

SPSS 10 for Windows SAMPLE SESSION

TIME SERIES

**Short Course Training Materials
Designing Policy Relevant Research and
Data Processing and Analysis with SPSS 10 for Windows
2th Edition**

**Department of Agricultural Economics, Michigan State University
East Lansing, Michigan
November 2000**

Components of the Time Series Training Materials

Section 0 - Discussion of the concept of levels of data. File structure for SPSS for Windows (Data and Syntax Editors and Output Navigator). Please be read before you start the Section1.

Section 1 - Basic SPSS functions: SPSS files, Descriptives and Data Transformation

Section 2 - Restructuring Data Files: Table Lookup & Aggregation

Section 3 - Importing data from other sources into SPSS, examples given for dBase and spreadsheet, how to open other data forms and save to SPSS data format.

Sections 4 to 9 develop skills in time series analysis using SPSS for Windows

Section 4 - Data Cleaning and Verification

Section 5 - Basic Analysis

Section 6 - Seasonal Analysis

Section 7 - Trends

Section 8 - Real Prices, Price Graphing and Tables

Section 9 - Margin Analysis

Section 10 - Graphs, Tables, Publications and Presentations: How to Bring Them Into a Word Processor.

Annexes - Presentation of filters versus temporary selections, graphing and data in chart options, and manipulating the output in SPSS.

Acknowledgments

Funding for this tutorial and the research data that were used as examples were provided by the Food Security II Cooperative Agreement between the Department of Agriculture Economics at Michigan State University and the United States Agency for International Development, Global Bureau, Office of Agriculture and Food Security.

SPSS for Windows TIME SERIES Sample Session

Section 0

Levels of data, time series and file structure (Data, Syntax and Output windows)

This section introduces the basic concept of levels of data, the notion of cross section analysis, and consequently, the methods of data organization. A brief description of the program and file structure of SPSS for Windows version 10 is given. It is essential that you read through this section before starting this sample session.

Levels of Data

The data file can be structured in more than one way. An article written by Chris Wolf, MSU Department of Agricultural Economics, goes into detail about the different ways to organize data. The title of this paper is: “Computer Analysis of Survey Data - File Organization for Multi-Level Data.” You can download this article in English or French at <http://www.aec.msu.edu/agecon/fs2/survey/index.htm>. It is important that you understand the concepts of data organization.

Alternative Methods of Data Organization—When Not to Follow the Rules

It should be clear from the paper written by Chris Wolf on levels of data that for any given survey there could be a wide range of possible data organizations, ranging between two extremes.

At one extreme, each individual data value could be stored as a separate case. The result is a very large number of cases, and may require many key variables in order to identify each case. There would be only one non-key variable in each case, and it would contain the data value. At the other extreme, all of the data could be entered into one case. It would not require any key variables, but there would be a huge number of variables.

Obviously neither of these alternatives is practical. The choice of an appropriate data organization involves finding a compromise somewhere between these two extremes which works well for your data and the type of analysis that you will perform on these data.

The rules for determining data organization that have been given in Chris Wolf’s paper are based on the principles of relational databases. If you follow them, you will always create a flexible, and understandable data set that can be used for multiple purposes. However, there may be situations where you want to deviate somewhat from this data organization strategy. If you understand the rationale behind the relational database principles presented, you will be well prepared to make changes in the data organization for a particular project, in cases where it may aid the data management and analysis process.

Time Series Data

One situation where it might be advantageous to depart from the rules given above is working with data where a date/time component is involved. Let’s take, for example, a weekly survey of market prices over the course of several years. The survey might cover three types of prices (producer, wholesale, and retail) for 20 products in 40 markets.

If you look at these data primarily from the perspective of time-series analysis, it would be tempting to organize the data so that each case is an observation in time, i.e. one week's data. To do this, however, would require 2,400 variables per case (3 prices by 20 products by 40 markets). If your software application permitted these many variables, this organization would probably work for certain types of analysis, but for others it would be terribly cumbersome and error-prone.

In contrast, the strictly relational organization of this data would call for four key variables, identifying the week, market, product, and type of price. Each case would then have a single non-key variable, which would be the price itself. (Depending on the specific survey, there might be one or two additional variables, such as a variable to indicate the unit of measure corresponding to that price.) However, there may be drawbacks to this type of organization also, especially when the time aspect of the data is more important than the cross-sectional aspect of the data.

As a compromise, our researchers who have worked with these types of data suggest using key variables to identify the product and market, along with multiple variables (producer price, wholesale or assembly price, and retail or consumer price) to identify the prices. This structure will usually work well as a baseline data file that can be re-organized fairly easily to different levels, if necessary, for specific types of analysis.

Files Used In SPSS 10 for Windows

SPSS 10 for Windows provides three windows—A) the Syntax Editor, B) the Data Editor and C) the Viewer (including charts) to work with data. The contents of each can be saved into the appropriate SPSS 10 for Windows file type.

After you have started the SPSS 10 program, the default window that is opened is the “Data Editor” (B). In the upper left hand corner of the window, select **File**, then **Open** and you will see 5 choices for the different file types that are available to be used by SPSS:

Data	data files	(Extension *.sav)
Syntax	syntax or command files	(Extension *.sps)
Output	output files	(Extension *.spo)
Script	advanced programming files for use with Sax BASIC that are created automatically each time an Output is created	(Extension *.sbs)
Other	all files	(*.*)

Data files contain the data that will be used in the analysis process.

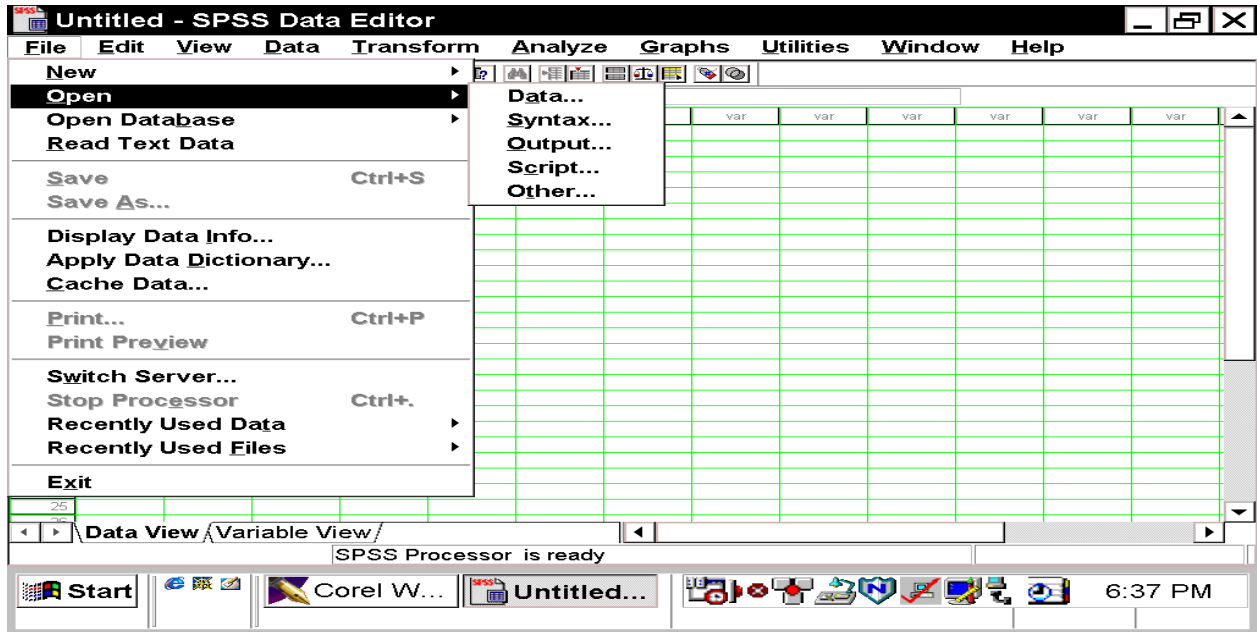
Syntax files contain the commands that are used to manipulate the data. Syntax files are not required, however we recommend that you keep the syntax commands you used during data analysis to be able to duplicate the output or rerun the analysis without having to recreate the commands.

Output files contain the results of the analysis. You may choose to keep the output for future reference or to be able to include specific results in your final report.

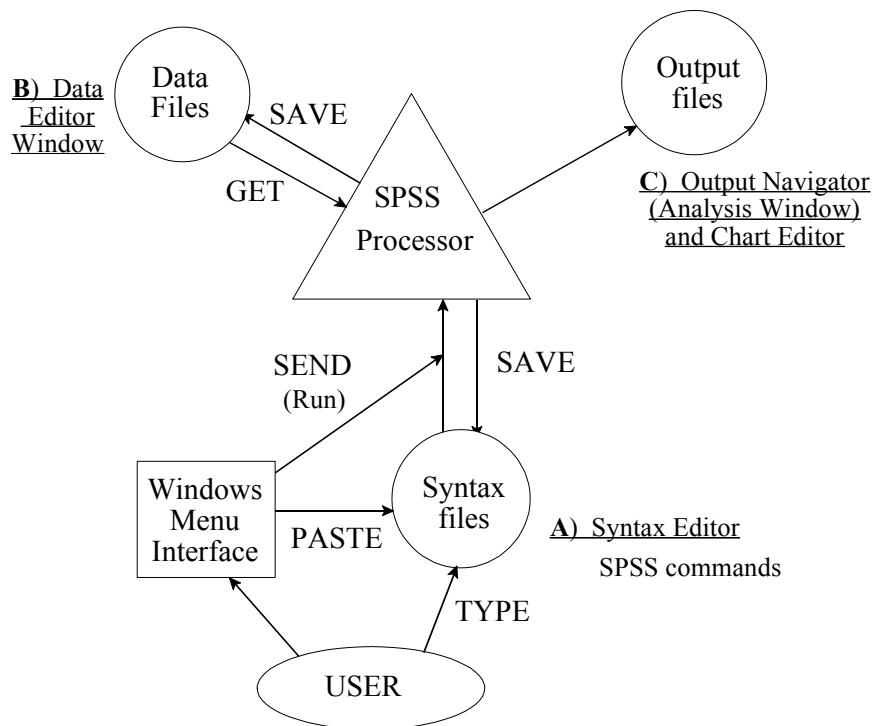
Script files are programming files that you can create to automatic many repetitive activities.

The Other file type shows all files that are available from within the directory no matter what the file extension may be.

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



The diagram below illustrates the relationship between the three main types of files we will be working with throughout this tutorial.



A) The Syntax Editor

The Syntax Editor is the window where syntax is pasted or commands are written before they are submitted to the SPSS processor. To put commands in the Syntax Editor you can **type** the commands directly into the Syntax Editor or you can use the pull down menus and select **Paste** when you are finished customizing the command. There are four main uses of the Syntax Editor:

- ! To type commands directly or to paste commands from the Data Editor to be processed later by SPSS 10 for Windows,
- ! To send these commands to SPSS 10 for Windows for processing,
- ! To write or save these commands to a file for future use, and
- ! To retrieve files of commands that you have saved previously.

It is important to understand that the commands you put in the Syntax Editor will not be executed (no output will be produced) until you send the commands to the processor. The Syntax Editor is simply an area that helps you prepare the commands. To send the commands to the processor, you use the **Run** button **O** in the Syntax Editor window toolbar of SPSS 10 (or select **Run ... Current** from the Menus). Once you press the **Run** button, the computer sends the command(s) to the processor. The processor reads the commands and executes them. If a command accesses the data (examples is Frequencies), the Viewer is automatically becomes the active window so you can examine the results of the command. You can then switch back to the Syntax Editor and add new commands or edit old ones and execute these changes to observe different results.

A good method to begin to learn the structure of the SPSS commands is to use the **Paste** option as you build the command from the menus, rather than to just click on the **OK** option. If you wish to have the commands written to the output file so that you can see which commands produced which output, do the following:

From the Menu (in any of the three windows), select

Edit

Options ...

- Select the **Viewer** tab
 - Under “Initial Output State”, click in the box next to **Display commands in log**
 - At the bottom of the dialog box, click on **OK**

As you successfully complete each step in your analysis (or when you are ready to end an SPSS 10 for Windows session, even if it was not completely successful), you should save the commands to a file so that you can use the commands another time. To save the commands, make the Syntax Editor the active window and select **Save** from the File menu. A file created from the Syntax Editor is called the *syntax (or command) file*. It is a file containing only commands; it never contains any of the data you may be analyzing with the commands. You must save your data separately, as described in the following section. We suggest that you use the default extension of .SPS when you give a name to a syntax file. Examples are: REP7.SPS, DEM-ALL.SPS, and SECT1.SPS.

By writing your commands to a syntax file, you can retrieve, look at, or modify sets of commands and rerun them. You can retrieve a syntax file by pulling down the **File** menu from any of the SPSS windows and selecting **Open**. Select **Syntax** and retrieve the filename under which you had last saved the file. Once you have opened a specific file, you can use the commands from the file, without having to recreate or type them again. If you make changes to the Syntax file that you wish to keep, make sure you save them to disk again.

B) The Data Editor Window

SPSS 10 for Windows stores your data in a data file. In addition to the values themselves, a data file contains such things as variable labels and value labels, formatting information, and missing-value specifications. Before you can perform any data analysis in SPSS 10 for Windows, you must first tell SPSS to open a Data file. First select **File** from the menu, select **Open, Data** and highlight a data file. You have two choices at this point:

- 1) click on **Paste** to paste the command to the Syntax Editor and then run the command, or
- 2) run the command directly from the dialog box by clicking on the **Open** button).

After running this command, the data in the file is available to SPSS 10 for Windows in the Data Editor window.

Two views of the data are available in the Data Editor window. **Data View** displays the actual values for the variables in the data file. **Variable View** displays the data dictionary which includes variable labels, value labels data type and other information. To switch between the views, click on the tab at the bottom of the screen.

You will often get a data file, compute new variables, make transformations, and finally save the modified set of data to use at another time. For example, you might retrieve a data file with land area per crop, add to it production per crop from another file, and then calculate yield. If you want to use these new production and yield variables at a later time, you must make sure that the data file is saved with the new variables in it. To save a data file, make the Data Editor the active window, select **Save As...** from the **File** menu and give the file a new name. Note, you **must** be in the Data Editor window to save your data unless you run a `SAVE OUTFILE` command from the Syntax Editor. You may choose to write over the old file by saving the file to the same file name.

C) The Viewer

SPSS 10 for Windows automatically writes all messages and output that result from the execution of your commands to the Viewer. For example, if you run a frequency command, then the frequency table you specify will be written to the Viewer. Similarly, if you generate a table or a graph, the table or graph will appear in the Viewer. To save the contents of the Viewer to a file, make the Viewer active, pull down the **File** menu and select **Save As....** When you give the file a name, SPSS will automatically attach the *extension* `.SPO`. It is very important to save the *output file*. The Output file gives you access to your results after your SPSS 10 for Windows session has ended. For example, you can print the output of your session in order to examine the results and verify for errors. In the sample session, you will see how to save the contents of the Viewer and give the file from each session a different name. One final note, you can manipulate the output produced just as if you were using a file manager (called Windows Explorer). In the Viewer there are two panes: the one on the right contains the results, the one on the left shows an outline view of the contents. From within this pane, you manage the results by copying, moving or deleting the results, hiding a table or chart, renaming titles, inserting titles or text or a chart.

Summary of the Basic File Types

Syntax files (or command files) contain commands saved in the Syntax Editor. They do not contain output or data—only commands. Syntax files are made accessible to SPSS for Windows with an **Open..Syntax** command. Like in earlier versions of SPSS for Windows, the extensions are `*.SPS` (was `*.LOG` in SPSS/PC+ (DOS)).

Output files contain statistical output, data information and presentation (tables, graphs, charts), generated by the SPSS 10 for Windows processor, given selected commands. They do not contain data. Output files are made accessible to SPSS for Windows with an **Open..Output** command. The new extensions are *.SPO (was *.LIS in SPSS/PC+ (DOS) and *.LST in SPSS 6.1.3).

Data files contain data, including original survey variables plus new created variables through various SPSS 10 for Windows commands such as the COMPUTE or AGGREGATE commands. Data files are made accessible to SPSS for Windows with an **Open..Data** command. For earlier versions of SPSS for Windows, the extensions are *.SAV (was *.SYS for SPSS/PC+).

Section 1

Basic functions: SPSS files, Descriptives & Data Transformation

Introduction

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

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

Questionnaire Tables	SPSS for Windows Data File
Main Household Section	C-HH.SAV
Table IA: Household Member Characteristics	C-Q1A.SAV
Table IV: Characteristics of Production	C-Q4.SAV
Table V: Sales of Farm Products	C-Q5.SAV
Conversion factors for computing kilograms	CONVER.SAV
Conversion factors for computing calories	CALORIES.SAV

Each of the training sections in this manual should take approximately two hours. We recommend that you complete each section in a single sitting. These tutorial materials make the following assumptions:

- You know how to use Windows with a mouse
- The six data files listed above are stored in the directory `c:\sample` on your hard disk. If you have not done so already, you need to copy the files from `sample.zip` to this directory.
- Under **Options** in the **Edit** Menu the following items are set:
 - list variables in the same order they are listed in the file
 - list commands in the output window
 - display the variable names rather than the variable labels
 - Syntax Editor does not come up at the start of the session.

You can modify any of the settings that control how SPSS works from this screen as well.

Important: Always remember to *SAVE* the changes to the data after each exercise and section, using a *new* file name. Also, you should save the syntax files and output files created during each session, using logical names, such as **sect1.sps** or **sect1.spo**. If you are not sure of any of the above, ask the person helping you to check them or check with the nearest computer service center or specialist.

Open your SPSS software. If you have not read or completed [Section 0](#), please do so now to clarify the concept of the Syntax Editor, where you **paste** or type commands, the Viewer where SPSS for Windows displays the results of your commands and the Data Editor window where the working data file and variable information are displayed.

Data Files and the Working File

Data from questionnaires that has been entered into SPSS 10 for Windows is stored in what are called data files. If we want to work with a set of data, we must open the corresponding data file, so that it is available to the program.

When a data file is opened, it is loaded from the disk into memory (the computer's "RAM"), making it the working file. This means that the data from this file is now available for you to use. You can have only one data file open per session. With SPSS 10, however, you can open more than one session. If you want to look at a different file, open another session of SPSS 10 and then open the data file.

Let's start with the questionnaire for Table IA: Household Member Characteristics. The data file that corresponds to it is C-Q1A.SAV. To open this file, perform the following steps:

1. From the **File** menu, select **Open...**, select **Data**
This will open the Open File dialog box.
2. Change to the directory where your sample session data are and select the file **c-q1a.sav**.
3. Click on the **Paste** button to place the command in the Syntax Editor.
The Syntax Editor will now become the active window and you will see the text

```
GET  
FILE='C:\sample\C-Q1A.SAV' .
```

in the Syntax Editor.
4. Click on the Run **O** button on the Toolbar.
Note that the GET FILE command you just ran will be written to the Viewer.

The data editor becomes the active window and the household-member data file is now in memory.

One key thing we often want to know about a data file is what variables it contains. We can find this out, along with other information, by using the **Variables...** command on the **Utilities** menu, in both the Syntax Editor and the Data Editor. You can browse through the variable definitions and variable labels. To do this, perform the following steps:

1. From the **Utilities** menu select **Variables...**
2. Select a variable name - the information about that variable will appear to the right.

This display shows definition information about each of the variables. We see the variable names: **district**, **vil**, **ca1**, **ca2**, **ca4**, **ca5**, **ca6**, and **univ**; the value labels for variables, the type of variable (numeric, string, date, etc.), the display width of the variable in characters, the number of decimal places (if Type is Numeric), and any values defined as user missing values.

Click on the **Close** button when you are finished exploring this window.

To write all of this information to your Viewer for documentation purposes or to exam at a later time, do the following:


Pull down the **Utilities** menu and select **File Info**.

This command will execute immediately. The Viewer will become active and will contain a listing of all the variables with their definitions.

You can see the name of each of the variables, their labels, and the various formats. For example: F8.2 means a variable can contain 5 numbers to the left of the decimal and 2 numbers to the right of the decimal. The decimal point counts as part of the total width. This tool is an excellent way to begin to document the contents of the data file. You can copy this information to a word processing file to begin the documentation process.



If you want to look at the structure of each variable, there is a new way in SPSS 10. In the Data Editor window, select the **Variable View** tab at the bottom left on your screen, rather than the Data View. You can directly change the characteristics of your variables here, just as you can change values in your data in the **Data View** window. As an example, the variable **DISTRICT** is shown in Table 1.1 on the next page, with a brief explanation of the choices in each column.

If you want to modify one of the parameters about a variable, click on the cell. If there are specific choices to be made, a small shaded box will appear in the right corner for that specific cell. Click on the box. in order to see the choices, add a new possibility, or view the other options. In some columns, such as Width, Decimals, and Column, instead of a box, arrows are shown to increase or decrease the size.

Example: For the variable **DISTRICT**, click on the column **Values**. Click in the cell for the variable (**DISTRICT**). You will see a small gray box  Click on this box.

A dialog box appears entitled: Value Labels

To add a new label for the value of 4

- type **4** in the **Value** box and press the <tab> key,
- then type Nampula in the **Value Label** box,
- click on the  bottom .
- Usually, you would select , but we don't want to keep this change so select





You can use these steps to modify or delete an existing label. Highlight the specific label and then click  or .

Table 1.1. Basic Structure of Variable View in SPSS 10

Number of the variable	Name	Type	Width	Decimals	Label	Values	Missing	Columns	Align	Measure
Explanation	Variable name	Numeric or alpha-numeric (<i>String</i>)	Space required to write variable in the data set	Number of digits to the right of the decimal	Label for the variable	Labels for the values, e.g. labels for categorical variables,	Declared user missing values (example: - 99), indicates cases that should be excluded from calculations	Display width of variable in Data View	Alignment of the data in Data View only: <i>Left, Right Center</i>	Measurement level of variable: <i>Scale, Nominal, Ordinal</i>
<i>Example:</i>										
1	district	Numeric	1	0	DISTRICT	1= MONAPO 2= RIBAUE 3= ANGOCHE	None	8	Right	Scale

¹ There are three categories of measurement level:

Scale: These are variables with values that are generally continuous or in intervals (integers) (e.g.: yield or age);

Ordinal: Values or alphanumeric variables that consist of categories with an intrinsic ordering (e.g. 1= short; 2=medium; 3 = tall);

Nominal: Values or alphanumeric variables that consist of categories with no intrinsic ordering (e.g. 1=man; 2=woman).


Descriptive Statistics - involving one variable

One of the first analyses done is to run descriptive statistics (e.g. averages, maximum, minimum, and standard deviations) for all variables. This type of analysis helps you to find data entry errors, to give you a "feel" for what your data is like, to see that missing values have been defined correctly, etc. It may be tempting to skip this step for some data sets or for some variables, but this is an important step that will almost always save time later and improve analysis. For example, finding out the average age of all respondents may not be something you are interested in knowing, but if the average age turns out to be 91.3 yrs, this would alert you that there are possibly data entry errors or something else might be incorrect.

Basic descriptive statistics can be obtained from two common SPSS for Windows commands—**Descriptives** and **Frequencies**. **Descriptives** is used for continuous variables, while **Frequencies** is used for categorical variables.

A continuous variable is a variable that does not have a fixed number of values. A categorical variable is a variable that has a limited number of values that form categories. For example, look at the Annex Table IA: Household Member questionnaire. Variable **ca3 (age)** is a continuous variable because age can take on many different values. Variable **ca2 (relation to head)** is a categorical variable because its values are limited to the categories 1-6.



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


1. Click on the Go To Data Editor  button on the Toolbar.
2. Scroll through the data.
A period in a field indicates a missing value or sysmis.

This will give you a "feel" for what your data are like. You might see obvious errors, e.g. a variable whose values are missing for all listed cases. Decide which of the variables are continuous and which are categorical (normally you would refer to the questionnaire to make this decision). You need to know this in order to select the right procedure to use for each variable. If you mistakenly perform a **Frequencies** on a continuous variable, you will probably get more output than you really want, with possibly hundreds of different "categories", one for each different value found. If you perform a **Descriptives** on a categorical variable, you will usually get meaningless results, since the average value of a variable that consists of categories has no real significance.

Descriptives






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

1. From the **Analyze** menu select **Descriptive Statistics....Descriptives**
The Descriptives dialog box opens.
2. Select **ca3 (age)** from the list on the left and click on the  button.
ca3 will move to the **Variable(s):** box on the right
3. Click on the  button to put the command into the Syntax Editor. If you do not have the Syntax Editor open, the Syntax Editor automatically becomes the active window. (If the Syntax Editor did not become the active window, you can go there by clicking on the syntax button on the Windows taskbar at the bottom of the screen.)

- Execute the command by clicking on the Run  button located on the Toolbar. (Note that this time we did not have to move the cursor since it was already positioned in one of the lines of the `Descriptives` command.)

The Viewer will become the active window and the results of the command will show. You can see that the mean for **age (ca3)** is 21.34 years.

Another useful way to examine a continuous variable is to run a Frequency command to view a histogram and the distribution of a variable.

- From **Analyze Descriptive Statistics...** select **Frequencies**.
- Select **ca3 (age)** from the list on the left and click on the  button.
- Click on  button and select Histograms, click on the  button.
- Click on  to put the command into the Syntax Editor. Click on the Syntax Editor to make it the active window.
- To execute the command, click on the Run  button located on the Toolbar.





View the distribution of ages in the data.

Save the Output File

Now that you have output in the SPSS Viewer, it is a good time to save that output file. Go to the Viewer window. Click on **File...Save As...** from the menu at the top right. In the "File Name" box, type `Session1`. Make sure that the directory is the one where you want save the output. SPSS will automatically add the extension `.SPO` to indicate an output file.

Frequencies

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

- Analyze...Descriptive Statistics...** select **Frequencies ...**
This will give you the Frequencies dialog box.
- Click the  button to clear the Variable(s): box.
- Select **ca1** from the list on the left and click on the  button.
ca1 will move to the Variable(s): box on the right
- Repeat step 3 until **ca2, ca4, ca5** and **ca6** have all been moved to the Variable(s): box.
- Click on  to put the command into the Syntax Editor. Make the Editor active.
- Execute the command by clicking on the Run  button located on the Toolbar.

The Viewer will become the active window. You will see, for example, the results for **ca1** show that 70.7% of the household members work on a farm. The results for **ca6** show that 38.0% of those surveyed are in monogamous marriages.

For a complete description of the output you receive from **Descriptives** and **Frequencies** refer to the SPSS for Windows Base System User's Guide Release 10, pages 213-221.

Explore

Another command used to produce many types of descriptive statistics is the **Explore** command. One of the most useful outputs for this statistic is that it produces a list of data that can be considered as outliers relative to the rest of the data. The **Explore** command can produce large amounts of output if you select the defaults that SPSS has determined. We will change some of the defaults to limit the output to statistics.

Run the **Explore** command on the variable **ca3** (**age**) using the following steps:

1. From the **Analyze...Descriptive Statistics** menu select **Explore...**
2. Select **ca3** from the list on the left and click on the next to **Dependent List**.
3. In the lower left corner of the dialog box is a box called **Display**. Click on the radio button (circle) next to **Statistics**.
We will get statistics only and no plots.
4. Next click on the **Statistics...** button.
The Explore: Statistics dialog box appears.
 - A. Click once on the square next to **Outliers** to put an H in the box.
*You will notice there is already an H in the box next to **Descriptives**.*
 - B. Click on the **Continue** button.
You will return to the Explore dialog box.
5. Click on **Paste** to put the command in the Syntax Editor and make it active.
6. Click on **Run**.

