

**SPSS for Windows SAMPLE SESSION**

# **Time series**

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

**Prepared by  
Jean-Charles Le Vallée, Visiting Research Specialist  
levallee@pilot.msu.edu**

**Department of Agricultural Economics, Michigan State University  
East Lansing, Michigan  
February 1999**

**Module 0** - Paper describing the notion of levels, time series appendix and paper on file structure for SPSS for Windows (Data and Syntax Editors and Output Navigator). Must be read before starting the sample session.

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

**Module 2** - Restructuring Data Files: Table Lookup & Aggregation

**Module 3** - Entering data from other sources into SPSS, examples for dbase and spreadsheet, opening and saving data.

**Module 4** - Data Cleaning and Verification

**Module 5** - Basic Analysis

**Module 6** - Seasonal Analysis

**Module 7** - Trends

**Module 8** - Real Prices, Price Graphing and Tables

**Module 9** - Margin Analysis

**Module 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 for Windows 7.5.

### **Acknowledgments**

Funding for this research was 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 (USAID). The author would like to thank the Agricultural Economics Computer Service for the first two modules of the cross-sectional sample session which have been adapted here for this time series sample session for SPSS for Windows. The author also thanks Jim Tefft for his contribution in developing, and help in revising the time series sample session. The author also thanks Dr. Weber for the concept.

**SPSS for Windows SAMPLE SESSION**  
**Module 0 - Levels, time series and file structure for SPSS for Windows 7.5**  
**(Data, Syntax and Output windows)**

**Short Course Training Materials**  
**Designing Policy Relevant Research and**  
**Data Processing and Analysis with SPSS for Windows 7.5**  
**3<sup>rd</sup> Edition**

**Department of Agricultural Economics, Michigan State University**  
**East Lansing, Michigan**  
**February 1999**

The following module introduces the basic concepts of levels, the notion of time series and consequently, the methods of data organization. This module will also give a brief description of the file structure of SPSS for Windows version 7.5. It is essential that you read through this module before starting the time series sample session.

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

It should be clear from the preceding discussion in the paper on levels that for any given survey there is a wide range of possible data organizations, ranging between two extremes.

In the one extreme, each individual data value could be stored as a separate case. This would result in a huge number of cases, and would require a large number of key variables in order to identify each case. There would be only a single non-key variable on each case, and it would contain the data value. In the other extreme, all of the data could be entered as a single, extremely long case. It would not require any key variables, but there would be a huge number of data 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 will be performed on the data.

The rules for determining data organization that have been given in this paper are based on the principles of relational databases. If you follow them, you will always get a good, 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.

### **An Example—Time Series Data**

One situation where it might be advantageous to depart from the rules given above is when dealing with data that has a time component. Let's take as an 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 twenty products in forty markets.

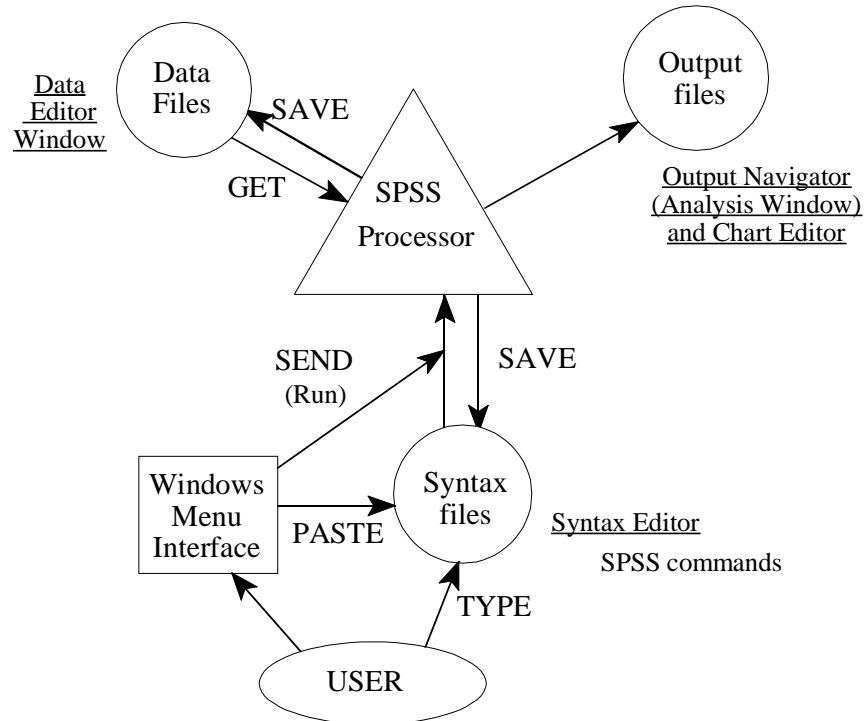
If you look at this data primarily from the perspective of time-series analysis, it would be tempting to organize the data so that each case is an observation in time, i.e. one week's data. To do this, however, would require 2,400 variables per case (3 prices by 20 products by 40 markets). If you had a software program that allowed 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 this type 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 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 for Windows 7.5

