 versions is the maximum number of variables and observations that can be handled. The one that we will use is the 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.

7.1 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 below:



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

2. Variable *Properties* Window: This window displays the properties of each variable in the data set. 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. When pressing the *Enter* key, Stata immediately executes the command. If you click on any variable in the variables window, its name will copied in the *Command* window.

4. *Results* Window: This is the window where executed commands together with their outputs displayed. The commands are shown in the *Results* window preceded by a period (.). But, the period is not typed in the *Command* window.

5. *Review* Window: This window displays a list of all commands you have used in the order you used them. If you click on any command in this window, it will be immediately copied to the *Command* window. This is especially helpful, when mistyping a command, to edit it and run again. If you double-click on a command in this window, it is immediately executed.

A few points of emphasis:

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

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

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

7.2 Stata Language Syntax

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

prefix: `command` *varlist* ,*options*

For the most part, Stata will provide error messages when you type something wrong. The first thing to check is your 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 in the style of SPSS. When you use the menus (point-and-click), the command is generated for you, and appears in the *Review* and *Results* windows, just as if you 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

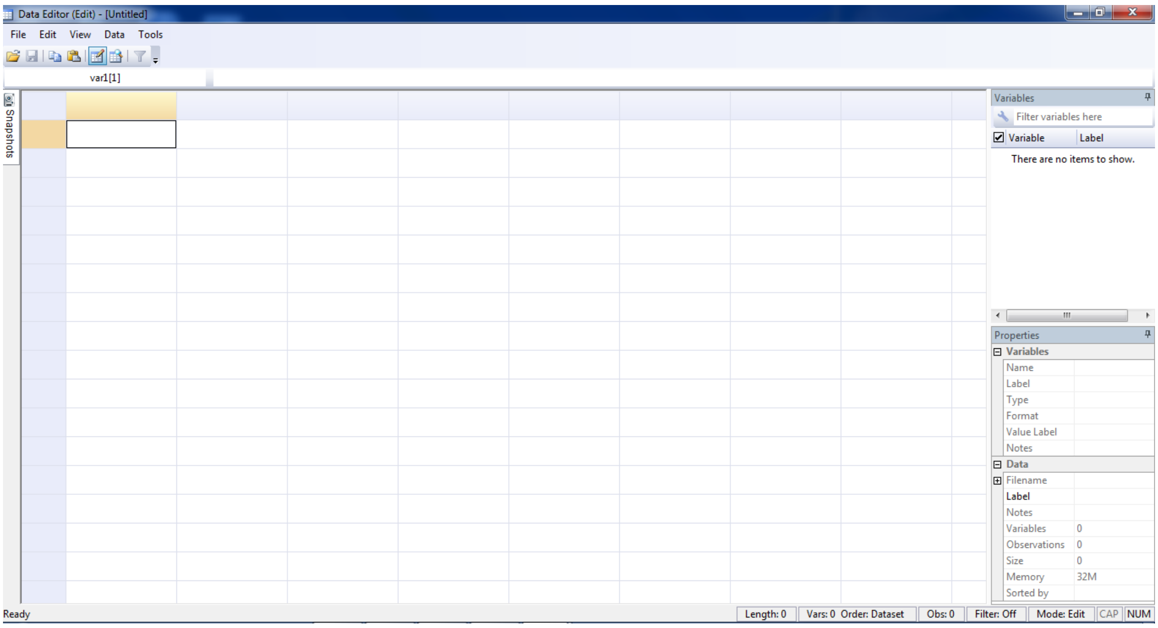
In this document, the following conventions are used: a `typewriter` font is used to represent the actual Stata commands and output that you see in the screen. Also, a *slanted* font is used for items such as variable names, filenames which you should replace with particular instances.

Chapter 8

Getting Data into Stata

8.1 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**, or click the Data Editor icon (fifth from the right) on the Tool Bar of the main Stata interface. If you have a data set in memory, it is displayed in the editor. Otherwise, you get a blank editor screen, like as follows:



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.

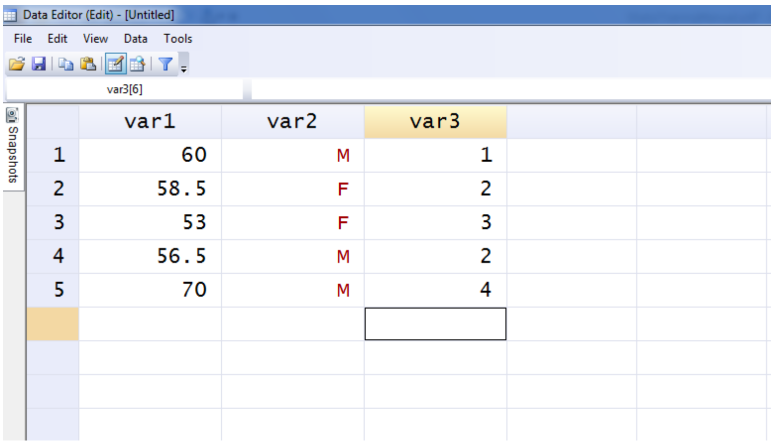
In the data editor, the default color for strings variables is red. The default color for numeric data is black as are dates. Numerically coded variables with labels, such as "male" and "female", but the underlying value is really a number, are blue.

8.1.1 Data Entry and Coding

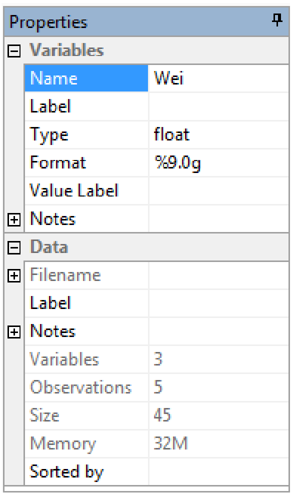
For illustration, let's enter the following data containing three variables (Weight, Sex and Marital Status (1=Single, 2=Married, 3=Divorced, 4=Other)) with five observations.

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 follow:



Stata assigned the default variable names `var1`, `var2`, `var3`. To change the variable names, click on the column heading `var1`. Then, on the properties window **Name** field, change `var1` to your preferred variable name, say, *Wei* and then press the **Enter** key.