In the Viewer you see the Descriptives Table which shows you the standard descriptives and the Extreme Values table which shows you the five highest and five lowest values occurring for **age** (**ca3**). You can then determine if these values should be considered as outliers. The cases are identified by the case number. Refer to pages 223-230 of the SPSS for Windows Base System User's Guide Release 10 for an explanation of the **Explore** command.

Save the Syntax File

It is a good practice to frequently save your syntax files while you are working. You may need to re-run the commands on the same file after correcting a data entry error or if your computer “crashes” due to a problem with SPSS or another program. To save the file, make the Syntax Editor window the active window, select **File /Save As...** on the SPSS menu at the top right, in the File Name box, type the file name **Session1**. It is useful to save the syntax file and the corresponding output file with the first part of the file name being the same. The extension will be different. SPSS will automatically add the **.SPS** extension to the syntax file. Verify that the directory is the correct one. You must be in the Syntax Editor window to save the syntax file.

To practice what you've just learned about descriptive statistics, do the following exercise.

Exercise 1.1: Run descriptive statistics on another sample file. Use the production questionnaire - Table IV, whose data are in the file **C-Q4.SAV**.

Hints:

- a. make **C-Q4.SAV** your working data file.
- b. Use the **Descriptives** command for continuous variables, and **Frequencies** for categorical variables.
- c. **Prod** is a categorical variable.
- d. Quantities (**p1b**, **p2b**, ...) are continuous variables.

- e. Units (**p1a**, **p2a**, ...) are categorical variables.
- f. **p4** (month in which stocks ran out last year) and **p6** (month in which stocks will run out this year) are categorical variables.

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

		PRODUCT			
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	cotton	83	4.9	4.9	4.9
	peanuts	144	8.5	8.5	13.4
	rough rice	155	9.2	9.2	22.6
	bananas	50	3.0	3.0	25.5
	sweet potato	12	.7	.7	26.2
	cashew liquor	24	1.4	1.4	27.6
	sugar cane liquor	11	.6	.6	28.3
	dried cashew	2	.1	.1	28.4
	sugar cane	13	.8	.8	29.2
	cashew nut	130	7.7	7.7	36.9
	coconut	45	2.7	2.7	39.5
	beans	279	16.5	16.5	56.0
	manteiga beans	7	.4	.4	56.4
	sunflower	5	.3	.3	56.7
	oranges	13	.8	.8	57.5
	cashew fruit	44	2.6	2.6	60.1
	manioc	338	20.0	20.0	80.0
	sorghum	124	7.3	7.3	87.4
	maize	192	11.3	11.3	98.7
	"ossura"	5	.3	.3	99.0
	tobacco	4	.2	.2	99.2
	tomato	13	.8	.8	100.0
	Total	1693	100.0	100.0	

Descriptive Statistics - involving two or more variables

Crosstabs

The **Crosstabs** command produces a table that shows the distribution of cases according to their values for two or more categorical variables.

Look at the household member questionnaire in the annex section, Annex Table IA. One thing you might be interested to know is how the gender of the respondents varied by their relationship to the head of household. This would tell you, for example, how many females are heads of households. The **Crosstabs** command will produce this type of summary. Make the household member file, C-Q1A.SAV the working data file.

1. Click on the open folder button in the top left of the Data Editor Taskbar

2. Select the file `c-q1a.sav`
3. Click on **Paste** to place the command in the Syntax Editor and make it active.
4. Place the cursor anywhere on the line containing the "GET" command and click on the Run button on the Toolbar.

To use the **Crosstabs** command do the following:

1. Select **Analyze /Descriptive statistics** from the menu.
2. Select **Crosstabs...**
This will bring up the Crosstabs dialog box.
3. Select **ca2 (relation to head)** from the list on the left and click on the next to **Row(s):**
4. Select **ca4 (sex)** from the list on the left and click on the next to **Column(s):**
5. Click on the **Cells...** button
This will bring up the Crosstabs: Cell Display dialog box
 - A. In the Counts section, click on the box next to **O**bserved to place an **H** in it, if there is not already one there.
 - B. In the Percentages section, click on the boxes next to **R**ow and **C**olumn to put H's in them.
 - C. Click on **Continue**
6. Click on **Paste**
7. Run the command in the Syntax Editor.

The Crosstabs:Cell Display dialog box specifies which statistics you want displayed in each cell of the table—in this case we wanted counts, row percentages, and column percentages. (Row percentages sum to 100 across all the cells in a row, while column percentages sum to 100 across all the cells in a column. By default the **Crosstabs** command just gives counts.) The table produced by this command tells you that there are 21 female heads of households, and that 6.1% of the total number of heads of households are female.

Means

The **Compare Means** command produces statistics about continuous variables. It shows how the mean and other statistics for a continuous variable differ by the values of one or more categorical variables. The difference between **Crosstabs** and **Compare Means** is that **Crosstabs** produces counts and percentages broken down by categories of one or more other variables, while **Compare Means** produces means, number, standard deviation and other statistics broken down by categories of one or more other variables.

Suppose we want to know how the age of the respondents varied by their relationship to the head of household. If we did this with **Crosstabs** we would get a table with dozens of cells for each age represented, The information is not useful. Instead we will use **Compare Means**.

1. Select **Compare Means** from the **Analyze** menu
2. Select **Means...**
3. Select **ca3 (age)** and click on the next to **D**ependent List:
4. Select **ca2 (relation to head)** and click on the next to **I**ndependent List:
5. Click on **Paste**
6. Run the command from the Syntax Editor

This command will calculate means of the dependent variable (age - **ca3**), which should normally be a continuous variable. The means will be calculated separately for each different value of the independent variable, which is a categorical variable (relation to household head - **ca2**).

From this output you find that the average age of the head of household is 41.5 years while the average age of the spouse is 33.2 years.

Data Transformations

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




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

Recode

To categorize a variable, you use the **Recode** command under **Transform**. Categorizing a continuous variable makes detailed information more general. If you want to keep the detailed information as well as the new general information, you must recode the variable into a different variable. If you recode into the same variable the original values will be lost.

In this particular file, if you use the **Recode Into Same Variable** command to transform **ca3 (age)**, **ca3** will take on the new categorical values assigned in the Recode statement, and the original ages will be lost. Since we want to preserve the original ages and store the categorized values in a separate variable, we will **Recode Into A Different Variable**.

Let's recode the variable **ca3** into a new variable called **age_gp** for age groups.

1. Select **Recode** from the **Transform** menu
2. Select **Into Different Variables...**
3. Select **ca3** from the list on the left
4. Click on the  next to Input Variable -> Output Variable: box
ca3 should move to the Input Variable->Output Variable: box and the name of the box will change to Numeric Variable -> Output Variable.
5. Click once in the empty box next to **Name:** in the Output Variable section to put the cursor there.
- 6.. Type **age_gp** in the box.
7. Click once in the empty box next to **Label:** in the Output Variable section.
8. Type **Age Group** in the box.
9. Click on  to have the variable name and label changes take effect.
10. Click on 
The Recode into Different Variables: Old and New Values dialog box will appear.

11. In the **Old Value** section click on the circle next to **Range:** through
Your cursor should be in the first box.
12. Type **0** in the first box. Press <Tab> and type **10** in the second box.
13. Press <Tab> twice.
*Your cursor will now be in the box next to **Value:** in the **New Value** section.*

OR

you may press the <Alt> key leaving your finger on the key while you press the “I” key to bring you to the **New Value** box.

14. Type **1** for the first age group.
15. Click once on **Add**
16. Click on the first box after **Range:** and repeat steps 11 through 16 to recode ages **11 thru 19** to **2** and ages **20 thru 60** to **3**.
17. To recode ages **61 and up** to **4**, click on the circle next to **Range:** through highest
18. Type **61** in the box and repeat steps 13 through 15 using **4** for the value.
19. Click on **Continue**
20. Click on **Paste**
21. Select the following text in the Syntax Editor

```
RECODE
  ca3
  (0 thru 10=1) (11 thru 19=2) (20 thru 60=3) (61 thru Highest=4)
  INTO age_gp .
VARIABLE LABELS age_gp 'age group'.
EXECUTE .
```


22. Run the command

Recode changes the values for **age_gp** to the codes we want to use—1, 2, 3, and 4. Switch to the Data Editor to view that the changes were made and verify that the recoding was done correctly..

To switch to the Data Editor window (*we will use a different method than we used earlier*):

1. Click on **Window** from the menu and select **c:\sample\q1a.sav**.
2. Scroll through the Data Editor with the scroll bars.

SPSS's standard format for displaying a numeric variable includes two decimal places, which is inappropriate for a variable we know will always have an integer value. To change the display format of **age_gp** to the same format as our other variables, click on the Variable View tab at the bottom of the screen.

1. Switch to the Data Editor window if you are not already there.
2. Select the Variable View tab from the bottom left bar.
3. Find the variable **age_gp** on line 12 and in the column for Type click in the box with “Numeric”.
4. Click on the small gray box  that appears to the right. The Variable Type dialog box for the variable **age_gp** will appear.
 - A. In the box next to **Width:** type **1**.
 - B. In the box next to **Decimal Places:** type **0**

C. The circle next to Numeric should be selected. If it isn't, select it.

D. Click on **Continue**.

*You do not have the option to **Paste** this command. The changes are made immediately when you click on **OK**.*

5. Click on **OK**.

Another method that you could have used was to change the size of the variable in the “**Width**” and “**Decimals**” columns. You must first change the decimals to 0 and then change the width to 1. These changes tell SPSS to display **age_gp** with a width of 1 digit and no decimal places.

When you recode into a new variable, the new variable does not have a variable label or value labels. The statistical output from SPSS always includes the names of the variables being analyzed, but sometimes the name of a variable does not tell us as much as we would like to know. Since names are limited to eight characters, they may not be descriptive enough for us to remember the complete question from the questionnaire (e.g. the variable **ca1** is **work on a farm or not**). The name also does not tell us what the individual values of a categorical variable refer to (e.g. **ca4** is **sex** and a value of 1 indicate male and 2 indicates female).

To make the output more understandable, we can add Variable Labels and Value Labels. To avoid confusion and mistakes, you should always add labels for any computed variable that you are going to save to be used later. The best time to add labels is immediately after you create the new variable. The **Recode** command facilitates adding a Variable Label during the building of the command. Value Labels must be added using other methods:

1. You should still be in the Variable View window from the last set of steps.
2. In the box in the “**Label**” column for the variable **age_gp**, you should see the text “Age Group” because it was included in the command..
3. If there is no text in the Label: box, type the text “Age Group” there.
4. Go to “**Values**” column for **age_gp** where it says “None”.
5. Click on the small gray box **..** once to enter the Value Labels dialog box.
6. Type **1** in the Value box, then hit <Tab> to move to the Value Labels box and type **0 to 10** in that box.
7. Click on **Add**.
You will have noticed there are two other options available as well, **Remove** to delete a value and value label set, and **Change** to modify the value label for a specific value.
8. Repeat steps 6 and 7 using the following information:
Value: Value Label:
2 11 to 19
3 20 to 60
4 61 and older
9. Click on **OK**.
10. Go to the Data View window and you will see that **age_gp** is now displayed as a single digit.
11. Select **Variables...** from the **Utilities** menu.
12. Click on **age_gp** to verify the changes you just made.
13. Click on **Close** when you are finished.

This new variable is not yet part of the data file stored on disk. We must save the file in order for this variable to be included permanently in a new data file. It is a good practice to save a file under a different name in case we want to go back to a previous version of a file. For this reason we will use the **Save As** command from the **File** menu with the new file name Q1A-AGE.SAV.

1. Make sure the Data Editor window is the one in front (the active window).
2. From the **File** menu select **Save As...**
The cursor should be in the box under File name: above the Save as type: SPSS (.SAV) drop-down box. Typing while that area is highlighted will automatically delete the current text.*
3. Type **q1a-age** (The .sav extension will be added automatically.)
4. Paste and run the command.

Now each time the data file Q1A-AGE.SAV is opened, the **age_gp** variable will be included.

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

1. Use **Analyze /Descriptive Statistics... /Crosstabs...** from the menus.
2. Use **age_gp** for Rows and **ca2 (relation to head)** for Columns.
3. Check the proper selections in the Cells choices at the bottom, for we want both Row and Column percentages.
4. Paste the command and run it.

From this analysis, you can see that 12% of heads of households are 61 years of age or older. Also, that of the people who are 61 years or older, 83.7% are heads of households.

Compare the information you get from this **Crosstabs** analysis with the information from the **Compare Means** command performed on **ca3 (age)** earlier. To do this, we will explore SPSS's ability to switch between the Syntax, Viewer, and Data windows.

To switch to the Viewer:

1. From the **Window** menu select Session1 - SPSS Viewer
2. Scroll back through the Viewer window with the scroll bars.
3. Find the Crosstabs table and compare with the Compare Means table.

To switch to the Syntax Editor:

1. From the **Window** menu select Session1 - SPSS Syntax Editor.
2. Scroll through the Syntax Editor window with the scroll bars.

To switch to the Data Editor:

1. From the **Window** menu select q1a - SPSS Data Editor.
2. Scroll through the Data Editor window with the scroll bars.

Please note it is also possible to switch from one window to another by clicking on the SPSS icons in the Windows taskbar, found usually at the bottom of the screen (the taskbar may be moved to any side of the screen).

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

Exercise 1.2: Using the Household Data and Questionnaire (latter available in the annex), find out the number of households in each district that have 1-4, 5-7, and more than 7 persons per household. One way to find out this information is to create the following table.

- Hints:
- Use the file C-HH.SAV.
 - Recode **h1** into **hsize** using the following groups: (1 thru 4) (5 thru 7) (8 thru Highest).
 - Add a variable label and value labels.
 - Run **Crosstabs** on this variable by **district**

Household size * DISTRICT Crosstabulation

			DISTRICT			Total
			MONAPO	RIBAUE	ANGOCHE	
Household size	1.00	Count	65	48	74	187
		% within Household size	34.8%	25.7%	39.6%	100.0%
		% within DISTRICT	60.7%	40.3%	64.3%	54.8%
		% of Total	19.1%	14.1%	21.7%	54.8%
	2.00	Count	39	56	36	131
		% within Household size	29.8%	42.7%	27.5%	100.0%
		% within DISTRICT	36.4%	47.1%	31.3%	38.4%
		% of Total	11.4%	16.4%	10.6%	38.4%
	3.00	Count	3	15	5	23
		% within Household size	13.0%	65.2%	21.7%	100.0%
		% within DISTRICT	2.8%	12.6%	4.3%	6.7%
		% of Total	.9%	4.4%	1.5%	6.7%
Total		Count	107	119	115	341
		% within Household size	31.4%	34.9%	33.7%	100.0%
		% within DISTRICT	100.0%	100.0%	100.0%	100.0%
		% of Total	31.4%	34.9%	33.7%	100.0%

Looking at the results for Monapo:

34.8% of all 1 to 4 member households (group 1) are found within Monapo and 60.7% of all households in Monapo have 1 to 4 members in a household.

Before exiting SPSS for Windows we should save the contents of the Viewer. The output window contains all of the command and the results of these commands. It is useful to keep this output in a file so you can review it later, print it or include it in a report.

- Make the Viewer the active window using its icon in the Windows taskbar.
- From the **File** menu select **Save As...**
- Type the filename **session1**
The .spo extension will be added to the name automatically.
- Click on **Save**

To exit SPSS for Windows:

1. From the **File** menu select **Exit SPSS**
A dialog box will prompt you to save the contents of C:\sample\c-hh.sav
2. Click on **No**
A dialog box will prompt you to save the contents of Syntax Editor to Syntax1.
3. Replace the proposed name **Syntax1** with a filename such as **Sect1.sps**. Click on **Save**

SPSS for Windows will close.

Section 2

Restructuring Data Files - Table Lookup & Aggregation

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

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

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

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

Districts	Calorie Production Quartile			
	1	2	3	4
Monapo				
Ribaue				
Angoche				

The data in their current form cannot answer the question. Therefore, many transformations are required to produce this table. This is a typical example of the complications you will encounter in real-world data analysis. This entire section will be devoted toward the goal of creating this table.

To begin, let's first take a look back at some of the files that we have and at the variables we need to use from each of these:

- **C-Q1A.SAV:** This file contains data on household member characteristics. It is at the household-member level. We need to use the variables ca3 (age) and ca4 (sex) in this exercise to compute the number of adult equivalents per household.
- **C-Q4.SAV:** This file contains data on crops produced by the household. The variables we need to calculate the total production of the household are:
 - a. **prod** - codes for the agricultural crop produced.
 - b. **p1a** - codes for the unit in which the production was measured (100 kg sack, 50 kg sack, etc).
 - c. **p1b** - number of units produced this year.

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

- **CONVER.SAV:** This is a table-lookup file. This file was created specifically to handle the problem of converting non-standard units to a standard unit. For each product-unit combination there is a conversion factor to convert the measurement to equal the weight in kilograms. In other words, there is a different conversion factor for each product-unit combination. For example, the conversion factor for a 50 kg sack of rice is 39.4; for a 50 kg sack of cotton it is 17.5, while a 50 kg sack of manioc root is 33.33. The variables in this file are:
 - a. **prod** - product (crop) code
 - b. **unit** - unit of measure
 - c. **conver** - conversion factor (equal to the number of actual kilograms for the combination of prod and unit)

Below, a sample of data from **CONVER.SAV** shows that

rice (**prod**=7) measured in a 20 liter can (**unit**=8) weighs 19 kg;
 rice (**prod**=7) measured in a 50 kg bag (**unit**=24) weighs 53 kg;
 beans (**prod**=30) measured in a 20 liter can (**unit**=8) weighs 17 kg;
 beans (**prod**=30) measured in a 50 kg bag (**unit**=24) weighs 47 kg.

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

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

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

1. We need to know **how many calories each household produced for the year**. We can generate a file with this information using data we have stored in three places—the production file, **C-Q4.SAV**, and two table-lookup files, **CONVER.SAV** and **CALORIES.SAV**.
2. We need to know **how many adult equivalents are in each household**. We can generate a file with this information using data from the member file, **C-Q1A.SAV**.

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

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

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

First, all production is currently measured in non-standard units whose weight is different for each product. Thus, we must first **convert all production into kilograms**.

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

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

With these points firmly in mind, let's begin by opening C-Q4.SAV.

1. Select **File... Open... Data...**
2. Select the file name **c-q4.sav**
3. Paste and run the command.

Let's first **convert all production of the crops into kilograms**. To find the conversion factor appropriate for each case in the production file (C-Q4.SAV), we need to look up the product and unit in the CONVER.SAV file. We will create a new file where each case has both the data from the production file and a variable containing the conversion factor for that product-unit combination. In SPSS for Windows, the command to do this is **Data /Merge Files /Add Variables**.

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

1. From the **Data** menu select **Sort Cases...**
The Sort Cases dialog box will come up.
2. Select **prod** and click on
3. Select **p1a** and click on
4. Paste and run the command.

The files are now ready to be merged. **Merge Files** requires at least two files as input. In this case, the two files are working data file and CONVER.SAV. We are doing a “**File - Table**” merge where the second file is our “**Lookup Table**”. The file created by **Merge Files** will become the working data file, replacing the current one.

1. From the **Data** menu select **Merge Files**, then select **Add Variables...**

- The Add Variables: Read File dialog box will come up.*
2. Select the filename `CONVER.SAV`
 3. Click on **Open**

The variables used to match cases must have the same names. First select **p1a** from the “New Working Data File” and move it into the box for “excluded variables”. We will **rename it to unit** and we will be able to use it as a variable to match the cases.

4. Select **p1a** from the list under New Working Data File: and click on **O**
5. Click on **Rename...**
*Now you to rename **p1a** to **unit** to match the conversion file.*
6. Next to New Name: type **unit**
7. Click on **Continue**

We cannot select the **variables to match by** until we **indicate we want to match cases on key variables**.

8. Check the box next to Match cases on key variables in sorted files
9. Click on radio button next to External file is keyed table
10. Select **prod** from the Excluded Variables: list
11. Click on **()** next to Key Variables: (bottom, right side of dialog box)
12. Repeat steps 10 and 11 for **unit**
13. Paste the command
A warning will come up telling you the data files must be sorted. Since we have sorted the files...
14. Click on **OK**
A dialog box will ask you if you want to save the contents of the Data Editor window. We do not want to save it, the new file can take its place, so...
15. Click on **NO**
16. Select both the “MATCH” and “EXECUTE” commands and run them.
The actual SPSS command for the Merge Files is “MATCH FILES”.

The above steps tell SPSS for Windows to merge the working data file (active in your Data Editor window) and the `CONVER.SAV` file, (using `CONVER.SAV` as a table lookup) to add the unit variable to our working data file. Since the key variables need to have the same names in both files we renamed **p1a** (the “unit” variable for our working file) to **unit** (**p1a** remains **p1a** in the file `c-q4.sav`).

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

The new working file produced by the merge now contains the needed conversion factor variable, **conver**. For every product-unit combination, **conver** is equal to the number of kilograms in that unit. It is always important to verify if the merge was successfully completed. Check the Viewer to see if there are any errors in the log and then return to the Data Editor and look at some cases to verify that the conversion factors match the products. For example, a 20-liter can filled with maize grain actually weighs 18 kilograms of maize grain. Check to see that when **PROD**=47 and **UNIT**=8, **CONVER**=18.

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

1. From the **Transform** menu select **Compute...**
2. Under **Target Variable:** type **qprod_tt** (for **total quantity of production in kg**)
3. Click on **Type & Label** to add a variable label for **qprod_tt**, then select **Continue**.
4. From the list on the left of the Compute Variable window, select **p1b** and click on **()** to put it in the right hand window, the Numeric **E**xpression: box.
5. The cursor will now be in the Numeric Expression: box. Type ***** or select the button in the dialog box to add the multiplier sign next to **p1b**.
6. From the list on the left select **conver** and click on **()**.
*The expression should look like p1b * conver.*
7. Paste, select and run the command

For the next part of step 1, we need to find out how many calories are in each kilogram of each product. The information is in the table-lookup file **CALORIES.SAV**. This file has two variables—product (**prod**) and number of calories per kilogram (**calories**). The key variable is **prod**. To get the variable **calories** into the working data file we need to do another merge with a keyed table lookup. This time the key variable will only be the product variable. Our working data file has already been sorted by product (because of the procedure to merge the kilogram information), so we don't need to sort it again.

1. From the **Data** menu select **Merge Files** then **Add Variables...**
2. Select the file **calories.sav** and click on **Open**
3. Check the Match cases... box
A. Check the **E**xternal file is keyed table box
4. Put **prod** in the Key **V**ariables: box
5. Paste the command
6. Clear the warnings as necessary
7. Select and run the command

The new working data file produced by the merge should now contain the needed calorie variable, **calories**. Check to make sure. In the **calories** variable for maize grain (**prod=47**) the value should be 3590. We are now ready to compute “total calories produced”.

1. Use **Transform /Compute...**
2. Type **kprod_tt** as the name for the **T**arget Variable: (for “Total calories produced”)
3. Click on **Type & Label** to add a label for **kprod_tt** here, if you wish, then select **Continue**.
4. Click in the Numeric **E**xpression box and type this equation **qprod_tt * calories**
5. Paste, select and run the command

This gives us a new working data file with total calories produced per product for each household. We are only interested in the seven staple food crops:

peanuts (prod=5)
rice (prod=6)
nhemba bean (prod=30)

manteiga bean (prod=31)
manioc (prod=41)
sorghum (prod=44)

corn (prod=47)

We can find these product coding by looking at **prod** in the questionnaire. Since we are only interested in those products, we can filter for just those cases. To make only these cases active we use **Select Cases**. **Select Cases** selects a subset of the cases based on particular criteria.

Select Cases has two options:

1. you can either “filter out” the unselected cases (a cross hatch appears in the case number column to indicate that the case has been “filtered”). With the filter command you can turn the filter off so that all the cases are available for analysis.

OR

2. “delete” the unselected cases. The “delete” selection removes all the cases that do not match the criteria. The cases that are deleted are no longer available for any kind of analysis. You must be very careful if you select this option and then want to save the file. You must save the selected cases to a NEW FILE NAME!!! If you save the selected cases to the same file name, you will lose all the unselected records permanently!

We will use the **Select Cases** with the “filter” option.

1. From the Date Editor window, select **Data /Select Cases**
*You should see the **Select Cases** dialog box.*
2. Select the round button next to If condition is satisfied
3. Click on the **If...** button directly under If condition is satisfied
4. Click **in** the box, to the right of **()**, **not** on the button itself.
5. Type the following text (without hard returns):
PROD = 5 | PROD = 6 | PROD = 30 | PROD = 31 | PROD = 41 | PROD = 44 |
PROD = 47

*The “|” are symbols for the word OR. We are telling SPSS to select all cases with **prod** equal to 47 or **prod** equal 30 or **prod** equal 31...*

6. Click on **Continue**
7. Select the round (radio) button next to Filtered (so that the cases with the other products are only temporarily not being used and may be brought into an analysis later).
8. Paste the command
9. Select the text (highlight it) in the Syntax Editor from the line with USE ALL to the line with EXECUTE and run the command.

Only cases with these product codes will now be used for **analysis** and saving the file to a new name. This subset of the data will be in effect until we open another file or use the **Data Select Cases** to **Select All** cases (unfilter the cases). Any transformations that are done will affect all the cases - transformations are not limited to just changing the selected cases.

The next piece that is necessary to be able to produce the final table is that we need to know how many calories were produced per household for all staple food products combined. To do this, we need to sum, for each household, the values of kprod_tt for all of the food crops the household produced. We need to add values “across cases” rather than “within a case”. In other words, we need to create a new file at the household-level where we have one case per household. This file is at the household - product level file where we can have many cases per household. SPSS uses the term “**AGGREGATE**” to calculate across cases and collapse the

number of cases at one level to a new level. We will sum all the cases for household-product to one case for household.

To create the new household-level file, we use the **Aggregate** which can be found under **Data**. The **Aggregate** command will create a new data file with one case per household and **kprod_tt** summed across the products for each household. It always uses the working data file as the file to be aggregated. We already have the production file open, so we are ready to aggregate.

1. From the **Data** menu select **Aggregate...**
The Aggregate Data window will appear.
2. Select **district**, **vil**, and **hh**, in that order, for the **Break Variable(s)**:
3. Select **kprod_tt** as the **Aggregate Variable(s)**:
4. Click on **Name & Label...**
 - A. Change the default name **kprod__1** to **kprod_tt**
 - B. Type the following label: **Calories Produced in Staple Foods**
 - C. Click on **Continue**
5. Click on **Function...** and
 1. Select “**Sum of values**” and click on **Continue**
6. Select “**Replace working data file**”, and click on **Paste** to past the command to the Syntax Editor.
8. Click on **No**. We do not want to save the contents of data window
9. Switch to the Syntax Editor and run the command.

If we had selected “**Create new data file**” instead of “**Replace working data file**”, the new aggregated data file would have been stored on disk. It would not have become our working file. In order to make the new aggregated data file the current working data file, we would have had to open the data file to access it.

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

Aggregate Variable(s) box is where you specify the variables and the type of function that is to be used for these variables. A new variable **kprod_tt**, was created in the aggregated file which summed the old variable **kprod_tt** (total calories produced) across all cases (the different food crops) for each household. The only variables which are contained in an aggregated file are the “break” variables and any new aggregated variables created (e.g. **kprod_tt**) from the specifications in the **Aggregate Variable(s)** box..

The new working data file now contains what we need -- total number of calories from staple foods produced per household. To be sure this new variable exists, do a **Descriptives** on **kprod_tt**. You should find that the average number of calories produced per household per year is **4,483,964.7**.