While using SPSS for Windows 7.5 in the manner taught in this tutorial, you are dealing with three different windows within the program—the Syntax Editor, the Data Editor window and the Output Navigator (including charts), the contents of each can be saved into the appropriate SPSS for Windows 7.5 file type. 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 Syntax Editor

The Syntax Editor is where 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 or paste commands for later processing by SPSS for Windows 7.5,
- To send these commands to SPSS for Windows 7.5 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 on the SPSS for Windows 7.5 Toolbar. Once you press the Run button, the computer sends the command(s) to the processor, which reads the commands written in the Syntax Editor and executes them. When all the commands have been processed, SPSS leaves you in the

Output Navigator to examine the results of your 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.

When you have successfully completed each step in your analysis (or when you are ready to end an SPSS for Windows 7.5 session, even if it was not completely successful) you should save the commands to a file for future use. To save the commands, make the Syntax Editor active and select **Save** from the File menu. A file created from the Syntax Editor is called the *syntax file (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 naming syntax files. *REP7.SPS*, *DEM-ALL.SPS*, and *SECTION1.SPS* are some examples.

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**. In the **Files of type** scroll-down menu box, select **Syntax(\*.sps)** 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 type them again. If you make changes to the Syntax file that you wish to keep, make sure you save them to disk again.

### The Data Editor Window

SPSS for Windows 7.5 stores your data in a *data file*. In addition to the values themselves, a data file contains such things as variable labels and value labels, formatting information, missing-value specifications, etc. To do any data analysis in SPSS for Windows 7.5, you must first tell SPSS to open a Data file. You do this by pulling down the File menu, selecting **Open**, and then either pasting the command to the Syntax Editor and running the command, or running 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 for Windows 7.5 in the Data Editor window.

You will often get a data file, then compute new variables and do other transformations, and finally want to save the modified set of data for later use. 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. You do this by having the Data Editor window active and selecting **Save As...** from the File menu and giving the file a new name. Or you may choose to write over the old file by saving the file keeping the same file name.

### The Output Navigator

SPSS for Windows 7.5 automatically writes all messages and output that result from the execution of your commands to the Output Navigator. For example, if you run a frequency command, then the frequency table you specify will be written to the Output Navigator. Similarly, if you generate a table or a graph, the table or graph will appear in the Output Navigator. To save the contents of the Output Navigator to a file, make the Output Navigator 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 for Windows 7.5 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 Output Navigator 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 explorer in Windows 95). In the Output Navigator, there are two boxes: the one on the right produces the results, the one on the left lets you manage the results, deleting data, renaming titles, moving results around, and so on.

## Summary of the File Types

**Syntax files** (or command files) contain commands saved in the Syntax Editor. They do not contain output or data—only commands. Like in SPSS version 6.1.3., the extensions are \*.SPS (was \*.LOG in SPSS/PC+).

**Output files** contain statistical output, data information and presentation (tables, graphs, charts), generated by the SPSS for Windows 7.5 processor, given selected commands. They do not contain data. The new extensions are \*.SPO (was \*.LIS in SPSS/PC+ and \*.LST in SPSS 6.1.3).