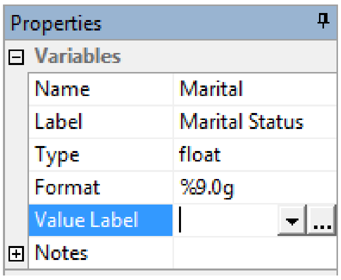
Similarly, change the variable names `var2`, `var3` to *Sex*, *Marital* respectively. The **Label** field of the properties window is also used to write a variable label. For example, by clicking on any cell of the variable we want to label, we can write the variable label in the **Label** field and press the **Enter** key.

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.

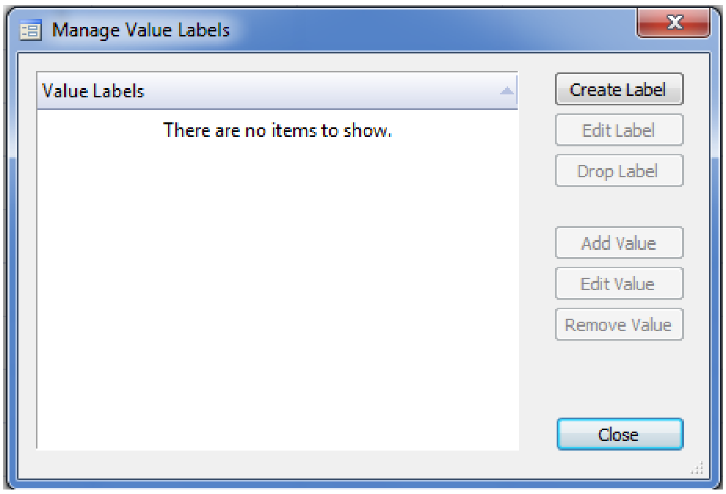
8.1.2 Creating Value Labels

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 the value label for *Marital Status*. First, a label that associates text with the codes should be created. In the variable *Properties* window, click on the value label field.

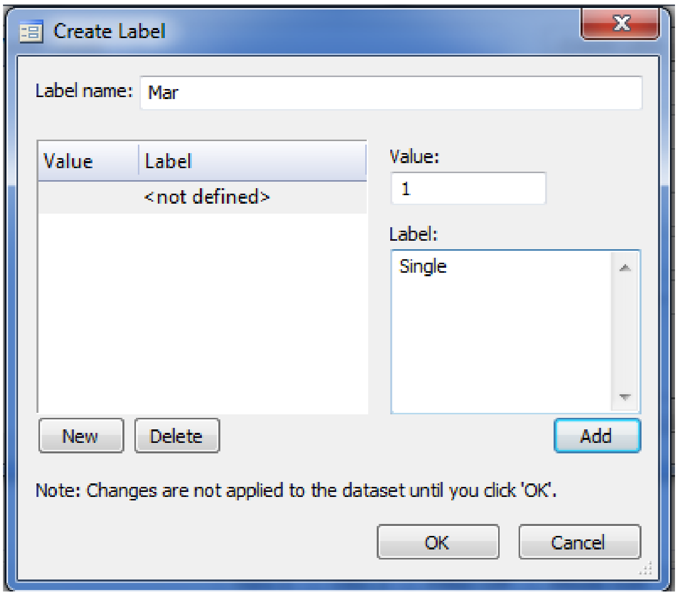


Then, click on the $\cdots$ tab which is next to the dropdown tab. This opens the **Manage Value Labels** dialogue box shown below.



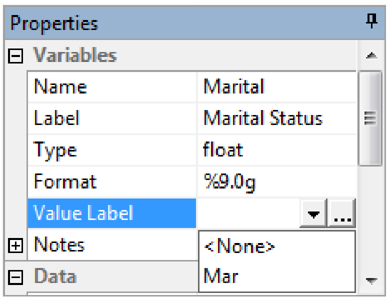
Now click on the **Create Label** button which opens **Create Label** dialogue box. Then, write the label name *Mar* (which can be the same to the variable name) in the **Label name:** field. Next,

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



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. Second, to associate the label with the variable, click on the variable *Marital* and then click on the dropdown tab in front of the **Value Label** field of the variable *Properties* window. Select *Mar* to attach these value labels to *Marital* variable.



Save the data file and exit Stata.

Exercise 8.1. Enter the following data 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

8.2 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.

You can only work on one Stata data set at a time. The data you are working on is stored in memory. In fact, there are two ways to load a data file, the quick menu-driven way and the longer manual-way. As you are just learning Stata, you should use the latter way.

8.2.1 Defining Working Directory: The `cd` Command

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

Create a new folder named *StataClass* on your 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, type

. `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.

8.2.2 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 to get ready to use a new data file. 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 data set will be shown in the variables window.
Notes:

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

Saving a Stata Data File: The `save` Command

One of the most important Stata commands to save your 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 your 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 you explicitly specify it, Stata will not write to any files with the same name as a file you already have. If we want to replace the existing data file, the option `replace` should be added in the `save` command. Thus, in order to force Stata to overwrite the file we already have:

. `save` *Student*, `replace`

As a result, you get a satisfying:

```
file Student saved
```

8.2.3 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 in an excel (`.xls`, `.xlsx`) data file format. 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 variable names cannot begin with a number. 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.

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.

8.2.4 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 8.2. In the folder given to you, there are two data sets in excel file named *JUSH_HAART.xlsx* and *CD4Count.xlsx*. Both data sets were obtained from Jimma University Specialized Hospital - HIV/AIDS Outpatient Clinic, South West of Ethiopia. The *JUSH_HAART.xlsx* 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.xlsx* 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.xlsx* 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.

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=Loss-to-follow
Defaulter	Dropped Out Patient	0=Active, 1=Defaulted
SurvTime	Survival Time (Months)	

Chapter 9

Basic Data Management

After creating a new data file or opening existing data file, it typically necessary to examine the data to identify possible problems. The question is "Does the data make sense?" out of range, missing, illogical/implausible values, consistency with other variables.

- How much are missing?
- Which variables have missing data?
- Any variable has value(s) which seem unusual/implausible? Example: Age of 150.
- Assess internal consistency. Example: 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.