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

1. Make the Data Editor window active.
2. Use **Save As...** from the **File** menu
3. Name the file **hh-file1**
4. Paste and run the command.

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

The data needed to calculate adult equivalents per household is in the member file, c-q1a.sav.

1. Click on the open folder button on the SPSS Data Editor Taskbar
2. Select the file name c-q1a.sav
3. Paste and run the command.

The rules we will use for calculating adult equivalents for this survey are:

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

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

Compute /if... is the tool to use to make this calculation. The adult equivalent variable to be created is **ae**.

1. From the **Transform** menu select **Compute...**
The Compute Variable window will appear.
2. For the **Target Variable:** type **ae**
3. Select the **Type & Label** box and type **Adult equivalent** in the Label. Click on **Continue**
3. In the Numeric Expression: box type a **1**
4. Click on **If...**
 - A. Select the radio button for **Include if case satisfies condition:**
 - B. Type the statement **ca4 = 1 & ca3 >= 10**
 - C. Click on **Continue**
5. Paste the command but don't run it yet.
6. Repeat steps 1, and 3-8 replacing the previous information with the following.
You are not obliged to use the menus within SPSS. Once you have a set of commands that you have pasted to the Syntax Editor, it may be easier to simply copy and paste the same command within the Syntax Editor itself and then edit the commands. If you prefer not to use the copy/paste manoeuver in the Syntax Editor, just repeat the steps above as indicated.

Numeric Expression	If... Statement
0.84	ca4 = 2 & ca3 >= 10 & ca3 <= 19
0.72	ca4 = 2 & ca3 >= 20
0.60	ca3 < 10

7. Select all of the If statements and run.

Your syntax should look like this:

```
IF (ca4 = 1 & ca3 >= 10) AE = 1 .
IF (ca4 = 2 & ca3 >= 10 & ca3 <= 19) AE = 0.84 .
IF (ca4 = 2 & ca3 >= 20) AE = 0.72 .
IF (ca3 <= 10) AE = 0.60 .
VARIABLE LABELS AE 'ADULT EQUIVALENT' .
EXECUTE .
```

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

1. You will need to select **Analyze /Descriptive Statistics /Frequencies...**
2. Use **ae**
3. Paste and run

You should see there are **1524 total cases**. Ideally there should be four values represented in the table — 1, 0.72, 0.84, and 0.60 — and no missing cases. However, we do have nine (9) missing cases. Our data file is missing either the age or the sex for nine people. Such missing data should have been identified during the cleaning process. Ideally the analyst should go back to the original questionnaires and try to find the correct values to replace the missing data.

If we leave these values missing, the sizes of our households will appear to be slightly smaller than they actually are, which will distort our results. We could avoid this problem by eliminating the households of those nine individuals from our analysis, but then we can't use the information about the food production from those households. Instead, we will try to make a reasonable assumption about those nine missing members. We know that the adult-equivalent values range from a low of .6 for children to a high of 1.0 for adult males, which is not a very wide range. Let's find out the average adult-equivalent value for our sample.

1. **Analyze /Descriptive Statistics /Descriptives...**
2. Variable is **ae**
3. Don't forget to paste before you run the command

The analysis shows that the mean value of **ae** for all individuals is **.79**, with a standard deviation of only **.17**. We will assume that the nine individuals with missing age or sex codes are all "average" individuals, and assign them the adult-equivalent value of **.79**. (**Warning:** be very cautious about "filling in" missing data this way, because careless use of this technique can give you misleading results. We are using this example strictly as an illustration of SPSS commands and not recommending that you do this routinely to compensate for missing data.).

1. **Transform /Recode /Into Same Variables...**
Recode into Same Variables dialog box will appear.
2. Move **ae** to Variables:
3. Click on **Old and New Values...**
 - A. Select System-missing
 - B. Select Value: in the New Value section and type **.79** in the box
 - C. Click on **Add**
 - D. **Continue**
4. Paste, select and run

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

1. From the **Data** menu select **Aggregate...**
2. Move **district**, **vil**, and **hh** to **Break Variable(s)**:
3. Move **ae** to **Aggregate Variable(s)**:
4. Click on **Name & Label...**
 - A. In the **Name:** box type **ae_tt**
 - B. In the **Label:** box type **Adult Equivalents**
5. **Continue**
6. **Function...**
 - A. Select **Sum of values**
 - B. **Continue**
7. Select **Replace working data file**
8. Paste, clear warnings and run.

Aggregate creates a new working file. The new working data file is at the household level, with one case per household. The variable **ae_tt** is the total adult equivalents for that household. To verify that this variable was created, run a **Descriptives** on **ae_tt**.

1. **Analyze /Descriptive Statistics /Descriptives...**
2. Select **ae_tt**.
2. Paste and run.

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

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

1. Make sure the Data Editor is active
2. **File /Save As...**
3. Filename **hh-file2**
4. Paste and run.

Step 3: We need to join the two files created in steps 1 & 2 together in order to compute calories produced per adult equivalent.

We have HH-FILE1.SAV containing the **calorie-production data** for all households, and we have HH-FILE2.SAV containing the **adult-equivalent data** for all households. We need to combine these files on a case-by-case to get both sets of data in a single file to be able to use the variables in the next calculation. To do this, we use **Merge Files**, but this time neither of the files are “keyed” tables.

We noted earlier that key variables are required for any merge that includes a keyed table lookup. When you are joining two files at the same level, as we are about to do, it may not seem important to include key variables, but it is. The key variables determine which cases are to be combined.

You should never use **Merge Files** without **Key Variables**. SPSS will not know how to match the cases between the two files if you do not specify the key variables. Without the key variables, the files are matched

by case - case 1 is matched to case 1, case 2 is matched to case 2 and so on. Without the key variables, if a household is missing in one file, then the cases for each file will no longer be matched to the correct household. The command will execute without any warnings or error messages, but the results may be incorrect.

Note: hh-file2.sav is still the working file

1. **Data /Merge Files /Add Variables...**
2. Select file `hh-file1.sav` for the Read File
3. **Open**
4. Select `Match cases on key variables...`
5. Select `Both files provide cases`
6. Key Variables: are **district**, **vil**, and **hh** respectively
7. Paste, clear warning, select and run.

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

1. **Transform /Compute...**
2. Target Variable: **`kprod_ae`**
3. **Type & Label...**
4. Label: **Calories produced per adult equivalent per day**
5. **Continue**
6. Numeric Expression: type in **`kprod_tt / ae_tt / 365`**
7. Paste, select and run

Before we can produce the table we want, we have to create one more variable, denoting which calorie-production quartile each household falls in within their district. **Rank Cases** can do this for us. **Rank Cases** computes a new value for each case in a new variable. This value shows how that case ranks within a group according to the value of another variable. In this case, we want to classify each household by how it ranks within its district in terms of calories produced per ae. Specifically, for each district, we want to break the households into four groups of equal size (quartiles), from lowest to highest calorie production. A new variable containing values from 1 to 4 will indicate to which quartile each household belongs.

1. **Transform /Rank Cases...**
2. Move **`kprod_ae`** to Variable(s):
3. Move **district** to By:
4. Click on **Rank Types...**
 - A. Unselect Rank
 - B. Select Ntiles: **4**
 - C. Click on **Continue**
5. Paste and run

*Note the new variable name in the Viewer; it should be **nkprod_a***

The first thing we specify is the variable containing the values to use for the ranking — in this case **`kprod_ae`**. Then we need the By variable to specify the variable(s) that define the groups — in this case **district**. **Rank Cases** has a number of different methods of ranking. We're using one of the simplest — `/NTILES(4)` tells SPSS for Windows to break the variable into quartiles. From this command, SPSS for Windows will create a

new variable that will contain the rankings and generate a name for that variable using the name of the variable that was used to determine ranking and adding an “n” at the beginning — **nkprod_a**.

We can now use **Means** to get the numbers to fill in our table.

1. **Analyze /Compare Means /Means...**
2. Move **kprod_ae** to Dependent List:
3. Move **nkprod_a** to Independent list: layer 1 of 1
nkprod_a came from the Rank Cases procedure.
4. **Next**
5. Move **district** to Independent List: layer 2 of 2
6. Paste and run

You should note that mean for the entire population is **4014.5183** and the mean for the 2nd quartile in Ribaue is **2517.4551**. The output from **Compare Means** calculates the numbers needed to complete the table that we wish to produce, although it is not formatted exactly as we showed the table at the beginning of Section 2.

Save this file as HH-FILE3.SAV.

1. Make the Data Editor active
2. **File /Save As...**
3. Filename is hh-file3
4. Paste and run

You should now save the contents of the Syntax Editor to a permanent command file for later use.

1. Make the Syntax Editor active
2. **File /Save As...**
3. Use the filename **session2**
The .sps extension will be added automatically.

This file now contains all the commands from the Syntax Editor. Whenever you do any substantial amount of work, you should always save the contents of the Syntax Editor to a command file. You may have noticed that throughout the Sample Session we could have run the commands by clicking on **OK** instead of **Paste**. Pasting commands into the Syntax Editor and then running them, rather than running them directly, documents your work and enables you to run the exact same analysis over again another time. Documenting now can save many steps later.

So now let's see how you would retrieve the command file you just created. To exit SPSS for Windows:

1. **File /Exit SPSS**
SPSS will prompt you to save the contents of the windows that have not been saved; in this case the Viewer.
2. Save the Viewer as **session2**

Start SPSS for Windows again. To open our command file:

1. **File /Open /Syntax...**
 2. Select the file: **session2.sps**
 3. **OK**
- The Syntax Editor c:\sample\session2.sps will be active

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

Your SESSION2.SPS should look similar to this:

```
GET
  FILE='C:\SAMPLE\C-Q4.SAV'.
USE ALL.
COMPUTE filter_$=(prod=47 or prod=30 or prod=31 or prod=41 or prod=6 or
  prod=44 or prod=5).
VARIABLE LABEL filter_$ 'prod=47 or prod=30 or prod=31 or prod=41 or prod=6'+
  ' or prod=44 or prod=5 (FILTER)'.
VALUE LABELS filter_$ 0 'Not Selected' 1 'Selected'.
FORMAT filter_$ (f1.0).
FILTER BY filter_$.
EXECUTE .
SORT CASES BY
  prod (A) pla (A) .
MATCH FILES /FILE=*
  /RENAME pla=unit
  /TABLE='C:\SAMPLE\CONVER.SAV'
  /BY prod unit.
EXECUTE.
COMPUTE qprod_tt = plb * conver .
EXECUTE .
MATCH FILES /FILE=*
  /TABLE='C:\SAMPLE\CALORIES.SAV'
  /BY prod.
EXECUTE.
COMPUTE kprod_tt = qprod_tt * calories .
EXECUTE .
AGGREGATE
  /OUTFILE=*
  /BREAK=district vil hh
  /kprod_tt 'Calories Produced in Staple Foods' = SUM(kprod_tt).
DESCRIPTIVES
  VARIABLES=kprod_tt
  /FORMAT=LABELS NOINDEX
  /STATISTICS=MEAN STDDEV MIN MAX
  /SORT=MEAN (A) .
SAVE OUTFILE='C:\SAMPLE\HH-FILE1.SAV'
  /COMPRESSED.
GET
  FILE='C:\SAMPLE\C-Q1A.SAV'.
EXECUTE .
IF (ca4 = 1 & ca3 >= 10) ae = 1 .
EXECUTE .
IF (ca4 = 2 & ca3 >= 10 & ca3 <= 19) ae = .84 .
EXECUTE .
IF (ca4 = 2 & ca3 >=20) ae = .72 .
EXECUTE .
IF (ca3 < 10) ae = .60 .
EXECUTE .
FREQUENCIES
  VARIABLES=ae .
DESCRIPTIVES
```

```

VARIABLES=ae
/FORMAT=LABELS NOINDEX
/STATISTICS=MEAN STDDEV MIN MAX
/SORT=MEAN (A) .
RECODE
  ae (SYSMIS=.79) .
EXECUTE .
AGGREGATE
  /OUTFILE=*
  /BREAK=district vil hh
  /ae_tt 'Adult Equivalents' = SUM(ae) .
DESCRIPTIVES
  VARIABLES=ae_tt
  /FORMAT=LABELS NOINDEX
  /STATISTICS=MEAN STDDEV MIN MAX
  /SORT=MEAN (A) .
SAVE OUTFILE='C:\SAMPLE\HH-FILE2.SAV'
  /COMPRESSED.
MATCH FILES /FILE=*
  /FILE='C:\SAMPLE\HH-FILE1.SAV'
  /BY district vil hh.
EXECUTE.
COMPUTE kprod_ae = kprod_tt/ae_tt/365 .
VARIABLE LABELS kprod_ae 'Calories produced per adult equivalent' .
EXECUTE .
RANK
  VARIABLES=kprod_ae (A) BY district /NTILES (4) /PRINT=YES
  /TIES=MEAN .
MEANS
  TABLES=kprod_ae BY nkprod_a BY district
  /CELLS MEAN STDDEV COUNT
  /FORMAT= LABELS .
SAVE OUTFILE='C:\SAMPLE\HH-FILE3.SAV'
  /COMPRESSED.

```

Exercise 2.1: Produce similar output using calories retained (production minus sales) instead of calories produced. It will show calories retained per adult equivalent per day from the total of the same six food crops. The output should be broken down by district and calorie production quartile.

- Hints:
- The procedure is very similar to the work that we just completed.
 - Sales come from **c-q5.sav**.
 - Check the file for the appropriate variable for the quantity of sold production. Note that the product codes are the same as for **c-q4.sav**. Also check for the variables by which to sort.
 - Retrieve the commands from generating the previous table and check each step for needed changes. There will be changes of product code, file names, and variables.
 - Computing the calories sold involves the same basic steps as computing the calories produced. (Step 1)
 - Merge this newly created file, (the file containing calories sold), with the file containing calories produced, **hh-file3.sav**.
 - Keep in mind that only 256 households sold products, but all 343 households produced and retained calories. If the calories-sold variable is missing, it means the household did not produce food, so it should be recoded to zero.

- h. Compute calories retained = calories produced - calories sold.
- i. Rank into quartiles.
- j. Use the **Compare Means** command to show calories retained by district and quartile.
- k. Save the data file.
- l. There's no need to save the contents of the Syntax Editor, from the exercise, to a file.
- m. Execute the newly created syntax file, select all and run

This is an example of the output you should produce:

```

- - Description of Subpopulations - -

Summaries of      KRET_AE      Calories Retained per adult equivalent
By levels of      NKRET_AE      NTILES of KRET_AE by DISTRICT
                  DISTRICT      DISTRICT

Variable          Value  Label              Mean      Std Dev      Cases
-----
For Entire Population              3044.2336  2370.1465      343

NKRET_AE          1              1098.8770   401.0378      84
  DISTRICT        1  MONAPO          1148.0448   409.6144      27
  DISTRICT        2  RIBAUE          1232.8030   350.2260      29
  DISTRICT        3  ANGOCHE           912.7559   384.7468      28

NKRET_AE          2              2015.5753   297.9913      86
  DISTRICT        1  MONAPO          2211.3833   205.7199      27
  DISTRICT        2  RIBAUE          2145.8446   202.8158      30
  DISTRICT        3  ANGOCHE          1698.5099   168.4997      29

NKRET_AE          3              2946.5741   547.1454      87
  DISTRICT        1  MONAPO          3314.8568   477.1234      28
  DISTRICT        2  RIBAUE          3126.3578   329.8936      30
  DISTRICT        3  ANGOCHE          2405.0077   336.4856      29

NKRET_AE          4              6071.8027  2821.2709      86
  DISTRICT        1  MONAPO          7619.1018  3557.1354      27
  DISTRICT        2  RIBAUE          5759.0391  1649.5839      30
  DISTRICT        3  ANGOCHE          4954.7625  2426.8245      29

Total Cases = 343

```

Section 3

Importing data from other formats into SPSS

There are several methods that can be used to enter data into SPSS for Windows.

1. You can key data directly into the program.
2. You can use SPSS Data Entry for Windows, Version 2.0. An SPSS data file is created and can be read directly in SPSS for Windows. Variable and value labels defined in the Data Entry are maintained in the data file when it is opened in SPSS for Windows.
3. Data can also be read in from a variety of different file formats. SPSS for Windows can read in data stored as Lotus 1-2-3 (*.w*) and Excel (*.xls) spreadsheets, dBase (*.dbf) databases, Systat files (*.syd and *.sys), ASCII text files (*.txt) as well as old versions of SPSS for DOS (*.sys) files.

Text files: Data stored in text files in one of these formats -- delimited (tab, space or comma) or fixed width -- can be read into SPSS using the “Text Import Wizard”. The wizard guides you through the specifications necessary to import the text data. If you will be importing text data on a regular basis, you can save the specified format to a filename so that you don’t have to redefine the parameters each time. You can also paste the commands to a syntax file and reuse the syntax whenever necessary.

In this section, we will read in a database (*.dbf) because it is the format under which many market information systems enter their raw data (an example is the Senegal SIM (Système d’informations de marché)). We will also read in a spreadsheet from Lotus (*.wk1). This file contains a price index which we will use to compute real or deflated prices in Section 8 of this time series tutorial.

Opening and saving data files

The **Open /Data** command from the **File** menu retrieves an existing SPSS data file in SPSS as the “working file” or you can retrieve a file in another format.

To read data from a text or ASCII file, select **File /Read Text Data**. From the Open File dialog box, select the file to open. The default extension for these types of files is *.txt. The Text Import Wizard dialog box appears to lead you through 6 steps. In the lower part of the dialog box you will see the first few rows on the text file. The first screen prompts for a filename of a predefined format. If you don’t have one (the default answer is “no”), click on **Next**. The second screen asks you to select a format, Delimited or Fixed Width. The second question asks if you have variable names in the first row. The default is “no”. The next several screens lead you through the rest of the specifications. You have the opportunity to name the variables or you can use the default. Once you have reached “Step 6”, you can **Paste** the command into a syntax file so you can use it again, or you can save the template to a name, or you can just click on **OK**.

If your data are in a predefined format that SPSS will read, select **File /Open /Data**. In the Open File dialog you will see listed all files with the specified extension of *.sav - SPSS for Windows default data file name. You can select the file to open. If the file is in a different directory, you can switch to the directory where your files are located. Use the Look in: drop down arrow to select a drive / directory. To change the type of file that you want to open, click on the drop down arrow for Files of type: SPSS will not read the file in correctly unless you have selected the appropriate type of file.

How SPSS Read in Spreadsheet Data

An SPSS data file is rectangular. The boundaries (or dimensions) of the data file are determined by the number of cases (rows) and variables (columns). All cells have a value, even if that value is "blank" (or what is called a system-missing value). The following rules apply to reading spreadsheet data:

- S Rows are equal to cases, columns are equal to variables.
- S Variable names can be read from the first row of the file. If the variable name is longer than 8 characters, the name is shortened to 8 characters. If the variable names are not unique within the first 8 characters, then SPSS will create a unique name. For Excel 5 or later files - the original variable name is used as the variable label in SPSS if the variable name is longer than 8 characters.
- S Variable names - older versions of Excel and Lotus - If the first row does not contain variable names, SPSS uses the column letters (A, B, C, ...)
- S For Excel 5 or later files the data type and width for each variable is determined by the column width and data type of the column. If the column contains more than one data type, the data type is set to string.
- S For earlier versions of Excel the data type and width for each variable is determined by the first row in the column. Values of other types are converted to the system-missing value. If the first data cell in the column is blank, the global default data type for the spreadsheet (usually numeric) is used.
- S For numeric variables, blank cells are converted to the system-missing value, indicated by a period. For string variables, a blank is a valid string value, and blank cells are treated as valid string values.

In Section 8, we will import a data file from a spreadsheet to merge a price index to deflate nominal prices.

How SPSS Reads in Database Files

dBase files

Field names in a dBase file are translated to variable names. The field names should follow the rules for naming SPSS variables. If they are longer than 8 characters, the name is truncated. If the field name is not unique within the first 8 characters, the field is dropped! Colons are translated to underscores.

Records marked for deletion in a dBase file will be imported into SPSS. A variable is added to the file called D_R. If there is an "*" in this field, the record has been marked for deletion. If you do not want the records that have been marked for deletion imported, pack the database before you import it into SPSS.

Databases

To be able to read in a database table, you must have the database driver installed on your computer for the type of database you will use. You can install the appropriate driver from the CD-ROM installation CD. SPSS Data Access Pack contains a variety of database formats. Microsoft Data Access Pack installs drivers for Microsoft products.

Queries are used to read in the data from a table. You can save a query specification (SPSS uses the extension .spq). To access a database, click on **File /Open Database /New Query**. The Database Wizard starts and leads you through the procedure to set up a data source and then select the file, tables, and fields that you want

to use. SPSS will modify the variable names to meet SPSS's rules for variable names - if the first 8 characters are valid, that name is used; otherwise SPSS assigns a valid variable name.

Let's read in a dBase file. We will use a file from the Senegal SIM (Market Information System) on cereals for 1995.

1. Open SPSS and click on **File /Open /Data...**
2. Check to see that you are in the C:\Sample directory. Click on the drop down arrow for the File of type: and scroll to the dBase choice and click on it.
*You should now see only those files with a *.dbf extension.*
3. Select **cer95.dbf**
4. Click on the **Paste** button to place the command in the Syntax Editor. The Syntax Editor will now become the active window and you will see the text
GET TRANSLATE
FILE='c:\sample\cer95.dbf'
/TYPE=DBF /MAP .
5. Click on the Run **O** button on the Toolbar.

Review the information in the Log output. Note that many of the variable names in dBase were longer than 8 characters and SPSS truncated them to 8 characters. Also note that two of the variables were given SPSS variable names (**V27** and **V28**). The reason given in the log is that the original name was either a reserve word or a duplicate. For these instances you can see the name was a duplicate. The variable **maz_imp_qt** was truncated to **maz_imp_**. The variable **maz_imp_br** could not be truncated to **maz_imp_** because it already existed, so SPSS gave it the name **V26**. The variable **maz_imp_en** has the same problem and was given the name **V27**.

Now switch to the Data Editor to look at the data. Click on the Variables View at the bottom left hand of the screen. Many of the variables have a Label but some do not. Only those variables where the name was truncated received a Label. The Label is the original variable name. There are no value labels for any of the variables. Missing values are also not defined.

Now, click on the Data View tab. The first variable (**d_r**) was discussed earlier as the field used by dBase to indicate if a case has been marked for deletion. Any cases with an "*" in this field should be deleted from the file. To check to see if there are any cases, you can run a **Frequencies** on this field.

1. **Analyze /Descriptive Statistics**, select **Frequencies ...**
2. Select **d_r** from the list on the left and click on the **O** button.
3. Click on **Paste** to put the command into the Syntax Editor. Switch to the Data Editor active.
4. Execute the command by clicking on the **Run O** button located on the Toolbar.

If there were cases with an "*", we could use the **Select Cases** command to select only those cases where **d_r** contained a blank space.

1. From the Date Editor, select **Data /Select Cases**
2. Select the round button next to **If condition is satisfied**
3. Click on the **If...** button directly under **If condition is satisfied**
4. Select the variable **d_r** from the left hand box and move it to the right hand box.
5. Click in the box to the right of the variable name and type
= " "

6. Click on **Continue**
7. Select the round (radio) button next to Deleted (so that the cases that have not been selected will be deleted).
8. Paste the command
9. Select the text (highlight it) in the Syntax Editor from the line with `FILTER OFF` to the line with `EXECUTE` and run the command.

Syntax:

```
FILTER OFF.
USE ALL.
SELECT IF(d_r = " ").
EXECUTE .
```

When a file is read into SPSS that is not an SPSS for Windows file, the name of the file does not appear at the top of the screen in the title bar. Let's save the file in SPSS format.

1. Make sure the Data Editor is active
2. Click on **File /Save As...**
3. Filename `cer95`
4. Paste and run.

Now the file name appears in the title bar.

Once the cases that have been marked for deletion are removed, we can delete the variable `d_r` since it takes up space in the file and on your hard disk. There are two methods you can use to remove a variable - one is manually and the other is with syntax.

Manual method - From the Data Editor click once on the **variable name** to select the whole column and then press the "Delete" key on your keyboard or choose **Edit /Clear** from the menu.

Syntax method - Use the Merge Files command to merge the file back to itself and drop the variable.

1. **Data /Merge Files /Add Variables...**
2. Select file `cer95.sav` for the Read File
3. **Open**
All the variables from the second file will be dropped because the names are the same.
4. Select `d_r` from the list of variables in New Working Data File box and move it into the Excluded Variables: box
Note that no Key Variables are required because we are matching the same file to itself.
5. Click on **Paste**, click on **No** and run.

The syntax looks like:

```
MATCH FILES /FILE=*
/RENAME (d_r = d0)
/FILE='C:\sample\cer95.sav'
/RENAME (d_r date enqueteu mai_imp_ mai_loc_ mais_dem mais_det mais_imp
mais_pro marche mil_deta mil_gros mil_imp mil_prod mil_quan paddy quantite
region riz_imp riz_imp_ riz_loc riz_loc_ riz_quan sol_demi sol_deta sol_prod
```

```

sor_imp_sor_loc_sorgho_i v26 v27 = d1 d2 d3 d4 d5 d6 d7 d8 d9 d10 d11 d12
d13 d14 d15 d16 d17 d18 d19 d20 d21 d22 d23 d24 d25 d26 d27 d28 d29 d30 d31)
/DROP= d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 d10 d11 d12 d13 d14 d15 d16 d17 d18 d19
d20 d21 d22 d23 d24 d25 d26 d27 d28 d29 d30 d31.
EXECUTE.

```

Using the menu, the above `match file` command is built. We can simplify it by eliminating the `/RENAME` subcommand and adding the real variable name to the `/DROP` subcommand, e.g.

```

MATCH FILES /FILE=*
  /FILE='C:\sample\cer95.sav'
  /DROP= d_r.
EXECUTE.

```

Let's look at the **date** variable in the **Data Editor**. We see *********. The asterisks mean that the display width is not large enough to display the date. We can widen the display width by clicking on the line to the right between the variable names and dragging the line to the right. The format of the date variable is presented as **day-month-year**. The display format can be modified to display as other formats. Click on the **Variable View** tab at the bottom of the screen. Row 4 is the Date variable. Click in the column "Type" in row 4. Click on the gray box to the right. There are many date format types available.

SPSS10 has provided many date manipulation functions that were not available in the earliest versions of the program. We can compute a new variable that contains only the day, week, month or year information to facilitate reorganization of the data to correspond to varying analytical needs.

Converting Variables from String/Numeric Format to Consecutively Numbered Numeric Variables

String Variables

For specific types of analysis with SPSS (i.e., `ONEWAY`, `ANOVA`, `MANOVA` and `DISCRIMINANT`), string variables cannot be used as an independent grouping variable. The **Automatic Recode** command will create a numeric variable that groups the string values and uses the string value as the value label in the newly created numeric variable. Another reason you might want to use the **Automatic Recode** command is that the `TABLES` command will only display the first 8 characters of a string variable. If you wish to see the full information, you can use this command to create a new variable containing the strings as value labels and use the new variable in the `TABLES` command.

String values are recoded in alphabetical order. Uppercase letters precede lowercase counterparts.

Numeric Variables

The `MANOVA` analysis requires that the numeric variable's values are in consecutive order. We can use **Automatic Recode** to recode the values into consecutive order. Benefits to recoding categorical numeric variables into sequential order is that memory requirements are reduced and calculation performance is improved. Any values that do not have a defined value label in the old variable are given the old value as a value label in the new variable.