**Data files** contain data, including original survey variables plus new created variables through various SPSS for Windows 7.5 commands such as the COMPUTE or AGGREGATE commands. Data files are made accessible to SPSS for Windows with a **Open...** command. For SPSS 6.1.3. and 7.5., the extensions are \*.SAV (was \*.SYS for SPSS/PC+).

\*Nota bene - Versions of SPSS software. Although originally written for SPSS 6.1.x, the versions available for download were re-written for SPSS 7.5.x. If you are using a newer version of SPSS 8.x or 9.x, please note the following:

For SPSS 8.x users, you may continue to follow the short courses, versions of SPSS 7.5.x and 8.x are compatible. The only changes in syntax across these versions involves the addition of new commands (most prominently, IGRAPH in SPSS 8.x), which are not included in the sample sessions. There may be some changes to sub-commands scattered across the many existing procedures but they are very few. The same is true for the menu options between 7.5.x and 8.x; most changes are to accommodate new procedures.

For SPSS 9.x users, SPSS did change the Statistics menu extensively, calling it "Analyze," and reordering some of the choices. The documentation for the short courses were developed with pre-SPSS 9.x menus, and might be confusing for SPSS 9.x users. The sample sessions may still be used but extra effort is required by the user to translate the pre-SPSS 9.x Statistics commands to work with the new "Analyze" menu commands, in most cases, simply by replacing the word Statistics for Analyze.

**SPSS for Windows SAMPLE SESSION**  
**Module 1 - Basic functions: SPSS files, Descriptives and Data Transformation**

**Short Course Training Materials**  
**Designing Policy Relevant Research and**  
**Data Processing and Analysis with SPSS for Windows 7.5**  
**3<sup>rd</sup> Edition**

**Department of Agricultural Economics, Michigan State University**  
**East Lansing, Michigan**  
**February 1999**

This is a self-paced training aid designed to introduce the commands needed for some typical survey analysis processes in SPSS for Windows 7.5. To use it most effectively, you will need a knowledgeable SPSS for Windows user to help you get started and to answer questions while you work independently through the session. It is nonetheless partly intended to be a completely stand-alone training tool, although it may also be used as a guide for classroom training.

A copy of the questionnaire on which the data is based can be found in the Mozambique project 1992 **NDAE Working Paper 3: A Socio-economic survey of the smallholder survey in the province of Nampula: Research Methods**, copies of the three tables which were made available and can be found at the end of the manual in the annex section (for further information please contact Dr. Weber at webermi@pilot.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 for conversion of units of measure.

Questionnaire Section	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

We recommend that you complete each section in a single sitting. These sessions make the following assumptions:

- You know how to use Windows with a mouse
  - The six data files 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.
  - Preferences in SPSS are set to list variables in the same order they are listed in the file
  - Preferences in SPSS are set to list commands in the Output Navigator
- You can modify any of the **Preferences** using **Edit/Options...** from the menu system.
- Syntax Editor does not come up at start



Note: Always remember to save, at the end of each module, the syntax and output you have created and used with corresponding filenames such as Module1.sps and Output1.spo in a separate directory. The person who is assisting you will show you how to start SPSS for Windows 7.5. Open your SPSS software. If you have not read or completed Module 0, please do so now to clarify the concept of the **Syntax Editor**, where you paste or type commands, the **Output Navigator** where SPSS for Windows displays the results of your commands and the **Data Editor** window where the working data file is displayed.

## **Data Files and the Working File**

Data from questionnaires that has been entered into SPSS for Windows 7.5 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, making it the working file. This means that the data from this file is now available for you to use. Let's start with the 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...**  
*This will open the Open Data File dialog box.*
2. Change to the directory your sample session data is in and select the file **c-q1a.sav**.
3. Click on the **Paste** button to place the command in the **Syntax** window.  
*The Syntax window will now become the active window and you will see the text*  

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

*in the Syntax window.*
4. Place the cursor anywhere on the line containing the "GET" command and click on the **Run** button on the Toolbar.