9.1 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 you to move around in, but not alter, the relevant data. But, if for some reason you want to alter the data, the `edit` command can be used.

```
. edit
```

If you want to alter anything, click on the cell you are looking to alter, type in the new value (just as you would in Excel). If you wish to make the changes permanent for future Stata sessions, you 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
```

9.2 Describing Data in Memory: The `describe` Command

The `describe` command gives some very 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
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 describe some of the results.

1. `variable label` is longer name associated with each variable. For example, the variable *SurvTime* have a label *Survival Time in Months* and the variable *CD4* have 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 labels for *MarStat=0* could be *Never Married*, *MarStat=1* could be the *Married*, and so on.

3. `storage type` tells whether a variable is numeric or string (`str`). Stata stores or formats 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 that appears in the data as *jerrygarcia*, it is a `str11` variable. In the above case, *Sex* is a 1-character string. Everything else is numeric.

   Numeric storage can be a bit complicated. 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 formats or storage types are:

   - `byte`: Integers between -127 and 100.
   - `int`: Integers are normally stored in 4 bytes (that's, 32 bits, i.e. 32 binary 0s and 1s) of storage space. 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 numbers also have a default storage allocation of 4 bytes. It 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 similar to real numbers but are allocated twice as much storage space, 8 bytes, so that they can hold more digits in the mantissa. It is a real number with about 16 digits of accuracy. These follow the same basic rules as for real numbers.

   If we want to change the double storage type of *MarStat* to integer storage type, the command `recast` is to be used:

   `. recast int` *MarStat*

4. `display format` tells how the values of a numeric variable (data) 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, %10.0`g` 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 you may need to force Stata to display decimal places. You can do this with the `format` command which always begin with the percent sign (%) and end with a letter. For example, to force Stata to display the weight variable using 2 digits with 1 decimal place:

```
.  format %2.1f Wei
```

Now you can 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.

9.3 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 you can observe the difference by using the `describe` command.

9.4 Listing Values of the Variables: The `list` Command

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

```
.  list
```

The partial output is:

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

```
     |           Active          |          26.966667          |
     +-----------------------------------------------------------+

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

Clearly, there is a lot of data there. The above result contains only the first two observations.

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 |
     |-----------|
```

Controlling Output: The `set more` Command

Notice that not all variables are listed from the `codebook` 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
```

```
     +-----------------------------------------------------------+
  1. | CardNum | Age | Sex |  Wei |       MarStat |    EducLev   |
```

```
     |     1202 |   40 |    F |    43 |      Separated |     Primary  |
     |----------------------------------------------------------------|
     |          Emp |  ClinStag |     FunStat |  CD4 |      Status |
     |   Unemployed |  Stage IV |     Working |  365 |      Active |
     |----------------------------------------------------------------|
     |          Defaulter           |              SurvTime            |
     |             Active           |              26.966667           |
     +----------------------------------------------------------------+

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

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 you get Stata running a seemingly never ending command because of long output, you can click on the break icon, the red circle with the white cross symbol ( ), located in the toolbar under the Window menu, to stop the output.

9.5 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 it provides 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* data set.

```
.  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 below 25 years, 25% of the patients were below 28 years, 50% of the patients were below 32 years, 75% of the patients were below 39 years and 90% of them were below 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 below 40 kilograms, 25% of the patients weights below 45 kilograms, 50% of the patients weights below 51 kilograms, 75% of the patients weights below 57.75 kilograms and 90% of them weights below 65 kilograms. 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 variables refers to all variables and tells Stata to list every single variable in the data set. That's, to get the information for all the variables in the file, simply type `codebook` without specifying the variable(s).

```
. codebook
```

If you've altered a data set, dropped and/or altered variables or just want a quick look at the data, the `codebook` command is a good place to start.

9.6 Frequency Tables for Categorical Variables

Stata can produce one-way and two-way frequency tables which are useful for categorical variables.

9.6.1 One-way Tables: The `tabulate` and `tab1` Commands

Now, let's start with the simplest of analytical commands and create some tables. The Stata command `tabulate` creates a frequency table of categorical variables. 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 is specified, missing values are included in the frequency counts as shown below. Recall from above that missing observations are shown with a period (.).

```
.  tabulate Emp ,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
```

The `missing` option forced Stata to include the missing responses. 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
```

9.6.2 Two-way Tables: The `tabulate` and `tab2` Commands

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. 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. From our 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  Loss-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
```

Also, the `tab2` command produces all possible two-variable tables from the list of variables. In other words, the command

. `tab2` *FunStat Status*

produces the same table, that is,

```
-> tabulation of FunStat by Status

Functional |              Survival Outcome
    Status |    Active       Dead  Transferr  Loss-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
```

It appears that both *Working* and *Ambulatory* were more likely to be active. Say we'd then like to know more 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` or `tab2` commands .

```
. tabulate FunStat Status ,row

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

Functional |              Survival Outcome
    Status |    Active       Dead  Transferr  Loss-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
```

9.7 Descriptive Statistics for Quantitative Variables

9.7.1 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. 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, 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
```

9.7.2 Compact Table of Summary Statistics: The tabstat Command

The `tabstat` command displays summary statistics for a series of numeric variables in one table, possibly broken down on (conditioned by) another variable using the `by()` option. It is a useful alternative to summarize, the default gives mean, because it allows to specify the list of descriptive statistics with the option `statistics` (`mean`, `median`, `min`, `max`, `range`, `sd`, `variance`, `skewness`, `kurtosis`, percentiles: `p1`, `p2`, $\cdots$, `p99`). Like the `summarize` command, the `tabstat` results are valid only for quantitative variables.

. `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
--------------------------------------------------
```

As shown above, by default, 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) columns(stat)`

The output is:

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

9.8 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 the various commands described above. 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 we already know. Say we're looking for a continuous variable age of patients. The `count` command in Stata counts the number of cases (observations) in the data set. So, how might we structure a command that counts age if the value is under 40? Notice that we can do this as:

```
. count if Age <40
1111
```

Similarly, we can count those patients who were 40 years. Apparently 88 patients were 40 years as determined by the command below.