Missing values are recoded into new missing values that are higher than any non-missing values. For example, if the original variable has 10 non-missing values, the lowest missing value would be recoded to 11, and the value 11 would be a missing value for the new variable.

Exercise 3.1.

To recode string or numeric values into consecutive integers, from the menus choose

1. **Transform /Automatic Recode...**
2. Select a variable from the list on the left, i.e. **marche** (Market) and move it to the Variables->New Name box.
3. In the box to the right of the grayed out **New name** button type the name of the new variable that you wish to create. Then click on **New name**. The name will move up into the Variables->New Name box.
*The default order is “ascending”. If you wish the order to be “descending”, click on the radio button next to **Highest value** in the Recode Starting from area.*
4. Paste and run the command.

Switch to the Viewer and look at the log file. A list of old value, new value, and label are shown. As you scroll down through the list, you can see some of the disadvantages to typing a string. Scroll down to the “T”s.

TABACOUNDA	55	TABACOUNDA
TAMABCOUNDA	56	TAMABCOUNDA
TAMBA	57	TAMBA
TAMBACOUNDA	58	TAMBACOUNDA

It’s possible that these 4 values are all the same market. If you run a frequencies on the new variable, any values that are less than 10 are possible typographical errors.

If you wish to save the syntax file, switch to the Syntax Editor and save the file to **sect3.sps** and close the file.

Section 4

Introduction to Time Series, Data Cleaning & Verification

Data used in this SPSS for Windows session was taken from weekly time series data gathered by the Market Information System (SIM) in Mali. Multiple markets were chosen to represent various participants in the market such as producer, wholesale and consumer (or retailer). Information for many different grains are available in this data set. For the purpose of this tutorial, we will focus on sorghum.

Questionnaire Tables	SPSS for Windows Data File
Price quantity data for the 1989 to 1995 seasons	PQ_8995.SAV
Price data for the 1982 to 1989 seasons	P_8289.SAV
Spreadsheet file containing price index data for 1995	INDEX.WK1
Spreadsheet file containing production data for growth rates	MALIPROD.WK1

This tutorial assumes you have an understanding of basic statistics, i.e.

R², measure of the goodness-of-fit of a linear model;

F, test to see how well the regression fits the data (Student's **t** test and distribution is the square root of **F** - allows to test linear relationship),

ANOVA variability and linear relationship, etc.

The Price and Quantity Data File

Let's look price and quantity data file PQ_8995.SAV.

1. From the **File** menu, select **Open/ Data...**
2. Change to the c:\sample directory where we can find the sample data files and select PQ_8995.SAV.
3. Click on the **Paste** button to place the command into the Syntax Editor.
The Syntax Editor will now become the active window and you will see the text

```
GET
FILE='C:\SAMPLE\PQ_8995.SAV'.
```
4. Click on the Run button **O** on the Toolbar.

The price and quantity data file for the years 1989 to 1995 is now the working file.

As in Section 1 of this tutorial, it is important for you to know about the data file you are working with and the variables it contains. There are three ways to learn about the file:

1. The **Utilities / Variables** menu choice to look at individual variables.
2. The **Utilities / File Info** menu choice to display the information about the whole file at once in the Viewer
3. The Variables Tab at the bottom of Data Editor displays the variable information in a table format.

Let's use the first method:

1. From the **Utilities** menu select **Variables...**
2. Select a variable name. The information about that variable will appear to the right.

On the left of the **Variables** dialog box a list of variables are displayed, e.g. **loc, mar, cer, annee** (location, market, cereal, year). If you select a specific variable, the display to the right variable name, the variable label, the type of variable and its display width (plus decimals), missing values and type of measurement. Below this information the value labels appear if it is a categorical variable. Look at each of variables so that you become familiar with the type of information that is in this file.

Select the variable **p4c** - the calculated mean consumer price. You will notice in the display on the right, a range of missing values from 997 to 999. These values are not prices but represent codes for reasons that explain any missing price data! The value labels for these values are the reasons. It is important to know why data are missing. Again, these codes are not prices. To exclude these values when analyses are performed, they have been defined as "User Missing". SPSS for Windows does not include "User Missing Values" in an analysis unless the user specifically requests in the command that they be used. (This option is not available for all commands.)

If you wish to have all this information written to your Output window for later examination, you may do the following:

Pull down the **Utilities** menu and select **File Info**.

This command execute immediately. The **Viewer** becomes the active window and you see text output that contains a listing of all the variables with their definitions. This information about your file provides you with an easy method to document your data file. As a reminder, it can be copied into a word processor and edited to add more information.

Checking for Data Errors

Before you can begin to perform any analysis on a set of data, you should always check the data for possible errors. Errors can be introduced during data entry where values are typed incorrectly or the enumerator may incorrectly record a value while collecting data at the markets or in the field. Some of these errors can be discovered by checking for discrepancies between market prices, .e.g, wholesale price is higher than retail price. Cleaning exercises help us to verify the data as best we can.

Ideally, there should be data in all the variables of all cases. There may be questions that do not apply to all cases, and therefore, will not have any data. In the best of circumstances, the values for the variables that are not pertinent to the case are filled with a code that is specified as "User Missing". The analyst knows absolutely then why data are missing. If the value in the variable is "System Missing" (i.e., a period for numeric variables or a blank for string variables), the analyst does not know if the data entry person forgot to type the data or the data have not yet been received or exactly why the data are not there. Generally, we reserve "System Missing" for those variables where the data are truly missing - no one knows why there isn't any data.

Let's look at the data in this file carefully. Select the file and take a few moments to view the data. You can scan through the data or you can run **Frequencies** on categorical variables and **Descriptives** on continuous variables. You can also use **CrossTabs** to look at two or more categorical variables in relation to each other and **Means** to look at continuous variables controlled for by categorical variables to find problems.

FREQUENCIES

```
VARIABLES=loc mar cer annee mois date sem  
/ORDER= ANALYSIS .
```

There are 27483 cases in this file.

DESCRIPTIVES

```
VARIABLES=p4c p4p p5c p5p q6 pr8 pr10 pga13 qga14 pgv16  
/STATISTICS=MEAN STDDEV MIN MAX .
```

In the output from the **Descriptives** command, you can see that the number of cases for each variable is much less than the total number of cases. Let's look at the data. You can see many "System Missing" values (a period). Is it a problem?

There is an explanation for the "System Missing" values. This file is a combination of many different types of markets. Only certain data are collected in the different types of markets. Producer markets collect production data, wholesale markets collect wholesale data. However, we still do have to be assured that all the "System Missing" values are valid.

For those who are not familiar with French, we have provided a translation to help you understand the data.

French	English	Variable
<i>Prix à la production</i>	producer price - price paid by merchants or assemblers to the producers in a given market	p4p
<i>Prix au regroupement</i>	assembly (wholesale) price - price received by a merchant for a certain commodity (assembly level data)	pr8
<i>Prix à la consommation</i>	consumer or retail price - price paid by consumers to retail merchants (usually per kilogram*)	p4c

*Local units of measurement may vary in size from market to market. The data in this set have already been standardized to a metric unit. You have completed such an exercise in Section 2 already so we will not do the conversions here. In Mali, the enumerators do this conversions in the field before submitting the data through radio transmissions to shorten the length of time required to transmit the data.

Logic of market prices between producer, wholesaler and retailer

For this exercise, we will verify the logic of market prices for sorghum. First select the cases where the cereal is sorghum. To do this, we can use the **Select Cases** command. You should already have the file

PQ_8995.SAV as the working data file in the Data Editor. If you are not in the Data Editor, click on the  button on the Toolbar.

1. From the **Data** menu choose **Select Cases**
2. Select the radio button next to **If condition is satisfied**
3. Click on **If...** under **If condition is satisfied**
4. Click on the variable **cer** in the left column, then on **O**
5. Click on the equal sign "=" and then click once on the value **8**.

The text in the box should look like: **CER = 8**

6. Click on **Continue**
7. Select the radio button next to Filtered
8. Paste, make the syntax window active and run the command.

Various market price logic errors may occur. To identify specific cleaning errors we must first create a new variable that will contain a certain value depending on the type of error. The value of this new variable is going to be 0. To do so:

1. Go to the Data Editor and from the **T**ransform menu select **C**ompute...
2. Under Target Variable: type the variable name **clean**
3. Click on **Type&Labels..**
4. In the Label: box, type the text "**Price logic error types**". Click on **Continue**
5. Type the value **0** in the Numeric Expression box
6. Paste, select and run the command

There are three sets of prices so there are three different errors to calculate:

- (1) the producer price is higher than the consumer price,
- (2) the producer price is higher than the wholesale price,
- (3) the wholesale price is higher than the consumer price.

For each of these errors we will assign a different value to the variable **clean** that was computed above.

To program this in SPSS, you must:





5. From the **T**ransform menu select **C**ompute...
6. For the Target Variable: the variable name **clean** should already appear. If it does not, type it in.
7. In the Numeric Expression box, replace the 0 with **1**
8. Click on **If...**
9. Select the radio button for Include if case satisfies condition:
10. Type the statement **p4p > 0 & p4c > 0 & p4p > p4c**
What we are saying here is that producer price has to be greater than 0 and consumer price has to be greater than 0 and that producer price has to be greater than consumer price.
11. Click on **Continue**
12. Paste the command.
13. You see a QUESTION box asking if you want to **C**hange existing variable? We do, so click on **OK**. **DO NOT** run the command yet.
14. Repeat steps 1, and 5-10 replacing the previous information with the following information:

Numeric Expression value	If... Statement
2	p4p > 0 & pr8 > 0 & p4p > pr8 producer price greater than 0 wholesale price greater than 0 producer price greater than wholesale price

3	<p>p4c > 0 & pr8 > 0 & pr8 > p4c consumer price greater than 0 wholesale price greater than 0 wholesale price greater than consumer price</p>
---	--

If you prefer to not use the menus, you can copy and paste the syntax from the first “IF” statement and edit the contents.

We want to add Value Labels to define the values that are assigned to the variable **clean** so we know what they mean. One method is to

15. Switch to the Data Editor and click on the Variable View tab.
16. Click in the cell in the column labeled “Values” for the variable **clean**.
17. Click on the Ellipses box  to open the Value Labels dialog box.
18. In the Value: box, type “1”
19. Press <Tab> once and type “**Producer price is higher than consumer price**”
20. Click on , or press the <Alt> and “A” keys on your keyboard
21. Repeat steps 15 through 17 using the following information:
Value: Value Label:
2 **Producer price is higher than the assembly price**
3 **Assembly price is higher than the consumer price**
22. Click on 
23. Click on 
24. Make the Syntax Editor active and select all of the IF statements and run.

If you wish to add the Value Labels command to your syntax file, you can type the commands directly into the syntax file. The command is:

```
Value Labels clean
  1 'Producer price is higher than consumer price'
  2 'Producer price is higher than assembly price'
  3 'Assembly price is higher than consumer price'.
```

Note that you must end the command with a “.”(period) .

Now let’s see if there are any errors in the data! To do this you can run the **Frequency** command on the **clean** variable.

1. Select **Analyze /Descriptive Statistics /Frequencies...**
2. Move the variable **clean** into the Variables: box
3. Paste and run

Look at the results: 8 errors were found! In all, there were no errors where the producer price was higher than the consumer price, 7 errors were found where the producer price is higher than the assembly price and 1 error was found where the assembly price is higher than the consumer price. To fix these errors we should go back to look at the questionnaires and also to talk with the enumerators.

There are many reasons for these errors; however, we cannot modify the data to make the cases fit our logical criteria. The recommended method to handle these types of errors, if they cannot be corrected, is to avoid

using these values in any further analysis. One way is to set up a filter which will exclude all values of the **clean** variable that are greater than 0. The procedure is the same as we used to select just sorghum for this cleaning analysis.

Now let's take a closer look at these errors. We will first select just the cases we want to see and then use the **Case summaries** command to view the prices, markets and the case number for each of the cleaning errors.

1. From the **Data** menu select **Select Cases**
2. Select the radio button next to **If condition is satisfied**
3. Click on **If...** under **If condition is satisfied**
You will notice that $cer = 8$ can still be found in the box.
4. Click to the right of the value **8**.
5. Click on **&** first and then on **clean** in the left column, then on **0**.
6. Click on the greater than sign ">" and then click once on the value **0**.
7. The text in the box should look like : **CER = 8 & CLEAN > 0**
8. Click on **Continue**
9. Select the radio button next to **Filtered**
10. Paste the command.

Now we will build the **Case Summaries** command.

11. Select **Reports** from the **Analyze** menu
12. Select **Case Summaries...**
13. Select **mar cer mois date p4p p4c pr8 clean** from the list on the left and click on the **0** next to **Variable(s)**:
(Hint: You can hold down the "Ctrl" key to select all the variables at the same time).
14. Check the box next to **Show case numbers**
15. Paste and run both the commands.

```
USE ALL.
COMPUTE filter_$=(cer = 8 & clean > 0).
VARIABLE LABEL filter_$ 'cer = 8 & clean > 0 (FILTER)'.
VALUE LABELS filter_$ 0 'Not Selected' 1 'Selected'.
FORMAT filter_$ (f1.0).
FILTER BY filter_$.
EXECUTE .
SUMMARIZE
  /TABLES=mar cer mois date p4p p4c pr8 clean
  /FORMAT=VALIDLIST CASENUM TOTAL LIMIT=100
  /TITLE='Case Summaries'
  /MISSING=VARIABLE
  /CELLS=COUNT .
```

Now you can see which errors were made, the prices, in which markets and the case numbers (see table below). You may notice the last two values for the consumer price (p4c) are missing. This is explained by the fact that the observations were made in Zangasso, market number 75, which is not a consumer market - and no prices were taken. This does not influence in any way the cleaning errors. These errors are based on observed prices. The problem with these two errors may also lie between **pr8** and **p4p**, the latter being greater than the former.

Case Summaries

	Case Number	MAR Marché	CER Céréale	MOIS Mois	DATE Jour	P4P Prix moyen calculé à la production	P4C Prix moyen calculé à la consommation	PR8 Prix au regroupement pondéré	CLEAN Price logic error types
1	25691	41 KOUTIALA	8 Sorgho	2 Février	18	40.00	48.00	39.70	2.00 'Producer price is higher than assembly p
2	25693	41 KOUTIALA	8 Sorgho	3 Mars	4	40.00	48.00	39.80	2.00 'Producer price is higher than assembly p
3	25705	41 KOUTIALA	8 Sorgho	5 Mai	27	48.00	57.00	47.60	2.00 'Producer price is higher than assembly p
4	25739	41 KOUTIALA	8 Sorgho	2 Février	10	38.00	41.00	43.40	3.00 'Assembly price is higher than consumer
5	25758	41 KOUTIALA	8 Sorgho	6 Juin	23	45.00	60.00	44.90	2.00 'Producer price is higher than assembly p
6	25775	41 KOUTIALA	8 Sorgho	10 Octobre	20	44.00	54.00	43.73	2.00 'Producer price is higher than assembly p
7	27020	75 ZANGASS	8 Sorgho	5 Mai	14	40.00	.	32.00	2.00 'Producer price is higher than assembly p
8	27086	75 ZANGASS	8 Sorgho	8 Août	19	40.00	.	33.20	2.00 'Producer price is higher than assembly p
Total	N	8	8	8	8	8	6	8	8

a. Limited to first 100 cases.

Exercise 4.1

Go through the same cleaning steps that we used to clean sorghum, but this time use maize as the crop. (Hint: Maize is commodity number 9). Use all cases and then set a filter to select maize. You will also need to compute a new **clean** variable for maize. Run the frequencies and list all the errors. You should get 14 errors in all: 9 errors with a value of 2 and 5 errors with a value of 3 in the **clean** variable..

Outliers

We could use the **Frequencies** command to view outliers (values that are extreme compared to the rest of the values), however, the command produces long lists which make it difficult to get a good a sense for the distribution of the data. Another way to explore the data for extraordinary prices or extreme values is to look for *outliers*, much like we did in the first section. The use of this descriptive statistic will help us find the largest and smallest values in the data. We can look at the distribution of the data using a stem and leaf plot.

Check to see if you have a filter on the data and, if so, replace the filter so that we are using only sorghum as the commodity for analysis. We will use the **Explore** menu command (SPSS command is EXAMINE) and specify the three price variables (**p4p**, **pr8** and **p4c**) using the following steps:

1. Check the filter to see that you have filtered for the commodity “sorghum” and replace it, if necessary.
2. Select **Analyze /Descriptive Statistics /Explore...**
3. Select **p4c** from the list on the left and click on the **O** button next to **Dependent List**.
4. In the lower left corner of the dialog box is a box called **Display**. Click on the radio button (circle) next to **Both**.
Selecting “Both” will allow us to specify statistics and plots.
5. Next, click on the **Statistics...** button.
This will bring up the Explore: Statistics dialog box.
6. Click once on the square next to **Outliers** to put a H in the box.
You will notice there is already an H in the box next to Descriptives.
7. Click on the **Continue** button.
You will return to the Explore dialog box.
8. Next click on the **Plots** button.
9. Click on the radio button **None** in the **Boxplots** box display.

10. Click on the **Continue** button.
11. Click on **Paste** to put the command in the Syntax Editor and run the command.
12. Repeat these steps for the producer (**p4p**) and wholesale (**pr8**) prices creating a separate EXAMINE command for each variable..

Switch to the **Viewer** to look at the results. You see descriptive statistics, the extreme values table of the five highest and five lowest values, and the stem and leaf distribution of sorghum prices. Looking at the extreme values, you can decide if there are problems. Note that the case number is listed as well so you can find the case by using the **Go to Case** choice from the **Data** menu. For **p4c** there was one outlier found: the value **775**. The next closest value is 150 FCFA. In all likelihood, the data entry person may have typed 7 twice. We can choose to ignore this value or you can check the questionnaire to verify the value. If it is really 775, then you must leave it as it is, otherwise you can replace it with the correct value using syntax to change the value so that you have documentation as to how the case was changed. (Please refer to the SPSS Base User's Guide for further explanations of the **Explore** command.)

Extreme Values

			Case Number	Value
P4C Prix moyen calculé à la consommation	Highest	1	1057	775.00
		2	1175	150.00
		3	6453	150.00
		4	13293	150.00
		5	18562	. ^a
	Lowest	1	26994	.00
		2	26845	30.00
		3	26844	30.00
		4	25528	34.10
		5	26843	. ^b

- a. Only a partial list of cases with the value 150 are shown in the table of upper extremes.
- b. Only a partial list of cases with the value 35 are shown in the table of lower extremes.

If you wish to eliminate the extreme prices, you can use the **Missing Values** command to specify the values that should not be considered for analysis.

To specify a missing value:

1. In the **Data Editor**, click on the “**Variables View**” tab at the bottom left side of the screen.
2. Row 8 contains the variable **p4c**. Click on that row in the column “**Missing**”.
3. Click on the **...** box to bring up the **Missing Values** dialog box.
4. In the **Discrete Value** box, type 775.
5. Click on **OK**.

Another way you can specify missing values is to use the syntax command
`MISSING VALUES p4c (997 thru 999, 775).`

We would like to save this file to be used for further analysis later.

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

1. Make the Data Editor window active.
2. Use **Save As...** from the **File** menu
Verify that you are saving the file in the c:\sample subdirectory.
3. Name the file **sect4**
The extension .sav will automatically be added.
4. Paste and run the command.

We would like to save the syntax file as well.

1. Make the Syntax Editor window active.
2. Use **Save As...** from the **File** menu
Verify that you are saving the file in the c:\sample subdirectory.
3. Name the file **sect4**
The extension .sps will automatically be added.
4. Paste and run the command.

Exercise 4.2.

Repeat the exercise above to look at the extreme values for corn for consumer prices (**p4c**). You may also run a **Frequencies** command to compare it with sorghum. You will find one extreme value. Try entering a range of missing values. (Hint: look for a consumer prices equal 380 FCFA and set the missing values for 380 and above). Once you have defined the missing values, run the **Explore** command again. Notice the price average has changed! Although no outliers will be found, you may also run this exercise on producer and wholesale prices. Be sure to save your changes to the data file.

Section 5

Basic Analysis (Time Series)

Basic analysis will give us the results from which we can interpret the data simply and quickly. Using these commands, we can prepare graphs and tables which we can use to distribute market information rapidly and effectively. The purpose of this section is to teach you how to use these tools and to prepare reports.

The following exercises are some examples of how market information can be presented and prepared using basic analysis. Although there are many ways to present data and statistical analysis, the following exercises will produce

- a table and graph of weekly percentage changes
- aggregation to weekly and monthly levels
- graph of mean price
- graphs by agricultural season or production year, adjusting values for aggregated seasonal index

Percentage change from previous week

To be able to view percentage changes between weeks, we must create a new time series variable which will contain the difference in price from the previous case. Next the percentage difference is computed. These changes can be represented in a graph or a table.


The file we will use is the file created in the previous section; the file name is **sect4.sav**.

1. Open the file, **sect4.sav**.

We will work only with the commodity **sorghum**. Use the **Select Cases** command from the **Data** menu to select just this commodity (**cer = 8**).

2. Select just those cases where **cer = 8** so the rest of the cases are deleted.

We also want to be sure that the cases are in order by the key variables and the time series variables. To calculate the difference and percentage for consumer prices, follow these steps:

3. Sort the cases by **loc mar cer annee mois date** . (**Data / Sort Cases...**)
4. Create a new variable using the **Transform / Create Time Series...** command.
5. Select **p4c** from the list on the left and click on the  button next to the **New Variables(s)** box.



*Notice **p4c_1=DIFF(p4c 1)** appears in the box. The difference in prices between two consecutive cases will be calculated. The default **Function**: difference is defined as the nonseasonal difference between successive values. **Order**: defines the number of previous values to use to calculate the difference.*

6. Paste and run the command.




Switch to the **Data Editor** to look at the newly computed variable. Drag the new variable over so that the column is next to **p4c**. The first case is system missing. We can adjust the values so that the difference is moved up to the first week. To do the adjustment we use the same command, **Create Time Series**, but use the **LEAD** function to move the value up one case. Below is a definition of **Lead** and **Lag** so that you can see the difference.

Definition of **Lead**. Value of a subsequent case, based on the specified lead order. The order is the number of cases after the current case from which the value is obtained. The number of cases with the system-missing value at the end of the series is equal to the order value.

Definition of **Lag**. Value of a previous case, based on the specified lag order. The order is the number of cases prior to the current case from which the value is obtained. The number of cases with the system-missing value at the beginning of the series is equal to the order value.

7. Click on **Transform / Create Time Series...**
8. Change the function. Click the down arrow next to the **Function** dialog box where the word **Difference** appears. Choose **Lead**.
9. If the variable we created above **p4c_1** appears in the **New Variable(s):** box, return it to the column on the left.
10. Then select **p4c_1** from the list on the left and click on the  button next to **New Variables(s):** box to move it back into the box.
11. Change the name of the variable to **p4c_1_1** to **p4c_d** (for difference) and click on the  button.
12. Paste and run the command.

You can move this new variable **p4c_d** next to **p4c_1** to compare the values. Now we want to calculate the percentage change between the two weeks. To do this we compute a new variable:

1. From the **Transform** menu select **Compute...**
2. If there is any information showing that has been retained from previous commands, click on the  button to clear all the information.
3. For the **Target Variable:** type **pctchg**
4. Click on  and type a label to describe the variable (e.g. **Percentage change**).
5. Click on 
6. In the **Numeric Expression** box, type $(p4c_d / p4c) * 100$
7. Paste and run the command.

Now let's look at the percentage changes for three markets over four weeks. SPSS provides a second method to select specific data. The first one that we learned was the **Select Cases** command (to set a filter to select a specific set of data until the filter is removed or to select a specific set of data and in the process remove the rest of the data is are not selected).

We can use the **Temporary** command. This command indicates the beginning of a temporary set of transformations that will be in effect just for the next procedure. It will no longer be in effect with the execution of a second procedure that reads the data. Refer to the SPSS Syntax manual for further information. For example: we will use the **Temporary** command to select a subset of data to produce a table. Once the table command has been executed, the **temporary** command is no longer in effect and all the data will be available to be used for the next procedure. This command is not available through the menus so we must type it in the **Syntax Editor**. Switch to the **Syntax Editor** and type the command:

```
Temporary.
```

Now we want to select only weeks 20 through 23. We can type the command:

```
Select if (sem >= 20 and sem <= 23).
```

Then we wish to only use markets 3, 10 and 41. Type this command.

```
Select if (mar = 3 or mar = 10 or mar = 41).
```

We have now temporarily selected data which we will depict in a table using the **Tables** command. Once the **Tables** command has been executed, the **Temporary** command will no longer be in effect and the whole data set will be available. With the **Tables** command we can calculate various statistics and present them in a variety of ways that you can control. **Tables** gives you the capability to do the following:

- to choose how you want to assemble variables and statistics for display in rows, columns, layers, nestings, and/or concatenations.
- to manipulate table structure, content, and presentation format.
- to present multiple tables in the same display (concatenate) and to nest multiple sub-tables in any dimension.
- to include flexible percentages, specifying the base for the percentages (the denominator) so that the numbers add to 100% across rows, columns, sub-tables, or whole tables.
- to display up to 60 characters for variable labels and value labels.

To create the table using the **Basic Tables** command, go to:

1. **Analyze /Custom Tables /Basic Tables...**
2. Move **pctchg** to Summaries:
3. Move **mar** to Down:
4. Move **sem** to Across:
5. Click on **Layout...**
6. In the Summary Variable Labels box, click the radio button next to Across the top
7. **Continue**
8. Click on **Titles...**
9. In the Title box type:
Weekly price changes in percentages for three Malian markets over four weeks
10. **Continue**
11. Paste and run all of the commands from Temporary through Tables.

Look at the table. We can see the changes in prices from one week to another. See how the changes vary by market!

Weekly price change in percentages for three Malian markets over four weeks

		Semaine			
		20	21	22	23
		Percentage change	Percentage change	Percentage change	Percentage change
Marché	3 BADALA	0	-1	4	2
	10 DIBIDA	-1	2	1	2
	41 KOUTIALA	2	3	4	3

The table in the Viewer shows 2 decimal places. To show only whole numbers, double-click on the table to go into edit mode. Select the cells where you want the format to change, and right click. Select “Cell Properties” and change the Decimals to 0.