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. It lets you 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, and the information about that variable will appear to the right.

This display shows additional information, including the value labels for variables like **district**, **vil**, **ca1**, **ca2**, **ca4**, **ca5**, **ca6**, and **univ**, the type of variable, the display width of the variable in characters, the number of decimal places (if type is Numeric), and any missing values.

Click on the **Close** button when you are finished.

To have all of this information written to your Output Navigator for later examination, do the following:

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

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

This information is included in your Output Navigator. You can see the name of each of the variables, their labels, and the various formats, e.g. F8.2 means width 8 with two decimal places. Displaying and saving this information about your files provides you with one way to document your data files.


### **Descriptive Statistics - involving one variable**

The first thing to do when starting analysis is to get 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 something is probably wrong.

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 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.*

This will give you a "feel" for what your data is like. It might also help point out obvious errors, e.g. a variable whose values are missing for all listed cases. Decide which of the variables 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.





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

1. From the **Statistics** menu select **Summarize/Descriptives...**  
*This will give you the Descriptives dialog box*
2. Select **ca3** 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 window and make the Syntax Editor active by clicking on the syntax button on the windows taskbar.
4. 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 Output Navigator will become active and the results of the command will be there. You will see that the mean for **ca3** is 21.34.

5. 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 the **Statistics** menu select **Summarize/Frequencies...**
6. Select **ca3** from the list on the left and click on the  button.
7. Click on  button and select Histograms, click on the  button.
8. Click on  to put the command into the Syntax Editor and make it active.
9. Execute the command by clicking on the Run  button located on the Toolbar. View the distribution.

Since the variables **ca1**, **ca2**, **ca4**, **ca5** and **ca6** are categorical, we will run a **Frequencies** on them. To run frequencies, do the following:






1. From the **Statistics** menu select **Summarize/Frequencies...**  
*This will give you the Frequencies dialog box*
2. Select **ca1** from the list on the left and click on the  button.  
*ca1 will move to the Variable(s): box on the right*
3. Repeat step 2 until **ca2**, **ca4**, **ca5** and **ca6** have all been moved to the **Variable(s):** box; you may select all the variables at once by holding the control key and clicking each of the variables - then moving them together through the  button.
4. Click on  to put the command into the Syntax Editor and make it active.
5. Execute the command by clicking on the Run  button located on the Toolbar.

The Output Navigator will become active and the results of the command will be there. You will see, for example, that **ca1** shows that 70.7% of the household members work and that **ca6** shows 38.0% of those surveyed have 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 7.5 pages 161-169.

Another command used to produce many types of descriptive statistics is the **Explore** command. One of the most useful statistics it produces is finding *outliers*. The **Explore** command can produce large amounts of output if used with its defaults. We will limit the output to statistics.

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

1. From the **Statistics** menu select **Summarize/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**.  
*This will give us statistics only and no plots.*
4. Next click on the  button.  
*This will bring up the Explore:Statistics dialog box.*
5. Click once on the square next to **Outliers** to put an X in the box.  
*You will notice there is already an X in the box next to **Descriptives**.*
6. Click on the  button.  
*This will bring you back to the Explore dialog box.*
7. Click on  to put the command in the Syntax Editor.
8. Click on  Run.

This will show you the five highest and five lowest values occurring for **ca3**, so you can tell if you have any extreme *outliers*. They will be identified by their case numbers. Refer to pages 171-177 of the SPSS for Windows Base System User's Guide Release 7.5 for an explanation of the **Explore** command.

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

**Exercise 1.1:** Run descriptive statistics on another sample file. Use the production questionnaire - Table IV, whose data is in 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** & **p6** are categorical variables.