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

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

```
. count if Wei>95
6
```

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. **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 follows.

```
. count if Wei>95 & Wei < .
2

. list if Wei>95 & Wei < .

      +-------------------------------------------------------------------+
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 |
      +-------------------------------------------------------------------+
```

Or

```
. count if Wei>95 & Wei != .
2

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

9.9 Manipulating Data

9.9.1 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 command

. `gsort` +*varname*, `stable`

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

To sort observations in descending order with order of observations with equal values is randomized, the command is used as follows:

. `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*

9.9.2 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 data set based on the list after it.

. `order` *Var2 Var1 Var3*

Like the command `order`, `aorder` allows to change the order of the variables in the dataset, however, in the alphabetical order. By simply writing

. `aorder`

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

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

The `drop` and `keep` commands can be used either on variables or observations. The `drop` command deletes records or variables that are listed after the command while the `keep` command deletes everything except the specified observations or variables.

. `drop` *varlist*
. `keep` *varlist*

Whether to use the `keep` or `drop` command to get rid of the unnecessary variables depends upon the number of variables you want. If there are only a few variables that you do not want, then use `drop`. If, however, there are more variables that you do not want, then use `keep`. These commands can be combined with the `if` and `in` qualifiers. Be careful when dropping variables as you will not be able to get them back once saved.

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

. `drop` *Wei*==.

or

. `keep` *Wei*! = .

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

. `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.

9.9.4 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, the *JUSH_HAART.sav* has the identifier *CardNum*. 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 our data set. 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`

9.9.5 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* by *Weight*, then the command

. `rename` *Wei Weight*

changes the name *Wei* by the new name *Weight*.

9.9.6 Replacing Values: The `replace` Command

This command is used to change the contents of a variable when the variable already exists. 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 included in double quotes as follows.

. `replace` *varname= "newcharacters"* if *Sex == "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"*

9.9.7 Creating New Variables: The `generate` Command

Variable transformation is a way of creating new variables using existing continuous variables and formulae. To create a new variable, use the `generate` command. Spacing is not important; operators can have spaces before and/or after or none at all. Constants and variables can both be used to create new variables. Basic arithmetic expressions are formed using the operators: + for addition (numeric), - for subtraction, ∗ for multiplication, / for division and ∧ 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` *newvar* = *formula*

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

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

If you want to change an existing variable, you need to use the command `replace` instead of the `generate` command. That's,

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

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

9.9.8 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 data set, *CardNum* is patients card number which must be changed to string.

. `tostring` *CardNum* ,`generate`(*CardNum_str*)

```
CardNum_str generated as str4
```

9.9.9 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 second observation, the space in the second observation and the comma in the third 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. But, you must be extremely cautious when using this option because it can have unforeseen and undesired consequences. Use of the `generate` option to create a new variable is much safer than writing over the existing variable.

. `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, space and comma. 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. |  0,215        215       215 |
  5. |  0.299       .299      .299 |
     +----------------------------+
```

9.9.10 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*.

Here, the `encode` command can be used code labels of *Sex* and then generate a new variable, say, *Gender*.

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

If you 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 you 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 really the new variable, *Gender* in this example, is using the label, 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
```

9.9.11 Redefining a Categorical Variable: The `recode` Command

Now we'll do what we call 'recoding' a variable. The numeric values of *Gender* for 'F' and 'M' patients are 1 and 2, respectively. That's not what we want. We want the typical 0 = F, 1 = M setup. How might we do this? Yes, you guessed it, we'll be looking to recode the variable. The `recode` command is used to redefine the values of such a categorical variable according to the rules specified. Below are some examples:

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

Also, recoding can be done using the `recode` command with the `generate` option, which help us to create a new 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 1") (oldvalue2=newvalue2 "Label 2")* ,`generate`(*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.

```
. codebook 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
```

9.9.12 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. 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.

9.9.13 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 ≤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 (≤ 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.

9.9.14 Creating Dummy Variables: The `tabulate` Command

In section 9.6, 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` 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.

9.10 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.

9.10.1 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*, $\cdots$.

9.10.2 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
```

9.11 Combining Data Sets

It is often needed to merge several databases into one. Stata is very efficient in such kind of data handling. Two main ways of combining two or more data sets into one are to be considered. The first situation is when there are two data sets with *same variables but different observations (cases)* and the second situation is when there are two databases with the *same observations (cases) but different variables.*

Additional observations from one data file can be added to the end of another with the `append` command. Additional variables contained in one file can also be added to corresponding observations in another with the `merge` command. Both of these use a similar approach. A necessary condition is that both data sets should have an identifier of the observation, which might consist of a single variable (e.g. CardNum), or a series of variables. In both data sets, the variables must be coded the same way and should have the same format. Both data sets should be sorted by the identifiers first. The data file in memory (the one that's currently opened) is referred to as the 'master' (working) file. The file that's to be joined with the 'master' is known as the 'using' data file. Both files must be Stata files.

9.11.1 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 two files have different observations and are linked by having the same variables. But, be sure that the order of the variables should be the same in both files.

. `append using` *usingfile*

In the folder given to you, there are two data files, *DataForAppend_1.dta* and *DataForAppend_2.dta* having some same variables but different cases. Now let's add the cases from *DataForAppend_2.dta* to *DataForAppend_1.dta*. After opening *DataForAppend_1.dta*, type

. `append using` *DataForAppend_2.dta*

9.11.2 Adding Variables: The `merge` Command

The `merge` command adds additional variables. That's, the command combines two data files with different variables into one file. Both files should have a matching variable (or variables) that's 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 (`merge 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 (`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 (`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` *key* `using` *usingfile*

Options available with the `merge` command are `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.

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 master and using files. Normally, this is the desired value.

Consider an example. In the *JUSH_HAART.dta*, each patient has a unique card number. But, in the *CD4Count.dta*, the card numbers are repeated because the patients were followed for five years in which the CD4 was measured approximately every six months. Hence, each patient may have made many (1-10) visits to the clinic for CD4 counts, so there may be multiple observations for each card number. First, we have to sort both data sets by the patient's card number (*CardNum*). Then, after opening *JUSH_HAART.dta*, the `merge 1:m` command is used to join variables from the *CD4Count.dta* file by *CardNum*. That is,

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

If the merge worked as planned, Stata will give you 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 merged file can be saved as a new data file.