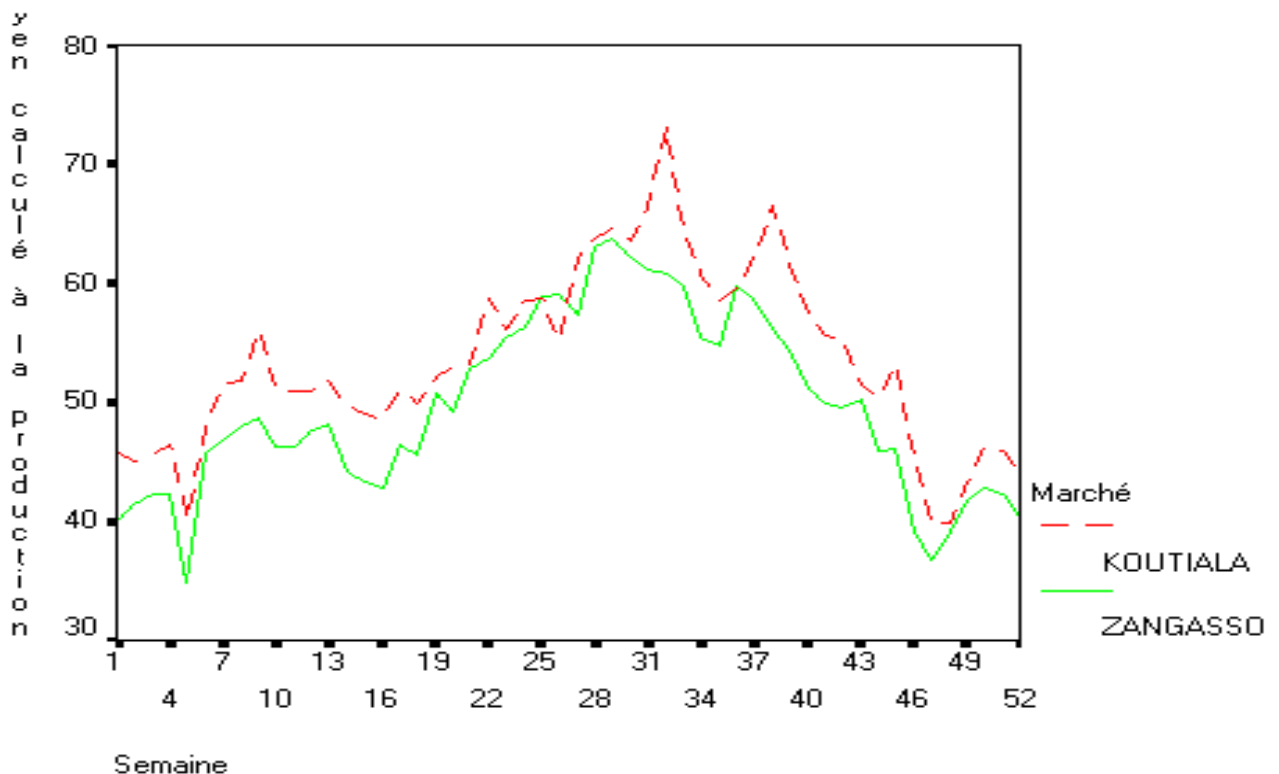
Let's now look at producer markets using a graph. We can select two different producer markets by typing in the Syntax Editor:

```
Temporary.
Select if mar = 41 or mar = 75.
```

Now plot the graph:

1. **Graphs /Line...**
2. Click on the square next to **Multiple**
3. Click on **Define**
4. Click in the radio button next to Other summary function in the Lines represents box.
5. Now choose the variable **p4p** - producer price - in the left column and click on the **O** next to Variable:
The mean value of producer prices will be graphed.
6. Choose **sem** for the Category Axis:
7. Choose **mar** for Define Lines by:
8. Click on **Options...**
9. Click on the radio box next to Display groups defined by missing values to uncheck the box.
This will eliminate the value on the graph representing the mean of all missing values
10. Click on **Continue**
11. Paste and run the command

Take a look at the graph: see how both markets follow the same trends but that sorghum prices are higher for Koutiala! Save your output in the Viewer using the filename **sect5.spo**



Exercise 5.1

Look at the consumer markets. To do so, you can select any of the consumer markets by typing

Temporary
Select if

commands as we did earlier. For this example, choose markets 3, 10, 20, 21, 51, 56, and 70 (do not run the command yet). Now copy and paste the graph command we built earlier, but replace the producer price variable with the consumer price variable **p4c**. Paste and then run both commands together. What do you see?

Observe that prices follow about the same trend over time. Also notice how all the curves are bunched together. One way to avoid this is to separate the curves using the **Split File** command:

1. Paste the Temporary and Select if commands from above.
2. Select **Data /Split File...**
3. Click the radio button next to **Compare groups**
*You may also choose **Organize output by groups** which will perform the same function as **Compare groups**. With graphs there is no difference in output. However, if a table is produced, **Compare groups** puts all the output in one table whereas **Organize output by groups** puts the output in a separate table for each group.*
4. Choose **mar** from the left column and put it in the **Groups based on:** box
5. Click on the radio button next to **File is already sorted**
6. Paste, but do not run the command

Now create the graph command. We want to view each graph separately so make sure to select the box next to **Simple**. Use the following steps:

7. **Graphs /Line...**
8. Click on the square next to **Simple**
9. Click on **Define**
10. Click in the radio button next to **Other summary function** in the **Lines represents** box.
11. Now choose the **p4c** consumer price variable in the left column and click on the **O** next to **Variable:**
The mean value of producer prices will be graphed.
12. Choose **sem** for the **Category Axis:**
13. Click on **Options...**
14. Click on the radio box next to **Display groups defined by missing values**
This will eliminate the value on the graph representing the mean of all missing values
15. Click on **Continue**
16. Paste the command

Go to the Syntax Editor and look at the commands. Because we used the Temporary command before the **Split File** command, this command is in effect for just the graph command. The next procedure that is executed would not be affected by the **Split File** command. If you specify this command without Temporary, it will remain in effect until we turn the command off. To turn the **Split File** command off, we can type **SPLIT FILE OFF** . on a new line following the end of the procedure we specified in the Syntax Editor or we can select the command from the menu. To select from the menu - click on **Data /Split File** and click on the radio button next to **Analyze all cases, do not create groups** and paste it to the Syntax Editor.

Now highlight with your mouse the commands to run - select **Temporary, Select if** through the **GRAPH** command. Run the commands and look at the new charts produced. See how each individual sorghum market has its own graph.

Aggregation to monthly levels and graphing mean price

Another way to represent data or group data for analysis is to aggregate data to another level. For example, if we wanted to view the average price for Bamako, which is composed of many markets, we can look at the prices by “localité”. This variable groups markets together by location. We can obtain an average value for each area. Let’s look at the Bamako average price:

1. Switch to the Syntax Editor and type:
Temporary.
Select if loc = 3.
2. Then create, paste and run the command for a simple line graph for consumer prices by week.

Note: For those of you who are comfortable with manipulating the syntax, you can copy and paste a previous graph command here, directly within the Syntax Editor. You may always do so at any time in future sections of the sample session. You might find this method will become easier as you become more familiar with the SPSS syntax. Remember to change the variables in the command and be careful to copy the correct graph command.

Exercise 5.2

1. Use the same **Temporary** and **Select if** commands as you used above. For the graph specifications, use month (**mois**) rather than week (**semaine**) as the Category Axis.
2. Practice the **Split File** command using localité (**loc**). Remember to type and run `SPLIT FILE OFF.` when you have finished using the **Split File** command (you may also paste this from the menu).
3. Look at other commodities (maize) as well. Save any graph that is of interest to you.
Should you forget to uncheck the missing values group while defining the graph, you will see a section of the command that includes `/Missing = Report.` This syntax is not necessary - we are not interested in groups defined by missing values at this time.

Adjusting values for aggregated seasonal index in order to graph by agricultural season or production year

The following sub-section demonstrates how to modify the presentation of the data by choosing November as the first month of the agricultural or production year rather than January as for a calendar year. To do this, we need to compute new variables for month and year adjusting for the agricultural season.

First, we should compute a variable that contains only the month and year together. This variable will identify each period of time represented in the graphs.

1. From the **Transform** menu select **Compute...**
2. For the **Target Variable:** type **date_m**
3. Click on **Type&Labels...** and type a label (e.g. **Month/Year**).
4. Click on **Continue**
5. Scroll down the choices in the **Functions:** box (or type a “D” and scroll) to the **DATE.MOYR(month,year)** function and click on the arrow to place it in the **Numeric Expression:** box.

6. The first ? is blocked, go to the list of variables and select **mois**. The question mark is replaced by this variable. In the Numeric Expression: box move over and block the second ? . Go back to the variable list and select **annee** to replace the second question mark.
7. Paste the command
8. Switch to the Syntax Editor. The format of the new variable is numeric. If we want to see the data as month/year we need to define the format for this variable. The next command is not available from the menu so we need to type it. Type the following:
FORMATS **date_m** (MOYR8) .
9. Run the compute and formats commands together.

Second, compute the season month and production year variables:

1. From the **Transform** menu select **Compute...**
2. Click on **Reset** to clear the previous compute.
3. For the **Target Variable:**
type **c_month**
4. Click on **Type&Labels...** and type a label (e.g. **Production Month**).
5. Click on **Continue**
6. In the Numeric **Expression:** box, type **0**
7. Paste the command
8. Return to the menu and select **Transform / Compute...**
9. In the Numeric **Expression:** box, type
mois + 2
10. Click on **If...**
11. Select the radio button for **Include if case satisfies condition:**
We are incrementing each month by 2 so we have to increment only those months less than or equal to 10.
12. In the **Expression** box type or paste **mois <= 10**
13. Click on **Continue** and paste the command.
We need to take care of months 11 and 12 next.
14. Repeat steps 7 through 13 replacing the previous information with the following:

Numeric Expression	If.. Statement
mois - 10	mois >= 11
15. Run all the commands.

Third, compute the production year variable:

16. Repeat steps 1 through 13. Use **c_year** as the variable name. Use the label “**Production year or agricultural seasons**”.
17. The criteria for the If statements are:

Numeric Expression	If.. Statement
annee + 1	c_month <= 2
annee	c_month >= 3

18. Run these commands.
19. Now we want to add Value Labels for each of the new months and production years. We can add the value labels by going to the Data Editor and typing in the values in the Variable View or we can use syntax in the Syntax Editor. The syntax is:

VALUE LABELS

```
/c_month 1 'NOV' 2 'DEC' 3 'JAN' 4 'FEB' 5 'MAR'
        6 'APR' 7 'MAY' 8 'JUNE' 9 'JUL' 10 'AUG' 11 'SEP' 12 'OCT'
/c_year 1982 '81-82' 1983 '82-83' 1984 '83-84' 1985 '84-85' 1986 '85-86'
        1987 '86-87' 1988 '87-88' 1989 '88-89' 1990 '89-90' 1991 '90-91'
        1992 '91-92' 1993 '92-93' 1994 '93-94' 1995 '94-95' 1996 '95-96'.
```

20. Let's format the variables as well so that on decimals are displayed.
Formats c_month c_year (F4.0).
21. Now run this command.
22. SAVE the data. IT IS VERY IMPORTANT THAT YOU DO NOT FORGET TO SAVE THE DATA FILE BECAUSE IT WILL BE USED IN THE SECTIONS AHEAD! Use the data filename **sect5.sav**.
23. Paste the save command and run it from the Syntax Editor.
24. Save your syntax file as **sect5.sps** and close it.

Exercise 5.3

Let's use the newly computed variables to look at the information based on an agricultural season from November to October. Graph the consumer price (**p4c**) by production month (i.e. **c_month**) and compare this graph with the graph that used calendar months. See how two months were adjusted to include these two months from the previous year and give away the last two months to the following marketing year!

Section 6

Seasonal Analysis (Time Series)

Seasonal analysis of weekly, monthly and yearly historical price or quantity data series for food and agricultural commodities constitutes an important part of market analysis (see Goetz and Weber 1986, IDWP 29, Michigan State University, for further discussion). This analysis seeks to separate out seasonal or intra-seasonal variations in the data series, from which behavioral inferences can be drawn. If there is seasonality in the production and marketing of crops, we should expect that prices and quantities will vary inversely over a production or marketing year. We would predict that prices are lowest right around harvest and then increasing as the time moves away from harvest. This seasonality is one of the possible components of a price and may mask overall trends or cycles in data series. Identifying seasonality then enables the analyst to identify the other components, enhancing our ability to understand or predict prices.

Merging and aggregating files and data

We want to perform seasonal analysis on the Market Information System data from 1982 to 1995. First, we need to merge data from two different files: `pq_8995.sav` (now `sect5.sav`) for the 1989-1995 data and `p_8289.sav` for the 1982 - 1989 data. For the purpose of this section, we will select data from Bamako and use sorghum as the cereal.

1. Start a new syntax window, and open `sect5.sav`. Paste and run.
2. Verify all filters and split files functions are off.
3. Now, as in past sections, please select out data for Bamako and sorghum by using **Data /Select Cases...** and specify `loc = 3` and `cer = 8`. Make sure to click on the radio button next to Deleted in the Unselected cases are box and then paste and run the command.

We will use the **Aggregate** command to aggregate data to a new level, in this case the monthly mean for Bamako. To do so, use the following steps:

4. Go to **Data/Aggregate...**
5. From the left column, select `annee mois date_m c_month c_year loc` and put the variables in the Break Variable(s) box
6. Select the consumer price variable `p4c` and put in the Aggregate variable(s) box.
7. Click on **Name & Label...**, type "Mean consumer price" as the label, and change the name from `p4c_1` to `p4c`. **Continue**
8. Click on **File...** and change the name of the file to `AGGBK1.sav`
9. **Save**
10. Paste and run the command
11. Now open the data file `p_8289.sav` using **File /Open /Data...**
12. Run a **Frequencies** on the variables `loc` and `cer` to verify that the only cases in this file are for Bamako and sorghum.

We do not need to aggregate the data in this file because it is already at the monthly level, so we can simply save this data in its present format. To do this, you must:
13. Go to **File /Save As...**
14. Give it the file name `AGGBK2.sav` and paste the command
15. Before you run this command, look at the syntax for the `SAVE` command. Click on the SYNTAX HELP button (second button from the right) on the toolbar. Save command syntax looks like:

```

SAVE OUTFILE=file
  [/VERSION={3**}]
      {2 }
  [/KEEP={ALL } ] [/DROP=varlist]
      {varlist}
  [/RENAME=(old varlist=new varlist)...]
  [/MAP] [/{COMPRESSED } ]
      {UNCOMPRESSED}
  [/UNSELECTED= [{RETAIN}]]
      {DELETE}]

```

Notice how the syntax differs here from the one pasted in the Syntax Editor? Although we cannot select many of the possible syntax functions from the menus for many of the commands, we can type the various options in the Syntax Editor. To do so, first:

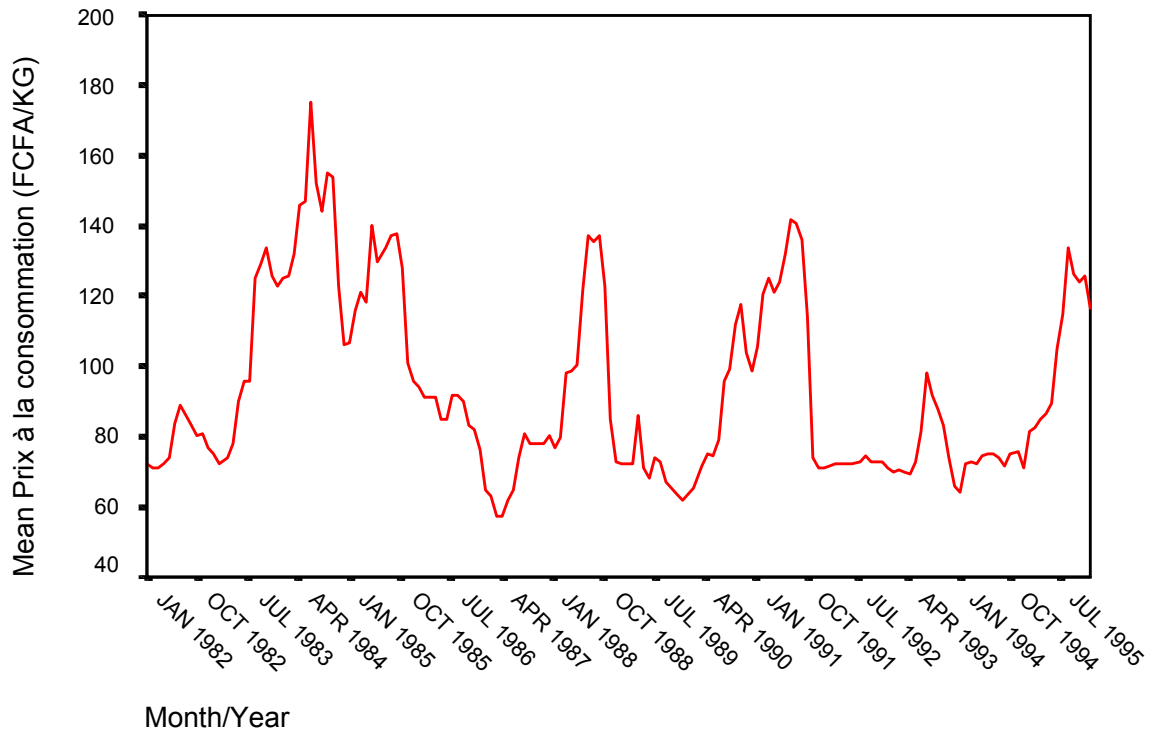
16. In the **Syntax Editor**, insert a line above the line with `/compressed`. Type the following on the line below `SAVE OUTFILE='C:\sample\aggbk2.sav'`:
`/KEEP annee mois date_m c_year c_month loc p4c`
*Rather than type the variable names, you can also use the **Utilities /Variables** dialog box to paste the variables into the **Syntax Editor**.*
17. Run the command.

Now that both files have the same variable names, we are ready to merge them together by adding the **cases** from the second file. By opening `aggbk1.sav` first, we can retain the production month and year labeling found in `Sect5.sav`.

1. Open the `aggbk1.sav` file
2. Go to **Data /Merge File /Add Cases...**
3. Choose `aggbk2.sav` and click on **Open**
4. Paste and run the command
5. Sort the file by `c_year` and `c_month` to put the data in chronological order.
6. Save the file to `aggbk3.sav`. (Remember to paste your command.)
7. Run the command
8. Save your new syntax file to `Sect6.sps`

Open `aggbk3.sav` and look at the data. You will notice that price data for the years 1982 to 1989 are now appended to those of 1989 to 1995! Now let's take a look at the data by graphing the consumer prices over the whole period and then by marketing year.

- a) For the whole period, graph a simple curve for `p4c` by `date_m`. Go to the Viewer and see how the trends in the chart vary through time.



- b) Graph multiple production years together for **p4c** by **c_month** and by **c_year**. See how the curves vary by production year and compared between the years.
- c) For each individual production year, graph simple curves by **c_month** using the **Split file** command by **c_year**. Don't forget to type and run `SPLIT FILE OFF`. when done. Save the output for a), b) and c) as `sect6.spo` (If there are some items in the Viewer that you do not wish to save, you can eliminate these items by selecting them from the left column and pressing the <Delete> key).

Computing moving average and graphing nominal prices

A moving average helps with calculating the seasonal index. An observation will depend on some of the previous and subsequent values of the same variable. This means that if an individual observation is larger or smaller, the averaging procedure will bring that observation more in line with the other values in the series and fluctuations are thus eliminated. Removing these short term fluctuations in the time series allows the analyst to focus on important longer term patterns such as cycles and trends.

To compute the moving average we need to use the **Create Time Series** command using the following steps. (Make sure that the `SPLIT FILE OFF` command has been run.) This command creates a new series as a function of an existing series, in this case, prices.

1. Create a new variable using **Transform /Create Time Series...**
2. We need to use the Moving average function. You can change the function by clicking on the arrow facing down next to the **F**unction drop-down box with the word **Difference**. Choose **Centered moving average** (see explanation below).
3. Then just below, you must type 12 in the **S**pan: box (representing the 12 months)
4. Select **p4c** in the column on the left and click on the button next to **N**ew Variables(s).

5. Change the name of the variable from **p4c_1** to **p4c_ma** (for moving average) and click on the **Change** button.
6. Paste and run the command.
7. You may also give the variable a label by typing in the syntax:

```
VAR LAB p4c_ma 'Centered moving average price'.
```
8. Run the variable label syntax.

Centered moving average is an average of a span of series values surrounding and including the current value. The span is the number of series values used to compute the average. If the span is even, the moving average is computed by averaging each pair of uncentered means. The number of cases with the system-missing value at the beginning and at the end of the series for a span of n is equal to $n/2$ for even span values and for odd span values. For example, if the span is 5, the number of cases with the system-missing value at the beginning and at the end of the series is 2.

The above command produces information in the Viewer about the new variable that has been computed:

Result Variable	Missing Values Replaced	First Non-Miss	Last Non-Miss	Valid Cases	Creating Function
P4C_MA		7	162	156	MA (P4C, 12, 12)

Let's graph the nominal prices with the moving average.

1. **Graphs/Line...**
2. Click on the square next to **Multiple**
3. Choose the radio button next to **Values of individual cases.**
4. Click on **Define**
5. Put **p4c** and **p4c_ma** in the **Lines represent** box.
6. Click on the radio button next **Variable:** in the **Category Labels** box
7. Select **c_month** or **date_m** and put it as the choice of the variable in the **Category Labels** box.
8. Paste and run the command

Now view the graph. Look at how the moving average presents the overall trend and eliminates the short term fluctuations. You can change the style of the lines to demonstrate more clearly the differences

Seasonal index

Seasonal movements in price and quantity time series are particularly prevalent in agricultural crops. Prices tend to go lower when the quantities increase with harvest and then gradually increase through the agricultural production season, as the stocks are used and storage costs are included in the prices. Seasonal price movement can effect non-storable commodities as well. There may be seasonal aspects to demand as well. For example, the price of poultry increases around holidays when festive meals are prepared. A seasonal index can be calculated using the moving average, i.e. the nominal price is divided by the moving average. To calculate the seasonal index, use the following steps:

1. From the **Transform** menu select **Compute...**
2. For the **Target Variable:** type **season**
3. Click on **Type&Labels...** and type a label (e.g. **Seasonal index**).

Now is a good time to save the data file as well. It is always good to give a file a name which helps us define the type of data with which we are working.

1. Save the data file with the name **season.sav**
2. Paste and run the command.

Aggregating seasonal index by month to create and graph historical monthly average index and standard deviation

As explained above, you may wish to aggregate data to other levels for analysis. In this sub-section, we will aggregate data that contains multiple years - months to a file that contains only one case per month to look at the distribution of the standard deviation.

First let's aggregate by production month using the following steps:

1. Go to **Data/Aggregate...**
2. From the left column, select **c_month** and put it in the **Break Variable(s)** box
3. Select the variable **season** and put in the **Aggregate variable(s)** box, then give a label such as "**Seasonal index**"
4. Select the variable **season** a second time and put in the **Aggregate variable(s)** box. Click on **season_2 = MEAN (season)** within this box and give it the label "**Seasonal index SD**"
5. Click on **Continue**
6. With this variable still highlighted, click on **Function...** and select the radio button next to **Standard deviation**
7. **Continue**
8. Click on **File...** and change the name of the file to **AGGSEAS.sav**
9. **Save**
10. Paste and run the command

Now open the file **aggseas.sav** (don't forget to paste the command and then run it from the Syntax Editor) and look at the data. You will see a value each of the twelve months. To see the trend of the average seasonal index (**season_1**) over a production year, graph a simple curve by **c_month** for Values of individual cases. The standard deviations will indicate how much variability there is in each month from year to year.

To graph the aggregated seasonal index with the standard deviations we must first compute variables. Use the **Compute** command so that you obtain the following syntax statements:

```
COMPUTE SDPLUS = season_1 + season_2 .  
EXECUTE .  
COMPUTE SDMINUS = season_1 - season_2 .  
EXECUTE .
```

Paste and run the commands. Save the new data to **aggseas2.sav**, paste and run this command. To graph the curves, use multiple lines representing **season_1**, **sdplus** and **sdminus**. Graph by **c_month** for Summaries of separate variables. This graph shows that in October and November there is greater variability than in other months, variability generated by annual differences.

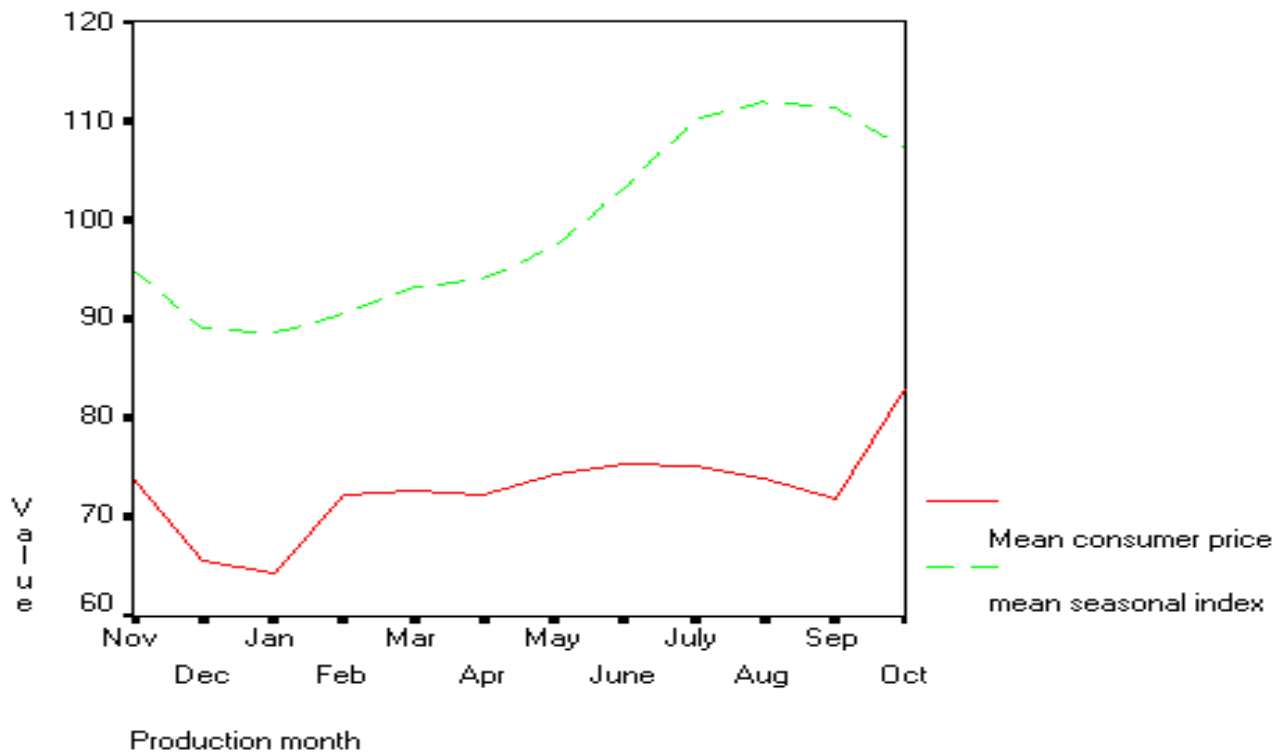
Leaving standard deviations aside, it is of interest to graph the aggregated seasonal index with nominal prices for a particular agricultural season or production year. Let's take the 1994 production year for this example.

1. Make the seasonal data active by opening the **season.sav** file (paste and run the command). Select the data for **c_year = 1994** (deleting all other cases) and then merge the data from the aggregated seasonal index with the standard deviation file, using the **Merge** command.
2. Go to **Data /Merge File /Add Variables...**
3. Select the filename **aggseas2.sav** and click on
4. Select **annee mois date_m loc season season_2 sadminus sdplus** from the list under **New Working Data File:** and click on
5. Check the box next to **Match cases on key variables in sorted files**
6. Click on radio button next to **External file is keyed table**
7. Select **c_month** from the **Excluded Variables:** list
8. Click on next to **Key Variables:** (bottom, right)
9. Paste the command
10. Click on , then
11. Select and run the command. Be sure to include EXECUTE.

Now let's graph the aggregated seasonal index with nominal prices. Try to graph the curves on your own but here are the steps if you are having trouble:

1. **Graphs /Line...**
2. Click on the square next to **Multiple**
3. Choose the radio button next to **Values of individual cases**
4. Click on
5. Put **p4c** and **season_1** in the **Lines represent box.**
6. Select **c_month** and put it as the choice of the variable in the **Category Labels box**
7. Paste and run the command

Take a look at the graph. See how the seasonal index is much higher than the consumer price for the 1994 production year. We cannot really do any comparison here until we place both series on the same scale. Another interesting manipulation of data for seasonal analysis is being able to modify the current season's nominal prices by dividing the monthly price by the November price such that the resulting values are on the



same scale as aggregated seasonal index with base for November. Using such a base, the harvest price can capture farmer's price appreciation throughout the marketing year. This type of graph allows you to compare price movements for one year against a mean seasonal average (and standard deviations).