. `save` *MergedData*

Before trying 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. Therefore, do not forget to rename or drop the `_merge` variable if you want to merge other data.

Chapter 10

Saving Command and Output Files

10.1 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: cut, copy, paste, undo, open, save and print. By opening such a file, one can later reproduce, edit or add to the work without having to re-type those commands. The advantages of using do files are to reproduce results exactly and quickly, continue analyses from the point where some one left off and to recall reasoning.

To open the do-file editor, click on the Do-file Editor icon (it looks like paper and pencil) (the sixth from last icon on the toolbar) or select **Window** → **Do-file Editor** → **New do-file Editor**.

Typing one command per line is the same as that would have done in the interactive mode (command window). If there is a need of breaking up a long command to more than one lines, three slashes (///) can be used as a continuation symbol (this is in the Do-editor only, not on the Stata command line). A single comment line can begin with a "∗" or a "//" on a new line. 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.

To execute all the commands, select **Tools** → **Execute (do)** or click on the "Do current do-file" icon in the Do-file Editor. If you want to execute only some of the commands in the Do-file editor, select the commands you 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; the Save and Open file menu selections in the Stata main window only save and open Stata data files.

10.2 The Log-File

As explained above, do-files, except for initial exploratory work, are the best way to document your work because the results can always be replicated, and they serve as documentation of your Stata session. A way to document your entire Stata session, including Stata output, whether you 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 as a Stata Markup and Control Language (SMCL) format `.smcl` (only readable in stata) or as a text file `.log`. The `.smcl` can be printed from Stata, in whole or part, retaining bolds, underlines, and italics as seen them in the Results window but cannot be edited. 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 11

Hypothesis Testing

11.1 Testing about a Single Population Mean: The `ttest` Command

A one-sample t-test helps determine whether the population mean (μ) is equal to a hypothesized value (μ_0). If the difference between the sample mean and the test mean is large relative to the variability of the sample mean, then μ is unlikely to be equal to the assumed value.

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 structure of the command 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 11.1. The thermostat in your classroom is set at 72°F, but you think the thermostat is not working well. On seven randomly selected days, you measure the temperature at your seat. Your measurements (in degrees Fahrenheit) are 71, 73, 69, 68, 69, 70, and 71. Let's test whether the mean temperature at your seat 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. 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 you want to test one-sided, you'll need to divide the $p-$value of the two-sided test by two - 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

One-sample t test
------------------------------------------------------------------------------
         |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
       x |       7    70.14286    .6335301    1.676163    68.59267    71.69305
------------------------------------------------------------------------------
    mean = mean(x)                                                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
```

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.

11.2 Comparing Two Population Means

11.2.1 Comparing 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 11.2. A researcher is interested in investigating whether alcohol has a positive or negative effect on 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, you may first compute the difference between the two variables, and then conduct one-sample t-test.

```
. generate di=pre_response-post_response
. ttest di=0
```

11.2.2 Comparing Independent Samples: The `ttest` Command

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 have the same variance. If the two population variances are assumed to be different, then they must be estimated separately and the test statistic is a little bit modified as

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

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, first the equality of variances should be checked. That is,

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 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 the 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 11.3. 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 groups (samples).

. `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 11.1. 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

11.3 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 data set 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 and the alternative hypothesis is at least one of the means is significantly different from the other. That is, if there are g groups, then $H_0 : \mu_1 = \mu_2 = \cdots = \mu_g$ vs H_1 : not H_0.

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 oneway anova is quite robust (relatively unperturbed by violations of its assumptions). But if the samples are different size, an appropriate non-parametric alternative for one-way ANOVA which is called the Kruskal-Wallis Test 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 you to perform multiple comparison tests. We'll use oneway now and the corresponding two-way ANOVA will show how to use the anova command.

11.3.1 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 11.4. 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 any difference among the teaching methods. After entering the teaching *program* 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 you'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
```

11.3.2 Twoway ANOVA: The `anova` Command

The Stata command to be used for twoway ANOVA is the `anova` command followed by the response variable then the factor variables.

. `anova` *response factors*

Example: A consumer research firm wants to compare three brands of tires (X, Y, and Z) in terms of the tyre life over different road surfaces. Random samples of four tires of each brand are selected for each of three surfaces (asphalt, concrete, gravel). The life time of the tyres (in months) under each surface is recorded as follows. Construct an ANOVA table and conduct F-tests for the presence of nonzero brand effects, road surface effects, and interaction effects.

Surface/Brand	X	Y	Z
Asphalt	36, 39, 39, 38	42, 40, 39, 42	32, 36, 35, 34
Concrete	38, 40, 41, 40	42, 45, 48, 47	37, 33, 33, 34
Gravel	34, 32, 34, 35	34, 34, 30, 31	36, 35, 35, 33

. anova *TyreLife Surface Brand*

```
                           Number of obs =      36     R-squared     =  0.5901
                           Root MSE      = 2.98308     Adj R-squared =  0.5372

                  Source |  Partial SS    df       MS           F     Prob > F
              -----------+----------------------------------------------------
                   Model |  397.111111     4  99.2777778      11.16     0.0000
                         |
                 Surface |  241.722222     2  120.861111      13.58     0.0001
                   Brand |  155.388889     2  77.6944444       8.73     0.0010
                         |
                Residual |  275.861111    31  8.89874552
              -----------+----------------------------------------------------
                   Total |  672.972222    35  19.2277778
```

11.4 The χ^2 Test of Association

The χ^2 test is used for testing the independence of two categorical variables. The null hypothesis is 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 provides a variety of tests and measures of association for two-way contingency tables.

Example 11.5. 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.

11.4.1 Relative Risk for 2×2 Contingency Tables: The cs Command

Relative risk (risk ratio) is used with cohort study data and sometimes with cross-sectional data. Risk is the proportion of subjects who become cases. The cs command calculates point estimates and confidence intervals for the risk difference, risk ratio, and (optionally) the odds ratio, along with attributable (prevented) fractions for the exposed and total population. The command structure is:

. `cs` *Outcome Exposure* `,level(95)`

The coding for both the *Outcome* and *Exposure* must use 0 and 1 in such a way that for the *Outcome* (1=Cases, 0=Noncases) and for *Exposure* (1=Exposed, 0=Unexposed).

Example 11.6. Let's consider *Gender_rec* (1=Male, 0=Female) as an exposure and *Defaulter* (1=Defaulted, 0=Active) as an outcome. Hence, here for this particular example, the case is being defaulted and the exposure is being male.

. `cs` *Defaulter Gender_rec*

```
                 | Sex                    |
                 |   Exposed   Unexposed  |      Total
-----------------+------------------------+------------
           Cases |       189         142  |        331
        Noncases |       741         392  |       1133
-----------------+------------------------+------------
           Total |       930         534  |       1464
                 |                        |
            Risk |  .2032258    .2659176  |   .2260929
                 |                        |
                 |      Point estimate    |    [95% Conf. Interval]
                 |------------------------+------------------------
 Risk difference |        -.0626918       |   -.1082232   -.0171604
      Risk ratio |         .7642435       |    .6320758    .9240477
 Prev. frac. ex. |         .2357565       |    .0759523    .3679242
 Prev. frac. pop |         .1497633       |
                 +-------------------------------------------------
                               chi2(1) =     7.62  Pr>chi2 = 0.0058
```

The relative risk is significantly different from 1 (less than 1). In particular, females are more likely to default than females.

Note that the command `csi` can be used to find the relative risk if the cell counts `a`, `b`, `c` and `d` are provided.

. `csi` 189 142 741 392

The results are completely the same to the one that we have obtained using the `cs` command.

11.4.2 Odds Ratio for 2×2 Contingency Tables: The `cc` Commands

The `cc` command, for case-control data, is used calculate point estimates and confidence intervals for the odds ratio along with attributable (prevented) fractions for the exposed and total population. The structure of the command is:

. `cc` *Outcome Exposure*

Example 11.7. Again, let's consider *Gender_rec* as an exposure and *Defaulter* as an outcome.

. `cc` *Defaulter Gender_rec*

```
                                                         Proportion
                 |   Exposed   Unexposed  |      Total     Exposed
-----------------+------------------------+------------------------
           Cases |       189         142  |        331       0.5710
        Controls |       741         392  |       1133       0.6540
-----------------+------------------------+------------------------
           Total |       930         534  |       1464       0.6352
                 |                        |
                 |      Point estimate    |    [95% Conf. Interval]
                 |------------------------+------------------------
      Odds ratio |         .7041113       |    .5445772    .9117642 (exact)
 Prev. frac. ex. |         .2958887       |    .0882358    .4554228 (exact)
 Prev. frac. pop |         .1935159       |
                 +-------------------------------------------------
                               chi2(1) =     7.62  Pr>chi2 = 0.0058
```

The above result clearly shows that the odds ratio is less than 1.

We can also use the command `cci` by providing the cell counts `a`, `b`, `c` and `d`.

11.5 Nonparametric Tests

So far we have stressed that in order to carry out hypothesis tests we need to make certain assumptions about the types of distributions from which we were sampling. For example, to do t tests we needed to assume that the populations involved were approximately normal. In the two sample t-test we needed to make the more specific assumption that the variances are equal. An important part of statistics deals with tests for which we do not need to make such specific assumptions. These tests are called nonparametric or distribution-free tests.

These tests would ordinarily be used if a parametric test were not appropriate. This might happen, for instance, if you were working with a non normal distribution or a distribution whose shape was not yet evident. It might also happen that you are working with some special type of data for which there was no appropriate parametric test.

Nonparametric tests can't use the estimations of population parameters. They use ranks instead of the original sample data.

11.5.1 The Wilcoxon Signed-Rank Test: The `signrank` Command

As with the one-sample t test for paired data, we begin by forming differences. Then the absolute values of the differences are assigned ranks; if there are ties in the differences, the average of the appropriate ranks is assigned. Next, we attach a + or a - signs back to each rank, depending on whether the corresponding difference is positive or negative. This is achieved by multiplying each rank by +1, -1, or 0 as the corresponding difference is positive, negative, or zero. The results are n signed ranks, one for each pair of observations; for example, if the difference is zero, its signed rank is zero. The basic idea is that if the mean difference is positive, there would be more and larger positive signed ranks; since if this were the case, most differences would be positive and larger in magnitude than the few negative differences, most of the ranks, especially the larger ones, would then be positively signed. In other words, we can base the test on the sum of the positive signed ranks.

Example 11.8. Let's consider the paired data on heart beat of alcohol, recall example 11.2.

. `signrank` *Before=After*

```
Wilcoxon signed-rank test

        sign |      obs   sum ranks    expected
-------------+---------------------------------
    positive |        1           1        10.5
    negative |        5          20        10.5
        zero |        0           0           0
-------------+---------------------------------
         all |        6          21          21

unadjusted variance       22.75
adjustment for ties       -0.13
adjustment for zeros       0.00
                     ----------
adjusted variance         22.63

Ho: Before = After
             z =  -1.997
    Prob > |z| =   0.0458
```

11.5.2 Wilcoxon Rank-Sum Test: The `ranksum` Command

The Wilcoxon rank-sum test is perhaps the most popular nonparametric procedure. It is a nonparametric counterpart of the two-sample t test; it is used to compare two samples that have been drawn from independent populations. But unlike the t-test, the Wilcoxon test does not assume that the underlying populations are normally distributed and is less affected by extreme observations. The Wilcoxon rank-sum test evaluates the null hypothesis that the medians of the two populations are identical (for a normally distributed population, the population median is also the population mean).

Example 11.9. Let's consider the comparing the drug's potency between the two samples that we did before in example 11.3.