What are the consumer price and seasonal price values for 1994? If we look in the Data Editor we can see the values for the month of November ($c_month = 1$) which are 73.85 and 94.91 respectively. To get the same base value or 100 in November we must divide all of the values throughout the production year by the November values. We can do this using the **Compute** command. Follow these steps to compute the indices:

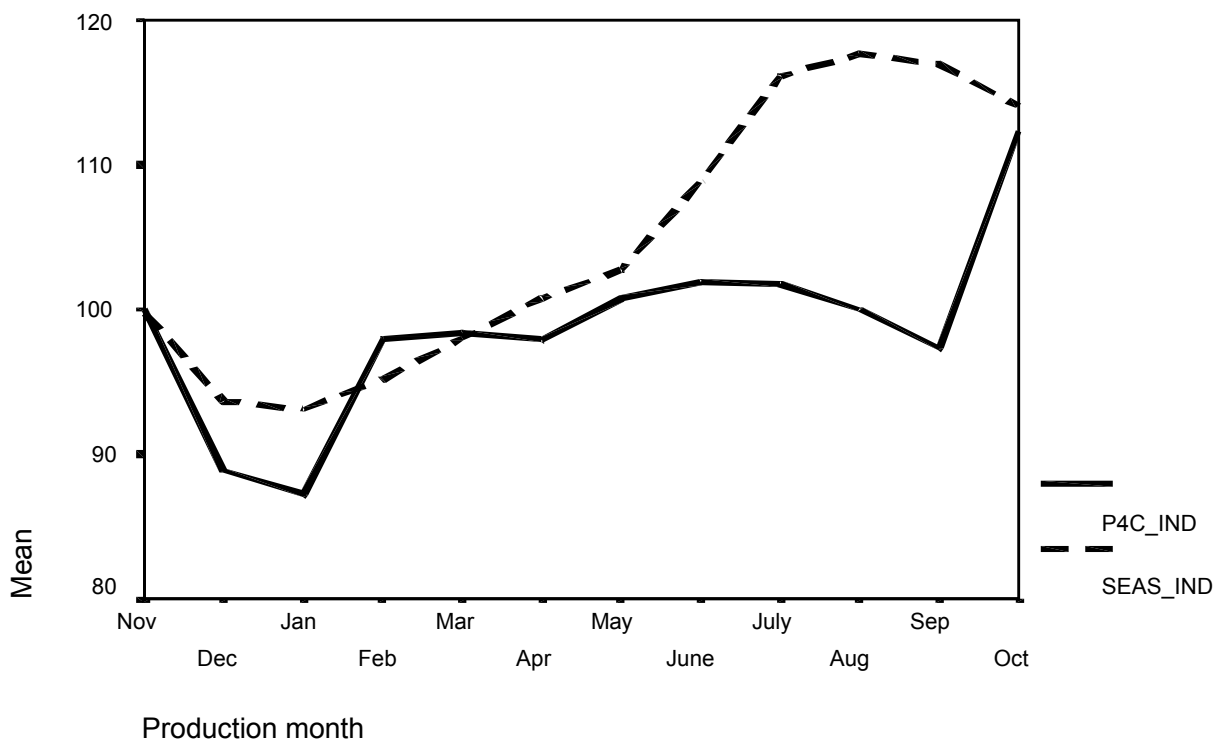
1. From the **Transform** menu select **Compute...**
2. For the **Target Variable:** type **p4c_ind** and type a label if you choose
3. In the Numeric **Expression** box, type **p4c/73.85*100**
4. Paste the command.
5. Return to the menu **Transform /Compute...**
6. For the **Target Variable:** type **seas_ind** and type a label if you choose
7. In the Numeric **Expression** box, type **season_1/94.91*100**
8. Paste the command and run both compute commands together.

We want to graph the aggregated seasonal index adjusted by month with adjusted nominal prices. Try to graph these curves on your own again but you may follow these steps if you encounter trouble:

1. **Graphs/Line...**
2. Click on the square next to Multiple

3. Click on the radio button next to Summaries of separate variables
4. Click on **Define**
5. Select **p4c_ind** and **seas_ind** from the left column and put them in the Lines represent box.
6. Now choose the **c_month** variable in the left column and click on the **O** next to Category Axis:
7. Click on **Options...**
8. Click on the radio box next to Display groups defined by missing values
9. **Continue**
10. Paste and run the command

Viewing the graph and the wide differences between the seasonally predicted values and the observed values, we can observe that 1994 was an odd year. Prices were, for one particular month, higher than the seasonal average but then remained pretty much around the same price. There was a major currency devaluation of the FCFA in 1994. There was initial fear of price increases in January (after the government in Mali had announced



ceiling prices). Once the traders were over the initial shock, the various actors realized that the harvest had been good, there were good stocks and prices reacted accordingly. To be able to better interpret the results, we should bring in production data, do formal interviews and so on. Write your ideas in the Viewer and compare with your colleagues. Don't forget to save your results (in sect6.spo).

Seasonal Decomposition (Optional)

Seasonal Decomposition estimates multiplicative or additive seasonal factors for time series. It creates new series containing:

seasonal adjustment factors (SAF) ,
the seasonally-adjusted series (SAS),
the trend-cycle components (STC), and
the error or random components (ERR).

Any series of monthly price or quantity observations over time can be decomposed into these four conceptual parts, each set of components being uniquely related to the actual observation.

Before we can do any seasonal decomposition analysis, we need to define dates in a certain way. To do this, we can use the **Define dates** command in the **Data** menu.

1. Make the **season.sav** data file active.
2. Go to **Data/Define dates...**
3. Select **Years, months** from the **Cases Are:** box
4. Type 1982 as the year for the **First case is:** box and leave 1 as the first month
5. Click on **OK**.

The command will be posted in the Viewer. To include in the syntax file, you will need to copy and paste from the output.

To run the seasonal decomposition analysis, we need to use the **Time Series** function.

6. Go to **Analyze /Time Series /Seasonal Decomposition...**
7. Choose the consumer price **p4c** in the left column and put it under **Variable(s):**
8. Leave **Multiplicative** in the **Model** as selected (explained below).
9. In the **Moving Average Weight** box, select the radio button next to **Endpoints weighted by .5** (see explanation of choice of models below).
10. Paste and run the command.

Types of models to choose from: Multiplicative or Additive.

Multiplicative Seasonal Adjustment.

Use this model when you believe that the seasonal component is a multiple of the original series. In this case, SPSS Trends estimates seasonal components that are proportional to the overall level of the series. Observations without seasonal variation have a seasonal component of 1.

Additive Seasonal Adjustment.

Use this model when you believe that the seasonal factor is simply additive and that the seasonal component does not depend on the level of the original series. In this case, observations without seasonal variation have a seasonal component of 0.

Look at the results in the Viewer and then in the Data Editor. The multiplicative seasonal model we chose is based on the following:

$$\text{PRICE} = \text{Trend} \times \text{Cyclical composition} \times \text{seasonal components} \times \text{error term.}$$

Example: for January of 1982 we have:
trend component of 80.049;
seasonal adjustment factor of 0.893 and
an original price of 72.

The modeled price is

$$\text{STC } (80.049) \times \text{SAF } (.893) = 71.49.$$

This compares to an observed price of 72.

The various seasonal variables computed by SPSS seem to correspond to the model. Be sure to save the changed data file: **season.sav**.

Graph the consumer price (**p4c**) with the seasonal adjusted series (**sas_1**) by **date_m** over the whole period, and then for each production year using the **Split file** command. See how the two curves vary from year to year!

Another interesting observation to make is to graph the mean SAF (seasonally adjusted factors for **p4c**) and the **p4c** indices. We can do so by dividing the monthly price by the November price for each year to obtain the same scale as the index with base 100 in November. To do this, we must first create a variable to represent the value for p4c. Call it **nov**.

1. Go to the menu **Transform/Compute...**
2. Type **nov** as the target variable.
3. In the Numeric Expression box, type **p4c**
4. Click on **If...**
5. Select the radio button for Include if case satisfies condition:
6. Type **mois = 11**
7. Click on **Continue**
8. Paste and run the command.
9. In the Syntax Editor, type
Temporary.
Select if **c_year** > 1982.
10. Now we will aggregate the data to year.
11. Go to **Data /Aggregate...**
12. From the left column, select **c_year** and put it in the **Break Variable(s)** box
13. Select the variable **nov** and put in the **Aggregate variable(s)** box.
14. Click on **File...** and change the name of the file to **AGGNOV.sav**
15. **Save**
16. Paste and run the command with the **Temporary** and **Select if** statement.

Now we need to merge the files so we can compute the indices and use the distributed values for November across their respective production years.

1. Go to **Data /Merge File /Add Variables...**
2. Select the filename **aggnov.sav** and click on **Open**
3. Check the box next to **Match cases on key variables in sorted files**
4. Click on radio button next to **External file is keyed table**
5. Select **c_year** from the **Excluded Variables:** list
6. Click on **O** next to **Key Variables:** (bottom, right)
7. Paste the command
8. Click on **OK**, then on **NO**
9. Select and run the command.

Now to compute the values for the indices, use the following steps :

10. From the **T**ransform menu select **C**ompute...
11. For the **T**arget Variable: type **p4c_ind** and a label of your choice
12. From the Numeric **E**xpression: box, type **p4c / nov_1*100**
13. Check to make sure that the IF statement says **I**nclude all cases.
14. Paste and run the command.
15. Repeat steps 1 through 4 with the target variable as **saf_ind** and the numeric expression as **saf_1*100**

Now we are ready to graph the seasonal adjusted factors with the consumer price indices. Use the **S**plit file command by production year and then graph as you have in past exercises, the variables **saf_ind** and **p4c_ind** by **c_month**. Notice how the curves vary from year to year and compare to each other. What interpretation can we make?

Save your syntax file (**Sect6.sps**) to add the syntax from the most recent exercises in the section. Close it so we can open a new one for the following section.

Exercise 6.1

The objective is to evaluate whether or not there is a seasonal component to the maize prices for Bamako, whether it is relatively stable over time, and what the best model might be. This calls for you to repeat all of the different types of seasonal analysis performed in this section. You must use maize (**cer** = 9) as the commodity for this exercise. Repeat each of the individual analysis listed below as best you can without referring to the section, but of course you may return to the section as often as needed. Always paste the commands to the syntax window and remember, you must type in some of the commands yourself. Do not forget to save your new data files, your syntax file for documentation and any graphs or output that you may choose for further analysis. Some of the data you will prepare in this exercise on maize will be used in exercises of sections ahead.

- i) First you need to aggregate and merge the same two data files **p_8289.sav** and **sect5.sav**. Select Bamako and maize such as was done in the section. Make sure you aggregate using the same variables as used above and breaking by the consumer price variable (**p4c**). Remember you will have to type part of the command (Hint: **/KEEP . . . / RENAME . . .**). When you have prepared the data, graph the consumer price by production year.
- ii) Compute the moving average for maize and then graph the consumer price against the moving average. (Hint: you will need to create the moving average using the **Transform/Create Time Series...** command)
- iii) Compute the season index and prepare a graph. (Hint: season index = nominal price/moving average).
- iv) Create a historical monthly average index and calculate the standard deviation using the aggregated seasonal index and then graph various curves (Hint: You will have to merge files, and then compute and use the values for the month of November...)
- i) Prepare seasonal calculations using the **define dates** and **seasonal decomposition** commands and then graph the mean seasonal adjusted factors against the price indices (Hint: You will need to use the **aggregate, merge** and **compute** commands).

Section 7

Trends (Time Series)

A time series may contain a natural or secular trend over time which may be increasing or decreasing. The trend factor is calculated by performing an ordinary least squares regression which will be presented in this session. It is useful to estimate the trend as a first step in determining the direction in which the series has been moving in the past, how it is likely to move in the future, and allow us to estimate future values. To reach these goals, we will use the **season.sav** file that was created in Section 6. As a reminder - this file contains a series of monthly consumer sorghum prices for Bamako from 1982 to 1995.

For the purpose of this section, we will focus on a shorter period of four years - from January 1992 to December 1995. We will select just those cases and save them to a new file for this analysis. The steps you will use are:

1. Open the **season.sav** file.
2. Select just the years we are interested in - 1992-1995 (use the option to delete unselected records).
3. Save the file to a new name - **trends.sav**.

We can study sorghum price trends in Bamako from 1992 to 1995 using a regression analysis. We will first use the **Curve Estimation** procedure::

4. **Analyze /Regression /Curve Estimation...**
5. Select the consumer price variable **p4c** in the left column and put it the Dependent(s): box
6. Click on the radio button next to Time in the Independent box
7. In the Models box, you will see Linear is already chosen.
Note that there are various models to choose from; you may also include a constant term, depending on the data.
8. At the bottom of the dialog box, check the box next to Display ANOVA table
9. Paste and run the command.

Among other things, the results in the **Viewer** provide us with the following information:

- S The adjusted R Square. This statistic gives us an empirical measure of the strength in the linear relation that can exist between the dependent and independent variables in the sample. It measures the proportion of the variation of the dependent variable that is explained by the regression. The coefficient has to be corrected by the degree of freedom to obtain the adjusted R Square coefficient that can be compared for the samples that have different sizes and different number of independent variables. In our example, the coefficient is adjusted **43%** (0.43064);
- S the coefficients of regression (B) that gives the linear relation between the dependent and the independent variables. In our example, you will see the constant term will be **61.58** FCFA and will increase by **0.84** FCFA per month;
- S the significance level (or associated probability) of testing the null hypothesis of the regression coefficients (SIG T) and of the overall regression (SIG F). These levels give the probability to accept or reject the fact that the regression coefficient is different from zero even if it is really equal to zero, in other words, there is no linear relationship between the dependent and the independent variables. The lower the probability, the better the regression will be. In our analysis, the probabilities are good for the individual regression coefficients just as for the overall regression, in the order of **0.0000**.

Dependent variable.. P4C

Method.. LINEAR

Listwise Deletion of Missing Data

Multiple R .66540
R Square .44276
Adjusted R Square .43064
Standard Error 13.38002

Analysis of Variance:

	DF	Sum of Squares	Mean Square
Regression	1	6543.2066	6543.2066
Residuals	46	8235.1530	179.0251

F = 36.54911 **Signif F = .0000**

----- Variables in the Equation -----

Variable	B	SE B	Beta	T	Sig T
Time	.842788	.139406	.665399	6.046	.0000
(Constant)	61.580932	3.923632		15.695	.0000

From the results of the regression analysis, we could conclude that consumer's prices of sorghum in Bamako have increased monthly in the order of 0.84 FCFA for the period 1992-1995. However, the least squares results are only valid under certain conditions (notably the absence of auto-correlation of the residuals, for example). With time series analysis, this condition is often not respected. When this condition is violated, the different coefficients described above have no statistical value. The Durbin-Watson test is a test for serially correlated (or auto-correlated) residuals. One of the assumptions of regression analysis is that the residuals for consecutive observations are uncorrelated. If this is true, the expected value of the Durbin-Watson statistic is 2. Values less than 2 indicate positive auto-correlation, a common problem in time-series data. Values greater than 2 indicate negative auto-correlation.

It is necessary to test the model in relation to the auto-correlation of the residuals. We've used the **Regression /Curve Estimation** command. Now we will use the **Regression /Linear** command. First, it is necessary to create the **time** variable (which will be used as the independent variable) since the **Linear** command does not include the option to regress over time. We will create it as a new variable as follows:

1. From the **Transform** menu select **Compute...**
2. For the **Target Variable:** type **time**
3. Add a label for this variable.
4. From the Numeric **E**xpression: box - type: (**annee** - 1992) * 12 + **mois**
5. Paste and run the command

We can now repeat the regression analysis but this time use the option to test the auto-correlation of the residuals:

1. Go to **Analysis /Regression /Linear...**
2. Put **p4c** in the Dependent: box
3. Put **time** in the Independent(s): box

4. **Statistics...**
5. Check the Durbin-Watson box in the Residuals box
6. **Continue**
7. Paste and run the commands.

Switch to the Viewer. You will notice that the coefficients are the same as those in the previous results and The Durbin-Watson (D.W.) coefficient is equal to **0.198**. When we compare this coefficient to the corresponding

Model Summary^b

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Durbin-Watson
1	.665 ^a	.443	.431	13.380	.198

- a. Predictors: (Constant), TIME Time
- b. Dependent Variable: P4C Prix à la consommation (FCFA/KG)

values in the table of the theoretical values of Durbin-Watson, we conclude a positive auto-correlation of the residuals. Thus, we must proceed to use the auto-correlation command to take into account the auto-correlation as follows:

1. Go to **Analysis /Time series /Autoregression...**
2. Put **p4c** in the Dependent: box
3. Put **time** in the Independent(s): box
4. In the Method box, select the Prais-Winsten method.
Note: you could have used the Cochrane-Orcutt method as it performs the same function but the first method is preferred.
5. Paste and run the command

The Viewer results reveal that the D.W. coefficient has increased to **1.37**, however it remains in the interval where the positive auto-correlation is considered. We can see a decrease in the adjusted coefficients of regression and of the significance levels of the null hypothesis tests for the regression coefficients.

Prais-Winsten Estimates

```
Multiple R          .33582303
R-Squared           .11277711
Adjusted R-Squared  .07334498
Standard Error      5.896952
Durbin-Watson       1.370633
```

In fact, data for long periods of time with many observations usually have more difficulties satisfying the theoretical values of D.W, which increase with the number of observations. Over a long period, we are generally confronted with trend reversals, which make the relation between the dependent and the independent variables problematic. Therefore, it is necessary with our data to subdivide the period on objective facts. For example, we could choose the devaluation that occurred in 1993 and consider a pre-devaluation period (1992-93) and a post-devaluation period (1994-95).

To perform the regression analysis for the 1992-93 period:

1. Use the Select If command to **filter** for data prior to 1994 (annee<1994).

2. Calculate the regression using the **Autoregression** command following the 5 steps used above.

To analyze the second period 1994-1995, we have to:

1. Change the **Select If** command to select the data ≥ 1994 .
2. Since the time variable for this period starts with 25, you must create a new variable `time_2`, which starts at 1.
3. From the **Transform** menu select **Compute...**
4. For the **Target Variable:** type `time_2` and add a variable label.
5. From the **Numeric Expression:** box type: `time - 24`
6. Select **If...** and click on the radio button next to Include if case satisfies condition:
7. Type: `annee > 1993` and press **Continue**
8. Paste and run the command
9. Run the regression analysis as was done previously but use `time_2` as the explained variable.

You will observe that the adjusted regression coefficient of the first period becomes negative (usually this is not possible since that coefficient varies from zero to one). The analysis procedure can produce a negative because the numbers are rounded up. Rounding up may cause the number to be greater than 1, even though theoretically it should only vary between 0 and 1. Since it is this number that is subtracted to one, it can happen that the adjusted coefficient is negative. We find this situation when the non-adjusted coefficient is close to zero. Moreover, in spite of an increase (doubling) of the coefficient using the auto-regression procedure compared to the “ordinary” method of the least squares (OLS), the D.W. coefficient still remains in the interval which makes us believe that a positive auto-correlation exists for the residuals. We also find that the trend becomes non-significant.

However, the second period that starts from the FCFA devaluation shows a price evolution without any auto-correlation with an adjusted regression coefficient of **35.8%**, and levels of significance for the regression coefficients of around **0.0000**.

The analysis that you have just run takes into account the trends of the observed prices and to a lesser extent, the cycle. It is important here to understand that cereal prices in the Sahel reflect more and more their availability in the context of market liberalization. This availability is closely related to drought cycles that can be more or less important. Moreover, taking into account the singleness of the harvest in a year, prices are subjected to seasonal variations. It would also be of interest here to run some seasonal analysis (see Section 6). We would need to “deseasonalize” **p4c** for example and use the new **sas_1** variable as the dependent variable in the **Autoregression** command.

You will see that with this data, all periods would benefit from a D.W. coefficient without auto-correlation of its residuals. However, for the 92-93 period, the trend shows that the increase in prices was non-significant and with a negative adjusted regression coefficient. We can also see a high determination coefficient of 57% for the post-devaluation period, against 12% for the 92-95 period. We also note that the increase in prices is also more important during the post-devaluation period: 2.64 F per month against 1.03 F per month for the 92-95 period.

Save your syntax file to **sect7.sps** and close it.

Exercise 7.1

Open the file with the maize price data you prepared in Exercise 6.1 in the last section. Then repeat the various trend analysis (through regression) you have just learned in Section 7. Interpret the results.

Section 8

Real Prices and Price Graphing and Tables (Time Series)

Real prices allow us to analyze data in real terms, taking into account inflation, meaning with the same base value across time. This important step in price analysis allows us to inflate/deflate current price data in order to bring all values to a common denominator. Although less important to farmers, economists prefer to work with real prices relative to the prices of other commodities in the economy by deflating prices. After such current time series have been deflated, they appear more unrealistic but if there was a significant trend in the data, it is good to know whether it is caused by inflation or by a natural trend in the real prices; hence, the deflating procedure can be used to answer this question. Using what we have learned in the previous section on trends, eventually it would be essential for, and effectual on, analysis to extrapolate the trend with linear projection to estimate the real price for the current month and establish a new monthly base. In this section, we will create real prices by importing price index data from 1988 to 1995 from a spreadsheet as was explained in Section 3.

Import consumer price index from a spreadsheet to compute real or deflated prices

We want to retrieve data from a spreadsheet file, the extension will be *.wk1 which is equivalent to *.w*. Follow these steps to import the price index data:

1. **File /Open...**
2. Change the Files of type: from SPSS(*.sav) to Lotus (*.w*)
3. Click on the file `index.wk1`
4. Click on paste **Paste**
5. An Opening File Options dialog box opens, just click on **OK**.
6. Run the command
7. Save the file as an SPSS data file; give it the name `pindex.sav`

We have just imported the spreadsheet into SPSS. You will see three columns: the first is the year, the second is the month and the third is the price index. We want to change the names of the variables. Manually, you can go to Variable View and type in the names. If you are developing a procedure that you want to run without any manual intervention, you can use code to make the name changes. We can merge the file to itself and change the names with the subcommand `Rename`.

1. Click on **Data /Merge Files /Add Variables...**
2. In the Excluded Variables: box, rename **a** to **annee**, **b** to **mois** and **c** to **pindex**.
3. Move all three variables back to the New Working Data File: box and move **a**, **b** and **c** from the New Working Data File: box to the Excluded Variables: box.
4. Then save the file to `pindex.sav`
5. Open the `season.sav` file. Select **File /Open /Data...**
6. Paste and run the command

We want to merge the `pindex.sav` file with this file, so we will match them by **annee** and **mois**. Remember - both files must be in the same sort order as the matching criteria.

7. Click on **Data /Merge Files /Add Variables**.
8. Check the box next to Match cases on key variables in sorted files
9. Click on radio button next to Both files provide cases
10. Select **annee** from the Excluded Variables: list
11. Click on **O** next to Key Variables: (bottom, right side of dialog box)
12. Repeat steps 10 and 11 for **mois**

13. Paste the command and run it.
14. Change the Variable Label for pindex to **Consumer price index**. Type in the Syntax Editor
Variable Labels pindex 'Consumer price index'.

Now that we have the price index we can calculate the real price by dividing the consumer price by the index:

1. From the **Transform** menu select **Compute...**
2. For the **Target Variable:** type **real** and add a variable label - **Real prices**
3. In the Numeric **Expression:** box, type **p4c / pindex * 100**
4. Paste and run the command
5. Then save the data file to **Real_prices.sav**, paste and run the command

Now lets see how the two prices compare by graphing the nominal and real prices together!

1. Since we are working with a price index on and after 1988, we must first select the data for the corresponding periods of time. As before, you must type:
Temporary.
Select if annee >=1988.
2. **Graphs/Line...**
3. Click on the square next to Multiple
4. Click on the radio button next to Summaries of separate variables
5. Click on **Define**
6. Select **p4c** and **real** from the left column and put them in **Lines represent**
7. Now choose the **c_month** variable in the left column and click on the **O** next to **Category Axis:**
8. Click on **Options...**
9. Click on the radio box next to **Display groups defined by missing values**
10. **Continue**
11. Paste and run the command
We will get a graph with average prices which includes prices before and after devaluation.
12. Repeat steps 1 through 11 but for **annee >= 1993** (average prices after devaluation)
13. Repeat step 12 for the years 1988 and up as well as 1993 and up but with **date_m** as the **Category Axis:** instead of **c_month**
This will show actual prices compared.

Now observe and compare all four graphs. Looking at the last graph, notice how the real prices differ from the consumer prices after devaluation on the CFA franc: see how nominal prices greatly increase while real prices only increase slightly and do not follow as much the same trend with nominal prices before the devaluation. When we compare the first two graphs, see how the average real prices are much lower and follow more or less the same trend if we exclude prices before devaluation!

Exercise 8.1

Use the same consumer price index to calculate real prices for maize for the same period. Chart the prices and then save your new maize price data and any of the charts produced.

Quantity and Price Graphing and Tables

This sub-section will show you how to manipulate quantity and price data together so we may analyze the variations and relations between the two. As was done in the Basic Analysis section, we will also go over presenting this data in graphs and tables.

First we need to make the price and quantity data file active and then we should select a different market which has quantities in the data e.g. Zangasso (market number 75). Since the data are only available after 1993, we should also select only these data accordingly.

1. **File /Open...**
2. Select **sect5.sav**, paste and run the command
3. From the **Data** menu, select **Select Cases...** and select the radio button next to **If condition is satisfied**
4. Click on **If...** under **If condition is satisfied**
5. Type the following in the box to the right: **mar = 75 and cer = 8 and annee >= 1993**
6. Click on **Continue**
7. Select the radio button next to **Deleted**
8. Paste and run the command.

We want to fit prices and quantities on the same graph. To do that we must first convert the quantity data from kilograms to tons. To convert the data into tons:

1. From the **Transform** menu select **Compute...**
2. For the **Target Variable**: type **q6_t** and add a label (e.g. **Quantities bought in tons**)
3. From the **Numeric Expression**: box, type **q6 / 1000**
4. Paste and run the command.
5. Save the data under a new name, e.g. **zan9395.sav**.