. `ranksum` *Potency* `,by(`*Sample*`)`

```
Two-sample Wilcoxon rank-sum (Mann-Whitney) test

      Sample |      obs    rank sum    expected
-------------+---------------------------------
           1 |        8         110          76
           2 |       10          61          95
-------------+---------------------------------
    combined |       18         171         171

unadjusted variance      126.67
adjustment for ties       -1.31
                     ----------
adjusted variance        125.36

Ho: Potency(Sample==1) = Potency(Sample==2)
             z =   3.037
    Prob > |z| =   0.0024
```

11.5.3 The Kruskal-Wallis Test: The `kwallis` Command

When we can assume that our data is normally distributed and that the population standard deviations are equal, we can test for a difference among several populations by using the one-way anova F test. However, when our data is not normal, or we aren't sure if it is, we can use the nonparametric Kruskal-Wallis test to compare more than two populations as long as our data come from a continuous distribution.

In the one-way anova F test, we are testing to see if our population means are equal. Since our data might not necessarily be symmetric in the nonparametric setting, it is better to use the median as the measure of center, and so in the Kruskal-Wallis test we are testing to see if our population medians are equal.

The idea of the Kruskal-Wallis rank test is to rank all the responses from all groups together and then apply one-way anova to the ranks rather than to the original observations. So like the Wilcoxon rank sum statistic, the Kruskal-Wallis test statistic is based on the sums of the ranks for the groups we are comparing. The more different these sums are, the stronger is the evidence that responses are systematically larger in some groups than in others. As usual, we again assign average ranks to tied observations.

Example 11.10. Now also let's consider the example that we have considered for oneway anova, example 11.4.

. `kwallis` *Score* ,by(*Method*)

```
Kruskal-Wallis equality-of-populations rank test

  +-----------------------------+
  |     Method | Obs | Rank Sum |
```

```
|------------+-----+----------|
|      Slide |   5 |    87.00 |
| Self-Study |   6 |    42.00 |
|    Lecture |   6 |   116.50 |
| Discussion |   7 |    54.50 |
+------------------------------+

chi-squared =    14.883 with 3 d.f.
probability =     0.0019

chi-squared with ties =    15.040 with 3 d.f.
probability =     0.0018
```

Chapter 12

Correlation and Linear Regression

This section describes the use of Stata to do regression analysis. Stata is capable of many types of regression analysis and statistical tests. In this section, we touch on only a few of the more common commands and procedures.

12.1 Correlation Analysis: The `correlate` and `pwcorr` Commands

Correlation is a statistical tool desired towards measuring the degree of the 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.

Scatter Plot

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 12.1. 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 (y)	66	68	65	67	69	70	71	60
Father Height (x_1)	65	67	66	67	68	69	69	62
Father Weight (x_2)	67	66	52	66	69	64	80	50

Let's obtain the scatter plot of son's height and father's height, son's height and father's weight, and father's height and father's weight.

Covariance

It 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:

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

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.

Pearson's Correlation Coefficient

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.

$$r = \frac{S_{xy}}{S_x S_y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}\sqrt{n\sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y_i)^2}}$$

Interpretations of r: The value of the correlation coefficient can be positive or negative, depending on the sign of the covariance. But, it lies between the limits -1 and +1; that is $-1 \le r \le 1$.

- 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 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.

Limitations of r:

1. If x and y are statistically independent, the correlation coefficient between them is zero; but the converse is not always true. In other words, *zero correlation does not necessarily imply independence* because correlation has no meaning for describing nonlinear relations. Thus, for example, even if $y = x^2$ is an exact relationship, yet r is zero. (Why?)
2. Although, it is a measure of the linear association between variables, it does not necessarily imply any cause and effect relationship.

The commands `correlate` and `pwcorr` can provide useful information for regression model specification and identifying possible sources of multicollinearity in the explanatory variables. 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. The `pwcorr` command displays all the pairwise correlation coefficients between the specified variables or, if no variable is specified, for all the variables in the dataset.

The structure of the commands is:

. `correlate` *varlist*

and

. `pwcorr` *varlist*

Example 12.2. Using the data given on example 12.1, let's perform correlation analysis of among son's height, father's height and father's 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

. correlate SonH FathH FathW ,cov
(obs=8)

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

12.2 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 variable which are used in determining the value of the dependent variable are called *explanatory (factor)* variables.

A *regression line* is a line 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 increment in the value of the dependent variable when the value of the j^{th} independent variable increases by 1 unit assuming all others the same.

Assumptions:

- Normal distribution: the response variable and the errors are normally distributed.
- Homoscedasticity: the variance of the response variable is constant for all values of the explanatory variable.
- Errors are independent and have a zero mean.
- No multicollinearity between the explanatory variables.

The objective in the above model is to estimate the regression parameters, β_0 and β_j; $j = 1, 2, \cdots, k$ using sample data. 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 *coefficient of determination* (R^2) tells how well the estimated model fits the data. That's, R^2 measures *the proportion (percentage) of the variation in the dependent variable explained by the explanatory variables.* The draw back of R^2 is that it increases when the number of explanatory variables increases. As a result, a modified R^2 called adjusted R^2 is better which penalizes for an increment in the number of variables (degrees of freedom).

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

. `regress` *depvar indepvars*

For example, the command

. `regress` y x_1 x_2 x_3

fits the response variable y on the three explanatory variables x_1, x_2 and x_3. Sometimes, we may want to estimate a regression for a subset of the sample. For example:

. `regress` y x_1 x_2 x_3 `if` $x_3 == 1$
. `regress` y x_1 x_2 x_3 `if` $x_3 < 30$

Note that in the regression-case, it does not matter whether the variable, you restrict upon, is included as an explanatory variable or not - Stata will detect that it is collinear (since there will be no variation in it) and automatically exclude it from the explanatory variables.