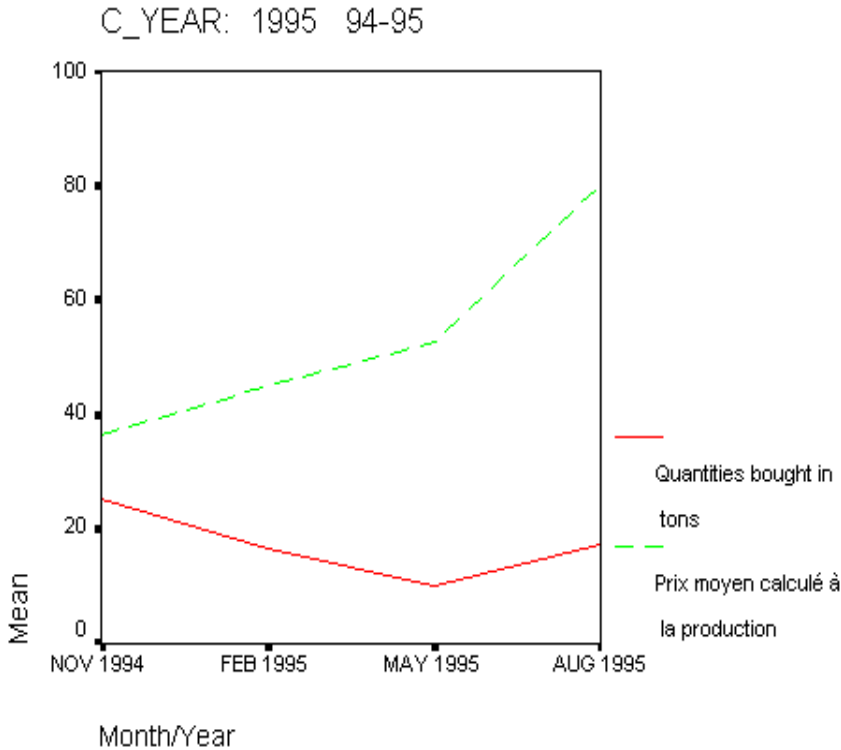
Now you are ready to plot a number of graphs to look at and analyze the data. If you have become familiar with copying and pasting the commands within the Syntax Editor, you may do so at any time. Let's try to graph the quantities against prices by quarter and then by production year:

1. Since we want to graph by quarter we must first select four months using the **Temporary** command. Type, but do not run, the following text in the Syntax Editor:
Temporary.
Select if c_month = 1 | c_month = 4 | c_month = 7 | c_month = 10.
2. **Graphs /Line...**
3. Click on the square next to **Multiple**
4. The radio button next to **Summaries of separate variables** should be selected
5. Click on **Define**
6. Select **q6_t** and **p4p** (the **producer price**) from the left column and put them in the **Lines** represent box.
7. Now choose the **date_m** variable in the left column and click on the **O** next to **Category Axis**:
8. Click on **Options...**
9. Click on the radio box next to **Display groups defined by missing values** **Continue**
10. Paste and run the command with the **Temporary** statement
11. Repeat steps 1 through 10 but use the **Split File** command to take a closer look at each production year. The syntax for this last statement should look like this:
Temporary.


```

Select if c_month = 1 or c_month=4 or c_month=7 or c_month=10.
SPLIT FILE by c_year.
GRAPH
  /LINE (MULTIPLE) =MEAN(p4p) MEAN(q6_t) BY date_m
  /MISSING=LISTWISE .

```



What can we say about this graph? The relationship between price and quantity is unclear but looking at each of the marketing years, it appears that as quantity decreases, prices increase. And we can also note a certain decrease in quantities during the summer/end of summer period (i.e. hivernage - the rainy season from June to September), the 93/94 season is a good example. But we can also observe increases in quantities at harvest. Save your charts from the Viewer to **Sect8.spo**. It has been some time since we saved the syntax file that contains all the commands we have developed so switch to the **Syntax editor** and save the syntax file to **sect8.sps**.

As we have done in earlier sections, let's prepare a graph and a basic table for presentation of

monthly price changes and for quantity and price data. For this example, we will choose the 1994 calendar year and we will look at monthly price changes. Since you are more familiar with the SPSS menus, we will just give you the basic instructions without the specific SPSS commands.

Make the **sect5.sav** file active, then create two **Select If** commands: one will select

```
annee = 1994 or c_year = 1994
```

and the second will select

```
cer = 8
```

Be sure you have selected the radio button next to Deleted. Then save the data under **TABLE94.sav**. (Always paste to your syntax file.)

The reason we are including **c_year** = 1994 in the select command is to include the month of December 1993 to calculate the monthly price change for January for the same market. If we did not do the selection this way, because of the structure of the data we would be calculating the change between December of a different market, which would be incorrect. The syntax should look like this:

```

GET
  FILE='C:\Sample\Sect5.sav'.
EXECUTE .
FILTER OFF.
USE ALL.
SELECT IF (annee = 1994 or c_year = 1994) .

```

```
EXECUTE .
SELECT IF(cer = 8).
EXECUTE .
SAVE OUTFILE='C:\Sample\Table94.sav'
/COMPRESSED.
```

After running the syntax, we want to aggregate prices and calculate the monthly price changes.

1. Now use the **Aggregate** command. Go to **Data /Aggregate...**
2. From the left column, select **loc mar cer annee mois** and put them in the **Break Variable(s)** box
3. Select the variable **p4c** and put in the **Aggregate variable(s)** box.
4. Give it a label such as “**Mean consumer price**” by clicking on **Name & Label...**
5. **Continue**
6. Click on **File...** and change the name of the file to **AGGTAB94.sav**. Click on **Save**
7. Paste and run the command
8. Now make the **aggtab94.sav** file active.
9. Create a new variable using the **Transform /Create Time Series...** command.
10. Select **p4c_1** from the list on the left and click on the **O** button next to **New Variables(s)** box. *You will notice $p4c_1=DIFF(p4c_1-1)$ appears in the box.*
11. Change the name of the variable from **p4c_1_1** to **p4c_p** (for price change) and click on the **Change** button.
12. Paste and run the command.
13. Give the new variable a variable label by typing the Variable Label command in the Syntax Editor.

```
Variable Label p4c_p 'Price change'.
```

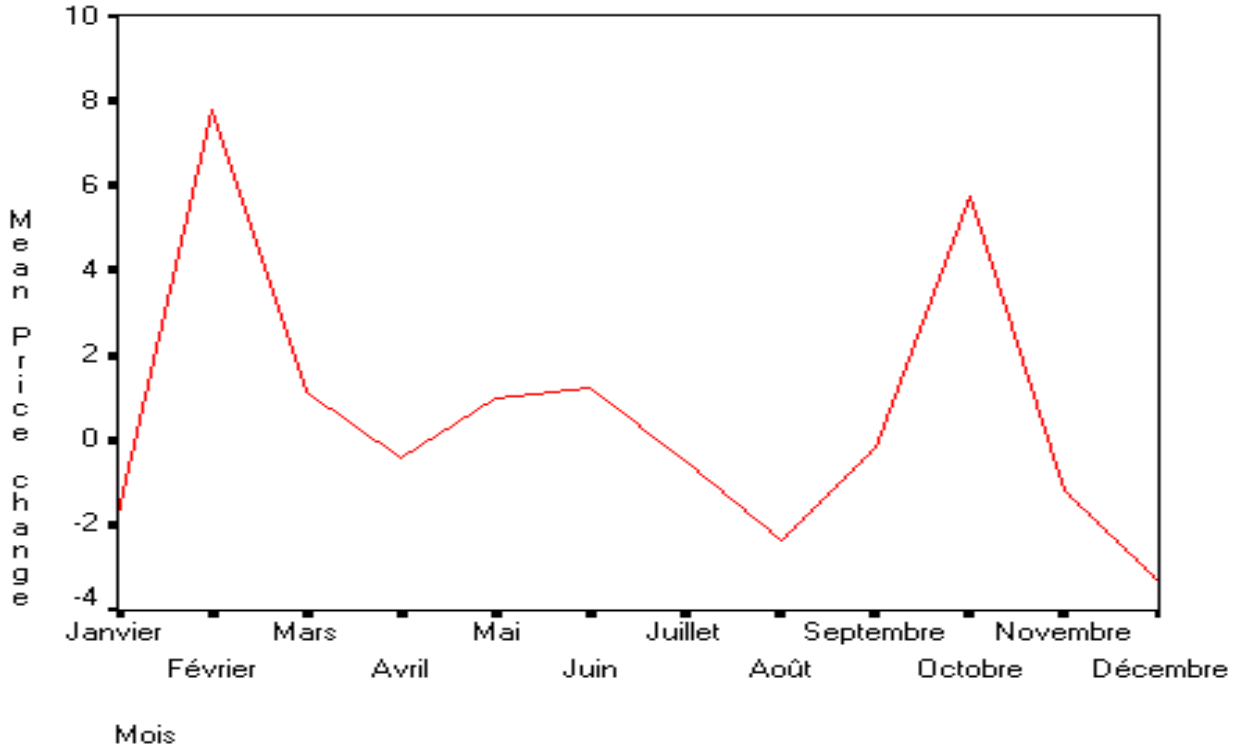
Now we are ready to graph the price changes for any market for any commodity during the 1994 year. For example, let's select sorghum for Bamako by typing in the Syntax Editor:

```
Temporary.
Select if mar = 3 and annee = 1994.
```

Now use the **Graph** command to create a simple curve to show the price changes **p4c_p** by month **mois**. The syntax should look like this:

```
GRAPH
/LINE(SIMPLE)=MEAN(p4c_p) BY mois.
```

Run both commands together. What do you see? Harvest is in October and November. Notice how the greatest variations start just before harvest (when supply is lowest) until February, a few months after harvest, and then remain in the same range over the rest of the year while increasing and then decreasing at the end of the production year just at harvest! Why is this? Write out your ideas in the Viewer and save it.



When you have finished practicing on different graphs, we will now prepare a table with prices for two months and percentage changes. Again we can choose any markets and commodities separately or together but for this example, we will look at sorghum for three different markets for the first two production months at harvest i.e. November and December. Type in the Syntax Editor:

```
Temporary.
Select if mar = 3 or mar = 10 or mar = 41.
Select if mois = 11 or mois = 12.
```

Using **Basic Tables**, prepare a table with prices for two months and percentage changes:

1. **Analyze /Custom Tables /Basic Tables...**
2. Move **p4c_1** to Summaries: and move **p4c_p** to Summaries:
3. Move **mar** to Down:
4. Move **cer** and then **mois** to Across:
5. Click on **Layout...**
6. In the Summary variable labels box, click the radial button next to Across the top
7. **Continue**
8. **Titles...**
9. In the Title box, type: **One month price change by market for sorghum**

10. Continue
11. Paste and run

Take a look at the table in the **Viewer**: see the price change for November and how the variable and results are displayed. This is an excellent exercise to prepare reports or market bulletins.

Exercise 8.2

Have fun modifying the output of the tables as you vary the display (properties, colors, format, widths and labels) for various markets for maize (**cer** = 9) and millet (**cer** = 7) for different months. Then try to graph various commodities for various markets using the **Temporary** command as we have done above: produce at least one graph on percentage changes for corn and another graph on quantities and prices for maize for each production year.

Growth Rates

A common task for agricultural economists and other analysts to do for market information systems is to calculate growth rates for production and other types of variables.

1. **File /Open /Data...**, for Files of type: select Lotus (*.w*)
2. Click on the file maliprod.wk1 and paste.
3. The Open File Options dialog box will appear, click OK.
4. Then run the command.
5. In the Data Editor, change the variable names to: **Campaign, year, productn** (tons) - the commodities are millet, sorghum, maize, rice and fonio.
6. Or - you can use the rename command to change the variable names. This command is not available from the menu system:
`RENAME (a b c = campaign year productn).`
7. Now let's calculate the growth rate using the **Curve Estimation** command. Go to **Analyze /Regression /Curve Estimation...**
8. Select the consumer price variable **productn** in the left column and put it the **Dependent(s):** box
9. Click on the radio button next to **Time** in the Independent box
10. In the **Models** box, you will see **Linear** is already chosen. Select the **Growth** and **Exponential** models as well.
11. At the bottom of the dialog box, check the radio box next to **Display ANOVA table**
12. Paste and run the command.

What do you see? The growth and exponential models give the same results, the difference is in the constant term. The time variable beta coefficient is the annual percentage growth rate. Save your syntax to **sect8.sps** and close the window.

```
Dependent variable.. PRODUCTN          Method.. LINEAR
```

```
Listwise Deletion of Missing Data
```

```
Multiple R          .89867
R Square            .80760
Adjusted R Square   .79386
Standard Error      252731.75158
```

```
Analysis of Variance:
```

	DF	Sum of Squares	Mean Square
Regression	1	3753492416174	3753492416174
Residuals	14	894226735608.9	63873338257.8

F = 58.76462 Signif F = .0000

----- Variables in the Equation -----

Variable	B	SE B	Beta	T	Sig T
Time	105069.898529	13706.30364	.898665	7.666	.0000
(Constant)	730969.050000	132533.6486		5.515	.0001

Dependent variable.. PRODUCTN Method.. GROWTH

Listwise Deletion of Missing Data

Multiple R .86226
R Square .74350
Adjusted R Square .72518
Standard Error .20501

Analysis of Variance:

	DF	Sum of Squares	Mean Square
Regression	1	1.7054863	1.7054863
Residuals	14	.5883807	.0420272

F = 40.58055 Signif F = .0000

----- Variables in the Equation -----

Variable	B	SE B	Beta	T	Sig T
Time	.070825	.011118	.862264	6.370	.0000
(Constant)	13.633042	.107506		126.812	.0000

Dependent variable.. PRODUCTN Method.. EXPONENT

Listwise Deletion of Missing Data

Multiple R .86226
R Square .74350
Adjusted R Square .72518
Standard Error .20501

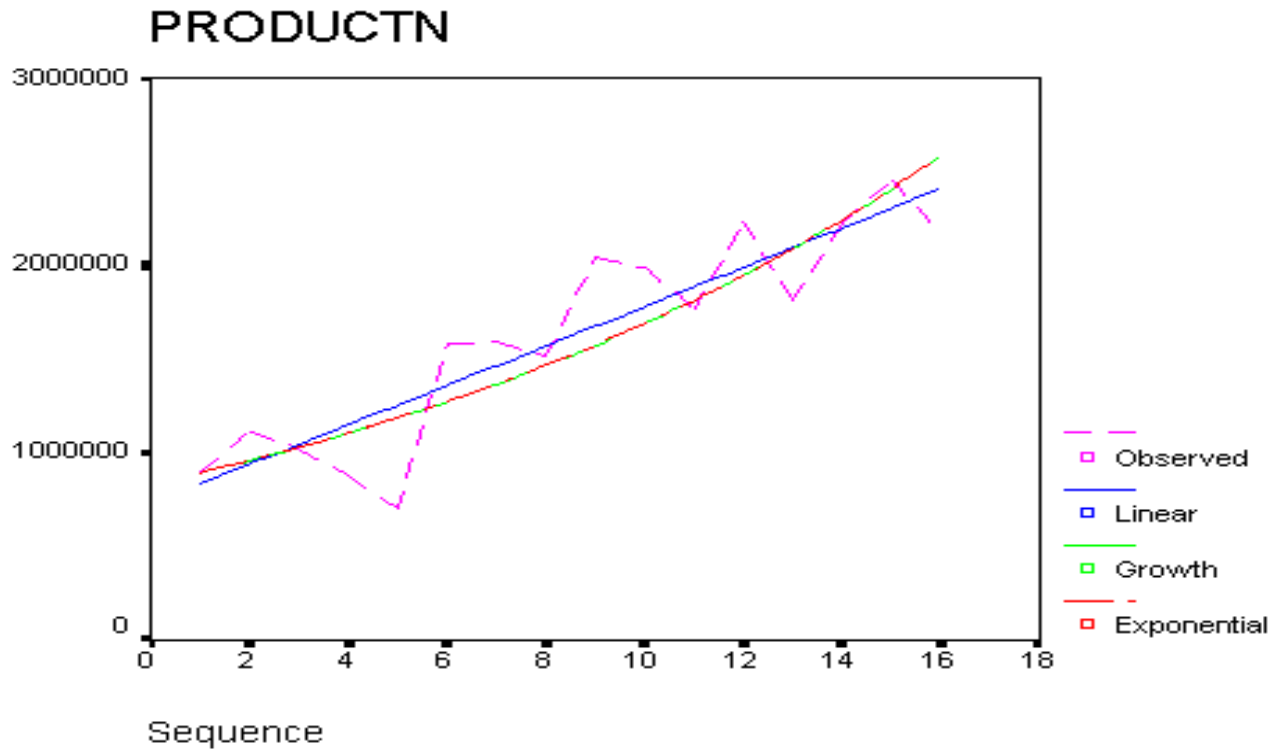
Analysis of Variance:

	DF	Sum of Squares	Mean Square
Regression	1	1.7054863	1.7054863
Residuals	14	.5883807	.0420272

F = 40.58055 Signif F = .0000

----- Variables in the Equation -----

Variable	B	SE B	Beta	T	Sig T
Time	.070825	.011118	.862264	6.370	.0000
(Constant)	833210.942117	89574.93202		9.302	.0000



Section 9

Margin Analysis (Time Series)

Marketed margins, in the simplest form, can be defined as the difference between prices paid for a commodity by consumers at the retail level and prices received by farmers when they sell their commodity to assemblers or other first handlers (see Goetz and Weber 1986, IDWP 29, Michigan State University for further discussion). The margins reflect the amount of services added to a commodity once it leaves the farm and is available to consumers in a retail outlet (i.e. spatial transformation, from the farm to the consumer, and form, paddy to rice). This gross consumer-producer margin will be disaggregated to more precise margins to show the different services added by the marketing system.

It is important to conceptualize the various dimensions in prices and visualize the marketing functions set of vertically related activities from the producer to the assembler, to the wholesaler, to the retailer and finally the consumer. These dimensions are: production and input costs, compensation for efforts and storage costs incurred, transfer and processing costs, risk and others. Marketed margins allow the analyst to assess the extent to which costs and profits are normal and/or excessive, relative efficiencies in carrying out services, comparison to added value, estimates of value of services and so on. Hence, we can gain insights into the sub-sector's performance. It is a good starting point for further analysis of marketing costs and it can help to explain the changes in prices at the different levels.

Before we calculate the margins for various markets, we need to merge producer market, assembly market, wholesale and consumer price data. We first need to choose one market for each stage of the marketing channel:

for the producer level we will use Zangasso (**loc**=57),
for the assembly level it will be Koutiala (**loc**=33),
for the wholesale/retail level we will use Bamako (**loc**=3).

As certain prices were only collected by the SIM in Mali starting in 1993, we will also select data from 1993 onward.

Data manipulation, aggregating and merging files for margin analysis

What you will do in this section is: for each of the markets mentioned above, prepare the data by selecting sorghum and the corresponding "localité" (representing numerous markets within a specific area) using a filter, and then aggregate it into a smaller file to be merged with the other smaller files representing each of the different markets. Hence, we will obtain our working file upon which we will perform the margin analysis. Lets work on the producer's market first.

PRODUCER'S MARKET

1. Open the **sect5.sav** file (paste and run the command)
2. From the **Data** menu select **Select Cases** and select the radio button next to If condition is satisfied
3. Click on **If...** under If condition is satisfied
4. Type **loc = 57 and cer = 8 and annee >= 1993**
5. Click on **Continue**
6. Select the radio button next to Filtered
7. Paste and run the command

8. Select **Data /Aggregate...**
9. From the left column, select **cer annee mois c_year c_month sem** and put them in the **Break Variable(s)** box
10. Select the variable **p4p** and put in the **Aggregate variable(s)** box.
11. Change the label to “**Average producer price for Zangasso**” and change the name to **p4p_zan** by clicking on **Name & Label...**
12. Click on **Continue**
13. Select the variable **pr8** and put in the **Aggregate variable(s)** box.
14. Change the label to “**Average assembly price for Zangasso**” and change the name to **pr8_zan** by clicking on **Name & Label...**
15. Click on **Continue**
16. Click on **File...** and change the name of the file to **MARGA.sav**
17. **Save**
18. Paste and run the command

ASSEMBLY WHOLESALE (gross price) MARKET

19. Change the filter by repeating steps 2 through 7 to select **loc = 33, cer = 8 and annee >=1993**
20. Repeat steps 8 and 9. Replace the variables in the **Aggregate variable(s)** box with **pr10**, giving it the name **pr10_kou** and the label “**Mean gross sale prices for Koutiala**”
21. Click on **File...** and change the name of the file to **MARGB.sav**
22. Click on **Save**
23. Paste and run the command

RETAIL (consumer price) MARKET

24. Change the filter by repeating steps 2 through 7 to select **loc = 3, cer =8 and annee >=1993**
25. Repeat steps 8 through 12 replacing the variable in the **Aggregate variable(s)** box by **p4c pga13 pgv16** giving them the respective names **p4c_bko ga13_bko gv16_bko** and respective labels “**Average consumer price for Bamako**”, “**Mean gross price at purchase**” and “**Mean gross price at sale**”
26. Click on **File...** and change the name of the file to **MARGC.sav**
27. Click on **Save**
28. Paste and run the command

MERGING FILES

1. **File /Open /Data...**
2. Select **marga.sav**, paste and run the command
3. Go to **Data /Merge File /Add Variables...**
4. Select the filename **margb.sav** and click on **Open**
5. Check the box next to **Match cases on key variables in sorted files**
6. Click on radio button next to **Both files provide cases**
7. Select **cer annee mois c_year c_month sem** from the **Excluded Variables: list**
8. Click on **O** next to **Key Variables: (bottom, right)**
9. Paste the command
10. Click on **OK**

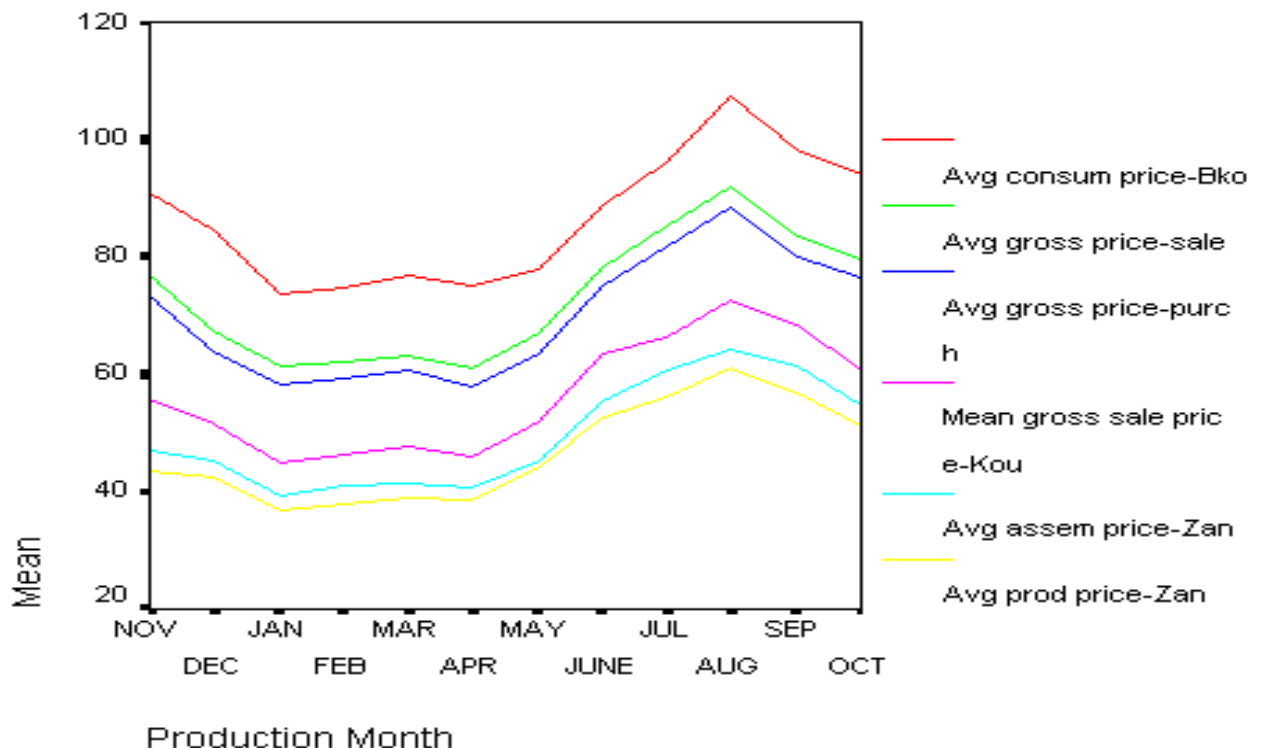
11. Repeat steps 3 to 10 selecting file `margc.sav`
12. Select and run both commands.
13. Save the new file as `margin.sav`

Graph to verify price logic

As it was explained in Section 4, data cleaning is fundamental and should be performed before any analysis. And as we had only verified certain prices such as the consumer, assembly and producer prices, we now need to verify the other prices we have introduced here (**pr10**, **ga13** and **gv16**). On this occasion, we will use a graphing technique to verify prices. Open the `margin.sav` file if you have not already done so.

1. Go to **Graphs /Line...** or copy/paste the syntax and replace the variables
2. Click on the square next to Multiple
3. The radio button next to Summaries of separate variables should be selected
4. Click on **Define**
5. Select **p4c_bko gv16_bko ga13_bko pr10_kou pr8_zan** and **p4p_zan** from the left column and put them in the Lines represent box.
6. Now choose the **c_month** variable in the left column and click on the **O** next to Category Axis:
7. Click on **Options...**
8. Uncheck the box next to Display groups defined by missing values and click on **Continue**
9. Paste and run the command

Switch to the Viewer to look at the graph. If any line crosses over another line, the price logic would not be respected e.g a producer price higher than an assembly price in Zangasso. We do not see any lines crossing, so



we may pursue further analysis. The graph also shows us the sequential logic of each price (see the change in

colors of each line and corresponding label). If we wanted, we could have also verified the prices more closely by looking at each production year using the **Split File** command.

Computing and graphing shares of the marketing channel (Optional)

Section 9 can be quite long so this section was chosen as an optional exercise. It looks at the various shares of the marketing channel up to consumer level.

The ratio of the producer price over the consumer price (or world/reference price) for a tradable good is the price share the producer receives for the sale of that good on the market. It can also indicate the possibility of increasing the producer price if there were no distortions present, e.g. political.

Hence, we can compute the share at each level by dividing each price by the consumer price in Bamako. Compute these shares using the following steps:

1. From the **Transform** menu select **Compute...**
2. For the **Target Variable:** type **prodshar** and a label (e.g. **Producer share**)
3. From the Numeric **E**xpression: box, type **p4p_zan / p4c_bko**
4. Paste and run the command
5. Repeat steps 1 to 4 replacing the target variable and numeric expressions by the following information (give each one an appropriate label):

Target Variable	Numeric Expression
ass1shar	$pr8_zan / p4c_bko$
ass2shar	$pr10_kou / p4c_bko$
wh1shar	$ga13_bko / p4c_bko$
wh2shar	$gv16_bko / p4c_bko$

Now lets graph the relative price ratios to get an idea of the variations across the year, and of the percentages for each level. Use the Graph command and make sure you type **wh2shar wh1shar ass2shar ass1shar** and **prodshar** from the left column in that order as you put them in the Lines represent box. Graph the shares by week (**sem**). This syntax should look like this:

```
GRAPH
  /LINE (MULTIPLE) =MEAN(wh2shar) MEAN(wh1shar) MEAN(ass2shar)
  MEAN(ass1shar) MEAN(prodshar)
  BY sem.
```

Computing and graphing gross margins between different marketing levels

To calculate the gross margins use the following instructions:

1. From the **Transform** menu select **Compute...**
2. For the **Target Variable:** type **marg1** and a label (e.g. **Collection**)
3. From the Numeric **E**xpression: box, type **pr8_zan - p4p_zan**
4. Paste and run the command

- Repeat steps 1 to 4 replacing the target variable, label and numeric expressions by the following information:

Target Variable	Label	Numeric Expression
marg2	Assembly	pr10_kou - pr8_zan
marg3	Rural Gross	ga13_bko - pr10_kou
marg4	Urban Gross	gv16_bko - ga13_bko
marg5	Retail	p4c_bko - gv16_bko

When you have computed all the margins, try to graph the margin data by week for all five margin variables together. What do you see? There are many variations between the lines which makes it difficult to comprehend. To get a better understanding of the data, a good analysis to run is descriptive statistics as we did in Section 1.

- From the **Analysis** menu select **Descriptive Statistics / Descriptives...**
- Select **marg1 marg2 marg3 marg4 marg5** from the list on the left and put them in the **Variable(s):** box
- Click on the **Paste** button and run the command

Looking at the results, we can see that the mean margins are highest for retail and gross rural and they are the lowest for the gross urban and collection margins. See how the margins vary: look at the minimums and maximums. How can we interpret these variations? Write down and save your ideas in the Viewer (Sect9.spo) and discuss them with your colleagues. Before we take a look at mean annual margins in various graphics, save the data (Sect9.sav) as we have calculated many variables and would like to keep them for further analysis.

Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation
MARG1 Collection	146	-8.00	12.60	3.0148	2.3457
MARG2 Assembly	134	-21.20	15.50	6.4396	4.0355
MARG3 Rural Gross	129	5.43	44.77	13.5557	4.8645
MARG4 Urban Gross	136	.27	7.03	3.1623	1.0570
MARG5 Retail	131	5.09	26.90	13.6574	4.4620
Valid N (listwise)	123				

Exercise 9.1

It can be of interest to lag prices to calculate lagged margins depending on the different market days for a given channel, and the time to get the product from farmer to assembler to wholesaler to consumer, which can take from days to weeks. Apply what you have learned in the first part of Section 5 to lag prices over two weeks. Then compute and graph lagged margins with what you have learned on gross margins. What do you observe?

Graphing pie and bar chart for market margins at different levels

Mean annual margins presented in pie.

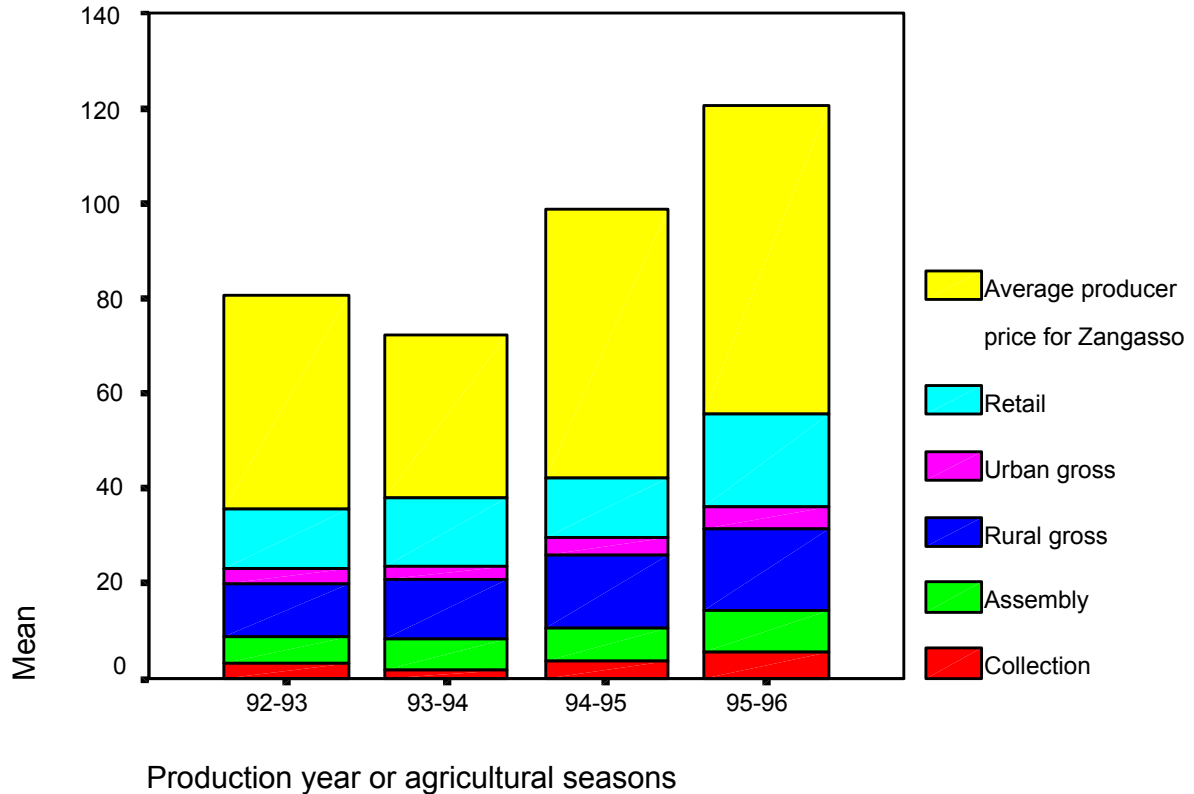
1. Go to **Graphs /Pie...**
2. The radio button next to Summaries of separate variables should be selected
3. Click on **Define**
4. Select **marg1 marg2 marg3 marg4 marg5** and **p4p_zan** from the left column and put them in the Slices represent box.
5. Select all of the variables and then click on **Change Summary**
6. Click on the radio box next to Mean of values
7. **Continue**
8. Paste and run the command

The pie gives us an overall idea of the distribution of the margins but let's take a closer look at the distribution annually using a bar chart and see if there are any changes.

Annual margins presented by year in bar charts.

1. Go to **Graphs /Bar...**
2. Click on the square next to Stacked
3. The radio button next to Summaries of separate variables should be selected
4. Click on **Define**
5. Select **marg1 marg2 marg3 marg4 marg5** and **p4p_zan** from the left column and put them in the Bars represent box.
6. Now choose the **c_year** variable in the left column and click on the **O** next to Category Axis:
7. Click on **Options...**
8. Click on the radio box next to Display groups defined by missing values **Continue**
9. Paste and run the command

See how the total mean margins decrease in 1994 and then increase in 1995. The largest absolute increase is at the producer level (**p4p_zan**) but we can see a high relative increase for the assembly level (**collection**) as well.



Aggregating data to monthly and yearly levels and calculating coefficients of variation in margins

Coefficients of variation are calculated by dividing the standard deviation by the mean. We need to build a different file so we may construct the data to allow us to compute the coefficients of variation. We are doing this in order to analyze the intra-annual variability in the different margins, which will give us an idea of which levels of the system are absorbing price changes. We will aggregate data to the **c_year** using the **sect9.sav** file. It should be the current file in memory. We will compute the mean and standard deviation for each of the gross margin variables plus the production price.

1. To aggregate, go to **Data /Aggregate...**
2. From the left column, select **c_year** and put it in the **Break Variable(s)** box
3. Select the variables **p4p_zan marg1 marg2 marg3 marg4 marg5** and put in the **Aggregate variable(s)** box.
4. Select the 6 variables again and put them in the **Aggregate variable(s)** box
5. Immediately click on **Function...**
6. *The 6 variables will still be selected in the **Aggregate variable(s)** box.* Click on the radio button next to **Standard deviation**
7. Click on **Continue**
8. Click on **File...** and change the name of the file to **MARGPLUS.sav**
9. **Save**
10. Paste and run the command

Open the `margplus.sav` file. Now we will compute the coefficient of variation:

11. From the **Transform** menu select **Compute...**
12. For the **Target Variable**: type `cv_p4p` (and add a label of your choice)
13. From the **Numeric Expression**: box, type `p4p_zs_2 / p4p_zs_1`
14. Paste and run the command
15. Repeat steps 11 to 14 replacing the target variable and numeric expression by the following information:

Target Variable	Numeric Expression
<code>cv_marg1</code>	<code>marg1_2 / marg1_1</code>
<code>cv_marg2</code>	<code>marg2_2 / marg2_1</code>
<code>cv_marg3</code>	<code>marg3_2 / marg3_1</code>
<code>cv_marg4</code>	<code>marg4_2 / marg4_1</code>
<code>cv_marg5</code>	<code>marg5_2 / marg5_1</code>

Using a simple graph, let's look at the mean coefficient of variations for the different margins for 1994 to 1995 (we could also use tables or other forms of presentation as well). Graph the six variables computed above by `c_year`. All the curves seem to follow the same trend except `cv_marg1` in 1993-94 which increased and has a value of 1.18! It would seem that this is due to an error (high standard deviation). In all it might look like rural wholesalers (who are active in `marg2` and `marg3`) are absorbing a lot of the variability in market prices.

Save this file (`margplus.sav`) so you may use it for further analysis.

Deflating margins by inflation rate

Refer to Section 8 for a brief explanation of deflating prices. Open the `sect9.sav` file (paste and run the command), then aggregate by `c_year` and `c_month` following these instructions:

1. Go to **Data /Aggregate...**
2. From the left column, select `c_year` and `c_month` and put them in **Break Variable(s)**
3. Select the variables `p4p_zs marg1 marg2 marg3 marg4 marg5` and put in the **Aggregate variable(s) box**.
4. Click on **File...** and change the name of the file to `MARGININD.SAV`
5. **Save**
6. Click on Paste and run the command

We want to bring in the consumer index information from the `pindex.sav` file into the `margind.sav` file, so we need to open the `margind.sav` file first.

7. Open `margind.sav`
8. Go to **Data /Merge File /Add Variables...**
9. Select the filename `pindex.sav` and click on **Open**
10. Check the box next to **Match cases on key variables in sorted files**
11. Click on radio button next to **External file is keyed table**

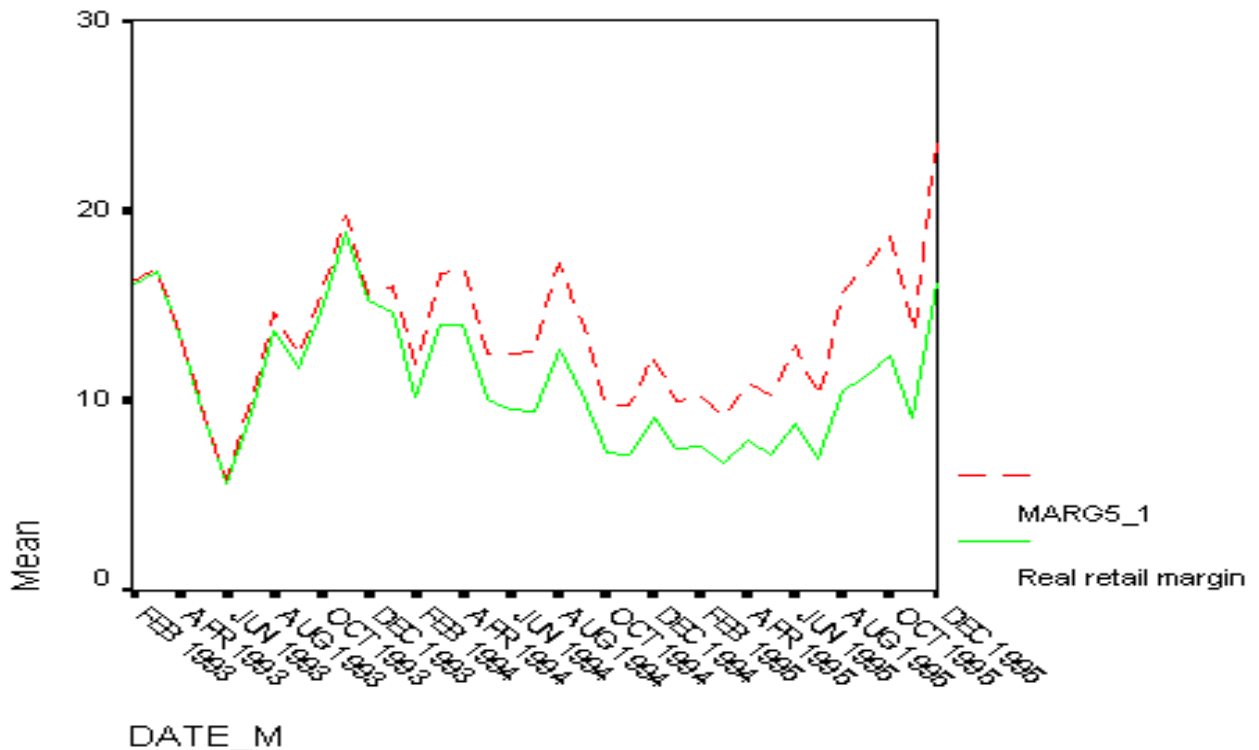
12. Select **annee** to **c_year** and click on **O** next to Key Variables: (bottom, right)
13. Paste the command, click on **OK** and run.

Now we can compute the real retail margin. As before, you need to use **Transform / Compute**. For the Target Variable: type **marg5_r** and a corresponding label, and in the Numeric Expression: box, type **marg5_1 / pindex*100**. Save this new data file under the **margreal.sav** filename.

Before we prepare a graph to view the real and nominal margins for presentation, it is nice to have a date variable with a combined time period of month and year. As we have done in the past, we can use the **DATE.MOYR(month,year)** function in the **Compute** command (if it is not already present in data):

1. From the **Transform** menu select **Compute...**
2. For the Target Variable: type **date_m**
3. Click on **Type&Labels...** and type a label (e.g. **Month/Year**).
4. Click on **Continue**
5. From the Functions: box, select the **DATE.MOYR(month,year)** function and put it in the Numeric Expression box.
6. Replace both ? of the function by **mois** and **annee**
7. Paste the command.
8. Switch to the Syntax Editor and type in the formats command to format the variables:
`Formats date_m (MOYR8).`

Now graph the nominal retail margin and the real retail margin by **date_m**. See how the two curves start to diverge at the beginning of 1994! The results show margins in real terms, taking into account the high inflation



rate after devaluation. If we had other price indexes, we could have done the other margins in real terms. It

would have been particularly interesting if we had wholesale price indexes but they are very hard to find in Africa.

Gross margins

In this sub-section we will show you how to assess the effect of consumer price indexes on margins, assess scale effect on marketing services and assess price-setting behavior.

Regressing Gross Marketing Margin by Consumer Price Index to Assess Effect of Index on Margins.

Since we are interested in how marketed margins behave over time, and as quantities put through the system for a particular sub-sector increase or decrease, marketing margins can be further analyzed using linear regression techniques. We can examine the relationship between a dependent and an independent variable such real margin and the price index. Regression also produces certain statistics (R^2 , F, ANOVA COEFF, and so on) depending upon the choice of regression.

We first need to compute a gross margin. Keep the same file open from the sub-section used above (`margreal.sav`). The gross margin can be calculated as follows:

1. From the **Transform** menu select **Compute...**
2. For the **Target Variable:** type **grossmarg** (and add a label of your choice)
3. In the **Numeric Expression:** box, type
marg1_1 + marg2_1 + marg3_1 + marg4_1 + marg5_1
4. Paste and run the command.

We want to discard the missing data for January of 1993.

5. Use the **Select If** command to select **grossmarg > 0**
6. Save the data file - `grossmarg.sav`.

To use a linear regression, follow these steps:

7. Go to **Analysis /Regression /Linear...**
8. Put **grossmarg** in the **Dependent:** box
9. Put **pindeX** in the **Independent(s):** box
10. Paste and run the command

If there is a one percent change in the inflation rate, then the gross margin increases or decreases by the value of the Beta coefficient. This is the inflationary effect on the gross margin. The results show a relatively good R Square which indicates a good fit and the index term is not significantly different from 0 (as significance for T and F tests are below 0.05).

Coefficients^a

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	14.436	6.401		2.255	.031
	PINDEX	.199	.050	.568	3.961	.000

a. Dependent Variable: GROSMARG Gross margin

Regressing Gross Marketing Margin by Quantity Marketed to Evaluate if Scale Effects in Marketing Services

We do not have enough data for sorghum to differentiate quantities by market destination. For this example, we will use rice as the commodity and selecting the Niono-Bamako marketing channel. We will use `sect5.sav`. Then select and aggregate rice for Bamako and Niono as follows:

1. Make `sect5.sav` file active.
2. From the **Data** menu select **Select Cases** and select the radio button next to If condition is satisfied
3. Click on **If...** under If condition is satisfied
4. Type `loc = 3` and `cer =4` and `annee >= 1993`
5. Click on **Continue**
6. Select the radio button next to **Filtered**
7. Paste and run the command.
8. Go to **Data /Aggregate...**
9. From the left column, select `annee mois` and `sem` and put them in the **Break Variable(s)** box
10. Select the variable `p4c` and put in the **Aggregate variable(s)** box.
11. Click on **File...** and change the name of the file to `REGRICE.SAV`
12. **Save**
13. Paste and run the command
14. Repeat steps 1 to 7 but replace `loc = 3` with `loc = 43` under If condition is satisfied
15. Repeat steps 8 to 10, replace aggregate variable `p4c` with `p4p` and `q6` and then click on the radio button next to **Replace working data file**
16. Paste, click **NO** and run the command
17. Go to **Data /Merge File /Add Variables...**
18. Select the filename `regrice.sav` and click on **Open**
19. Check the box next to **Match cases on key variables in sorted files**
20. Click on radio button next to **Both files provide cases**
21. Select `annee mois sem` from the **Excluded Variables:** list
22. Click on **O** next to **Key Variables:** (bottom, right)
23. Paste the command
24. Click on **OK**
25. Click on **NO** and then run the command.
26. Save the new data file to `sect9scal.sav`.

Now we need to compute the margin for rice.

1. Calculate this margin using the **Compute** command for the target variable `margrice` as you have done before
$$p4c_1 - p4p_1$$

Now we will run the linear regression:

2. Select the **Split File** command from **Data** and split by `annee`. Check to see that the **Compare groups** and the **File is already sorted** radio buttons are selected before pasting.
3. Go to **Analyze /Regression /Linear...**
4. Put `margrice` in the **Dependent:** box

5. Put **q6** in the Independent(s): box
6. Paste and run both the command

Looking at each of the three years, we can see that the results were not very good for 1994 or 1995, and are somewhat better for 1993 as the R square showed a slightly stronger fit (optimistic estimate of how well the model fits the data) and the only result that is significant. Marketing margins do increase slightly for the first two years with greater volume in the marketing channel (value of regression coefficient B).

Model Summary

Année	Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1993	1	.602 ^a	.363	.345	11.66633
1994	1	.432 ^a	.186	.166	13.89550
1995	1	.138 ^a	.019	-.005	17.17748

a. Predictors: (Constant), Q6_1

ANOVA^b

Année	Model		Sum of Squares	df	Mean Square	F	Sig.
1993	1	Regression	2864.462	1	2864.462	21.046	.000 ^a
		Residual	5035.819	37	136.103		
		Total	7900.282	38			
1994	1	Regression	1726.244	1	1726.244	8.940	.005 ^a
		Residual	7530.308	39	193.085		
		Total	9256.552	40			
1995	1	Regression	235.605	1	235.605	.798	.377 ^a
		Residual	12097.704	41	295.066		
		Total	12333.309	42			

a. Predictors: (Constant), Q6_1

b. Dependent Variable: MARGRICE

Regressing One Level Price by Another Level Price to Assess Price-setting Behavior

Do retail prices influence producer prices? Are retail prices dependent on producer prices? We can try to answer these questions using linear regression. Use the following steps to regress consumer and producer prices:

1. Go to **Analyze /Regression /Linear...**
2. Put **p4c** in the Dependent: box
3. Put **p4p** in the Independent(s): box
4. Paste and run the command

The results show a very good fit as the R square is fairly high and the results are very significant as well. The coefficient for the producer price (**p4p**) is significantly different from 1 so there is not constant markup in the

channel from the producer to the retail price. Retailers and those between, are not simply price takers. Retail prices change by more than one unit when the rural price changes.

Let's look at the producer/retail price relationship in the opposite way: does farm level price change in response to retail price changes?

1. Go to **Analyze /Regression /Linear...**
2. Put **p4p** in the Dependent: box
3. Put **p4c** in the Independent(s): box
4. Paste and run the command

It would seem so, the fit is fairly good and is again significant - and the value of β is high as well, close to 1. Save your syntax to **sect9.sps** and close it.

Exercise 9.2

This exercise will take a bit of time. You are to repeat all of the different types of margin analysis performed in this section. You must use maize (**cer** = 9) as the commodity for this exercise. Repeat each of the individual analysis that you performed for sorghum. Try to interpret the results - write down your ideas in the output file following the results of each analysis and compare these with your colleagues. Always paste the commands to the Syntax Editor and remember that you can type in some of the commands yourself. Do not forget to save your new data files, your Syntax file for documentation and any graphs or output.

- i) First, you need to aggregate and merge files for margin analysis. Use the same "localités" (loc) and time periods. Prices for maize are available at the various marketing levels.
- ii Graph to verify price logic and modify or recode if necessary.
- ii Calculate and graph the producer share as well as the gross margins between marketing levels. Practice using the pie and bar charts.
- iv Aggregate to monthly and yearly levels and calculate the coefficients of variation.
 - i. Deflate margins by inflation rate. Use the **pindex** variable and apply it to corn prices.
 - ii. Try to assess the effect of consumer price indexes on margins, assess scale effect on marketing services and assess price-setting behavior using linear regression like shown in this section.

Section 10

Graphs, tables, publications and presentations - how to bring them into word processor

The objective of this section is to give you the tools necessary to prepare reports, e.g. market bulletins, and learn how to move SPSS results into other applications. The methods used in this example will help you develop a system to disseminate market information.

Incorporating graphs and charts from SPSS Viewer can be done using a copy and paste procedure. Be aware that if you modify the data and rerun your procedures so that the chart or table changes, you must repeat the copy and paste procedure. We will use output that has been saved while you were running the various sections of this tutorial.

1. Go to **File /Open /Output...**
2. Select any of the files in the sample folder where you saved your output from the sample session (*.spo extension -) and open it.
3. Scroll down through the outline (left hand side of the Viewer) until you find a chart or table. Click on that item to select it. The item appears on the right hand side of the Viewer with a box around it and a red arrow on the left outside the box.
4. To edit this object, double-click on the object (or right click and select the last option in the dialog menu - SPSS Pivot Table Object to open the SPSS pivot table editor or SPSS Chart Object to open the SPSS chart editor).

It is highly recommended that you make all modifications to the graphs or tables while the object is still in the Viewer. Once the object is copied to a word processor, it is considered a picture. Making a change to the “picture” within a word processor may be allowed, however another graphic package is used and that package can change the image drastically. It will no longer retain the original format or layout.

5. When you have finished making adjustments to the table or graph, exit the edit box (click somewhere outside the box) to return to the Viewer.
6. There are two methods that can be used to take a copy of the table or graph and place it into another application.
7. Right click on the selected object, select **Copy** or use **Edit /Copy** through the menu system. This method copies the object to the clipboard.
8. Right click on the selected object, select **Export**. Within the Export Dialog box there are choices to be made. If you are exporting a chart, change the **Export: to Charts Only**. Under Export File - **File Prefix** you can specify the directory and file name where you want the exported file to be stored. If the object is a chart, you have a choice of formats under Export Format - **File Type**: The format BMP (Windows Bitmap) seems to work best. If the object is a table, you have two choices - HTML or text file.
9. Now open your word processor software if it is not already open.
10. Select a spot in your document where you would like to place the graph or table.
11. If you chose to use Copy from SPSS then:
12. Go to the **Edit** menu and select either **Paste** or **Paste Special**
 - i. If you select **Paste**, the graph or table will be pasted directly on your document. Adjust the size the graph as you wish.
 - ii. If you select **Paste Special**, a dialog box appears. You can select the type of paste/link you would like to set up. Select **Picture**. Adjust the size of your graph in your document.

13. If you have selected **Export** and saved the file - you can insert the file at the selected spot of your document.
14. Save your word processor document.

Remember, it is recommended that you make all modifications to the table or graph in SPSS before you bring it into your word processor.

Exercise 10.1.

Repeat steps 1 to 10 but instead of a graph, use a bar or a pie chart. Practice making various changes to the charts and tables in SPSS and copy the output to your word processor document. Create your own time series output notebook.

Annexes

The following annexes were prepared for users of the sample session as a brief reference guide, to explain the various functions of the SPSS commands most commonly used in the sample session, to describe the numerous options available to the user within the various menus and finally, to help manipulate results in the Viewer.

Filters Versus Temporary Selections

You can filter or delete cases that don't meet the selection criteria. When you set a filter from the **Select** command, unselected cases are filtered. Filtered cases remain in the data file but are excluded from analysis. SPSS creates a filter variable, `FILTER_$`, to indicate filter status. Selected cases have a value of 1; filtered cases have a value of 0. Filtered cases are indicated with a slash through the case (row) number in the Data Editor. To turn filtering off and include all cases in your analysis, select All cases in the **Select** command.

Another way of selecting specific data for analysis, without using solely the **Select** command, for filters or selecting out (i.e. eliminating) data, is to use the **Temporary** and **Select** command together. The **Temporary** command signals the beginning of temporary transformations that are in effect only for the following procedure. It will be in effect until the next command that reads the data. But it may include transformations like data manipulation (i.e. the transformations will apply for selecting data and for another command such as a table, graph, or regression and so on). This command is not available through the menus so we must type it in the Syntax Editor.

The Three Line Charts and Three Data in Charts Options

The **Line Charts** dialog box gives you three options for the type of line chart: simple, multiple and drop-line. In the dialog box, select the icon for the chart type you want, and select the option under Data in Chart Area that best describes your data. You can see a description of the three available Data in Chart types below. A category axis on a chart is an axis that displays values individually, without necessarily arranging them to scale. (A scale axis, in contrast, displays numerical values to scale.) Bar charts, line charts, and area charts usually have one category axis and at least one scale axis. Scatterplots and histograms do not have a category axis. The Missing Values options are available only when the new chart will display or summarize more than one variable (not including variables that define groups):

- Exclude cases listwise excludes a case from the entire chart if has a missing value for any of the variables summarized.
 - Exclude cases variable by variable excludes a case separately from each summary statistic calculated.
- Different chart elements may be based on different groups of cases.

Display groups defined by missing values is available only when you use a categorical variable to define groups for a new chart. If selected, each missing value for the grouping variable (including the system-missing value) will appear as a separate group in the chart. If not, cases with system-missing or user-missing values for the grouping variable are excluded from the chart. It is recommended to always uncheck this box as it is not of interest to show on a graph the user missing values or system missing values.

Simple lines

Summaries for Groups of Cases

Categories of a single variable are summarized. The y-height of the points is determined by the Line Represents option. A single Category Axis variable.

Summaries of Separate Variables

Two or more variables are summarized. Each point represents one of the variables. Two or more Line Represents variables.

Values of Individual Cases

A single variable is summarized. Each point represents an individual case. A single Line Represents variable.

Multiple lines

Summaries for Groups of Cases

Categories of one variable are summarized within categories of another variable. The y-height of the points is determined by the Lines Represent option.

A Category Axis variable (Category Variable 1).

A Define Lines by variable (Category Variable 2).

Summaries of Separate Variables

Two or more variables are summarized within categories of another variable.

Two or more Lines Represent variables (Var 1, Var 2).

A Category Axis variable (Category Variable).

Values of Individual Cases

Two or more variables are summarized for each case.

Two or more Lines Represent variables (Var 1, Var 2).

Manipulating Output in SPSS

Numerous modules could be dedicated to working with the Viewer. One suggestion would be to follow the tutorial within SPSS to learn about the countless possibilities and options which are available to the SPSS user in the Viewer. Your results have never looked this good! Easier and faster data exploration and the ability to drag icons in the Viewer outline and content panes on the left, expand and collapse the outline - help you to see the output you want; multi-dimensional pivot tables, swapping and hiding rows and columns, new and numerous styles for charts and tables, colors, fonts, line styles, text attributes; no loss of any custom formatting, dragging output from SPSS to a word processor (in windows metafile format); change a title directly within the output, right click for pop-up menus as shortcuts, and much more.

The object of this annex on output is to invite you to manipulate the output as much as possible. Of note, you may have trouble viewing the complete output following a SPSS command like **Frequencies** or **Tables**. It may run hundred and thousands of cases but will only show the first 50 for example. To view all of the specific output in this case, simply double click or right click on the selected output and choose **Open**. A new window opens and is called a pivot table. You can scroll down to see the whole output. You may also edit the table here as well. Enjoy using the various options given to you to modify the styles, formats, colors, text attributes and so on.