Example 12.3. Recall example 12.1. Let's perform regression analysis of son's height on father's height and father's 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
------------------------------------------------------------------------------
```

12.3 Factor Variables and Time-Series Operators

Many times, we do not want to enter variables in regression just like they are, but in some specific form. Basically, we are talking about prefixes for the variables to tell Stata what to do with it.

12.3.1 Categorical Variables: The `i.` Operator

If some of the explanatory variables are categorical 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) variables, should be created to represent the categories of the explanatory variable as shown so far in section 9.9.14 using the `tabulate` command with the `generate` option.

Suppose, for example; that one of the explanatory variable is marital status which has been coded as "Single", "Married", "Other". In this case two design variables (d_1 and d_2) are necessary. One possible coding strategy is that when the subject is "single" (reference), the two design variables, d_1 and d_2 would be set equal to 0; when the subject is "Married", d_1 would be set equal to 1 while d_2 is still equal to 0; when the marital status of the subject is "other", we would use $d_1 = 0$ and $d_2 = 1$. The following table shows this example of design variables for marital status with three levels.

Marital	Design Variables	
Status	d_1	d_2
Single	0	0
Married	1	0
Other	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 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 multicategory explanatory variable in a regression analysis, creating the design variables using the `tabulate` command with the `generate` option and entering the design variables in the model might be one solution, but probably not the most efficient one. Instead, we can simply write a prefix `i.` to the variable to tell Stata that the variable is multicategory, and then Stata will perform the regression analysis on the indicator (dummy) variables. For example, if `x2` is a multicategory variable, the `regress` command is written as:

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

Example 12.4. Recall example 12.1. Suppose the researcher wants, in addition to determining the relationship between the heights of sons and the heights of their fathers, the effect of race of the father (0=White, 1=Black and 2=Hispanic) on the height of sons. The complete data including the race of the fathers is as shown below.

Son Height (Y)	66	68	65	67	69	70	71	60
Father Height (X_1)	65	67	66	67	68	69	69	62
Father Weight (X_2)	67	66	52	66	69	64	80	50
Race (X_3)	1	1	2	1	0	0	2	1

Let's fit the son's height on the father's height, father's weight and father's race and interpret the results.

```
. reg SonH FathH FathW i.Race

      Source |       SS       df       MS              Number of obs =       8
-------------+------------------------------           F(  4,     3) =   60.51
       Model |  82.9716545     4  20.7429136           Prob > F      =  0.0034
    Residual |  1.02834552     3   .34278184           R-squared     =  0.9878
-------------+------------------------------           Adj R-squared =  0.9714
       Total |          84     7          12           Root MSE      =  .58548

------------------------------------------------------------------------------
        SonH |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
```

```
       FathH |   1.149221   .2070905     5.55   0.012     .4901665    1.808275
       FathW |   .1014186   .0392575     2.58   0.082    -.0235163    .2263534
             |
        Race |
      Black  |   -.084003   .7478607    -0.11   0.918     -2.46403    2.296024
   Hispanic  |  -.3000698    .616107    -0.49   0.660    -2.260797    1.660658
             |
       _cons |  -15.96597   12.21944    -1.31   0.282    -54.85368    22.92174
------------------------------------------------------------------------------
```

12.3.2 Lagged Variables: The `l.` Operator

In time series or panel analysis, you might want to use lagged variables. This is easily done with the prefix `l.`, for instance

. `reg` y `l.`y x_1 x_2

will perform an OLS estimation (regress) of y on the lagged value of y and two control variables x_1 and x_2. If you need more than one lag you can different choices:

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

12.3.3 Difference: The `d.` Operator

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

```
. gen dy=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.

12.3.4 Interaction Terms: The # Operator

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, you can use the symbol #. 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).

12.4 Model Diagnostics

After estimating a model, the next task is to check the entire regression for:

- Normality of the residuals
- Heteroscedasticity
- Collinearity
- Omitted and unnecessary variables

12.4.1 Normality: The swilk, sfrancia and sktest Commands

The swilk and sfrancia Commands

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.

Test for normality:

. `swilk` *depvar*
. `sfrancia` *depvar*

For our previous example:

. `swilk` *SonH*

```
                   Shapiro-Wilk W test for normal data

    Variable |    Obs       W           V         z       Prob>z
-------------+--------------------------------------------------
        SonH |      8    0.92589       1.032     0.052    0.47944
```

This tests the null hypothesis which states that the data is normal. This indicates that the null hypothesis is not rejected indicating normality.

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` *depvar*

For our example:

. sktest *SonH*

```
                    Skewness/Kurtosis tests for Normality
                                                        ------- joint ------
    Variable |    Obs   Pr(Skewness)   Pr(Kurtosis)  adj chi2(2)    Prob>chi2
-------------+---------------------------------------------------------------
        SonH |      8      0.1234         0.2401         4.03         0.1331
```

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

12.5 Checking Homoscedasticity

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

One of the classical assumptions of OLS is constant variance of the error term. After the model is estimated, check the residuals to see if this assumption is violated. The `rvfplot` command in Stata plots the residuals against the fitted values.

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

There shouldn't be any pattern if the residuals are homoskedastic. If you see 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 *FathH*

12.5.2 The Heteroskedasticity Test: The `estat hettest` Command

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

```
. estat hettest

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

         chi2(1)      =     0.07
         Prob > chi2  =   0.7940
```

The large $p-$value leads the non-rejection of the null hypothesis of constant variance.

12.5.3 Information Matrix: The `estat imtest` Command

```
. estat imtest

Cameron & Trivedi's decomposition of IM-test

---------------------------------------------------
              Source |       chi2     df      p
---------------------+-----------------------------
```

```
  Heteroskedasticity |       5.97      6    0.4271
            Skewness |       4.95      4    0.2928
            Kurtosis |       0.49      1    0.4817
---------------------+-----------------------------
               Total |      11.41     11    0.4098
---------------------------------------------------
```

The large $p-$values for the three tests ensures normality of residuals.

12.6 Detection of multicollinearity: The `estat vif` Command

Detection of multicollinearity - VIF (variance inflation factor) score.

```
.  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
-------------+----------------------
       FathH |      4.74    0.211030
       FathW |      2.87    0.348893
        Race |
          1  |      3.26    0.306441
          2  |      1.66    0.602025
-------------+----------------------
    Mean VIF |      3.13
```

12.7 Omitted Variables Test: The `estat ovtest` Command

Omitted variables are variables that significantly influence the response variable and so should be in the model, but are excluded.

```
.  estat ovtest

Ramsey RESET test using powers of the fitted values of SonH
       Ho:  model has no omitted variables
                  F(2, 1) =      0.53
                 Prob > F =      0.6973
```

This test shows that the model has no variables omitted.