A small sampling of what you should find out 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	
Total	1693	100,0		


**Descriptive Statistics**

	N	Minimum	Maximum	Mean	Std. Deviation
PROD THIS YR - # OF UNITS	1670	,0	5000,0	26,353	163,436
PROD NORMAL YR - # OF UNITS	1598	,5	5000,0	22,815	159,510
STOCK ENTERING HARVEST - # OF UNITS	173	,0	30,0	2,523	4,575
STORED FOR CONS THIS YR - # OF UNITS	1231	,0	1460,0	15,612	86,104
STOCK FOR SEED - # OF UNITS	869	,0	100,0	4,938	6,876
Valid N (listwise)	151				



## Descriptive Statistics - involving two or more variables

The **Crosstabs** command produces tables showing the distribution of cases according to their values for two or more categorical variables.

Look at the household member questionnaire from the annex, TABLE IA. One thing you might be interested in knowing is how the sex 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. This kind of summary can be produced using the **Crosstabs** command. 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. Change to the directory your sample session data is in and 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 **S**ummarize from the **S**tatistics menu
2. Select **C**rosstabs...  
*This will bring up the Crosstabs dialog box*
3. Select **ca4** from the list on the left and click on the  next to **R**ow(s):
4. Select **ca2** from the list on the left and click on the  next to **C**olumn(s):
5. Click on the **C**ells... button  
*This will bring up the Crosstabs:Cell Display dialog box*
6. In the **C**ounts section click on the box next to **O**bserved to place an X in it, if there is not already one there.
7. In the **P**ercentages section click on the boxes next to **R**ow and **C**olumn to put X's in them.
8. Click on **C**ontinue
9. Click on **P**aste
10. 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.

The **Compare Means** command is somewhat similar to **Crosstabs**, but it gives 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. Another way to look at the relationship between **Crosstabs** and **Compare Means** is that, **Crosstabs** is a way of getting **Frequencies**-type output broken down by categories of one or more other variables, while **Compare Means** is a way of

Getting **Descriptives**-type output 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 the different ages represented, which would be an unusable format. Instead we will use **Compare Means**.

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

This command will calculate means of the dependent variable, which should normally be a continuous variable. The means will be calculated separately for each different value of the independent variable, which should be a categorical variable. From this output you find that the average age of heads of households is 41.5 while the average age of their spouses is 33.2.

### 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.

To categorize a variable, you use the **Recode** command. 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**, **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** into a new variable called **age**.

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** 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
13. Press <Tab> and type 10 in the second box.
14. 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 "l" key to bring you to the "New Value" box.*
15. Type 1 for the first age group.
16. Click once on 
17. Repeat steps 11 through 16 to recode ages 11 thru 19 to 2 and ages 20 thru 60 to 3.
18. To recode ages 61 and up to 4, click on the circle next to Range:  through highest
19. Enter 61 in the box and repeat steps 14 through 16 using 4 for the value.
20. Click on 
21. Click on 
22. 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 .
VARIABLE LABELS age 'age group'.
EXECUTE .
```
23. Run the command

**Recode** changes the values for **age** to the codes we want to use—1,2,3, and 4. We will switch to the Data Editor to view that the changes were made. To switch to the Data Editor:

*This time we will use a different method than we used earlier.*

1. Select **c:\sample\q1a.sav** from the **W**indow menu.
2. Scroll through the window 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** to the same format as our other variables use **Data/Define Variable...**

1. Switch to the Data Editor window if you are not already there.
2. Click once on the gray bar where the variable name **age** appears.



3. Click on the **Data** menu.
4. Select **Define Variable...**
5. Click on **Type...**  
*The Define Variable Type: age dialog box will appear.*
6. In the box next to **Width:** type 1.
7. In the box next to **Decimal Places:** type 0
8. If the circle next to **Numeric** is not selected, select it.
9. Click on **Continue**  
*We will complete this procedure in the next set of steps.*

This tells SPSS for Windows to display **age** with a width of 1 digit and no decimal places. When you **Recode** a new variable, it does not have *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 **ca4**). The name also does not tell us what the individual values of a categorical variable refer to. To make the output more understandable, we 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 for later use. The best time to add labels is immediately after you create the new variable, because if you postpone it you may forget. The recode command facilitates this by allowing you to add the **Variable Label** when you do the recode. To add the **Value Labels** use the following steps:

*You should still be in the Define Variable dialog box from the last set of steps.*

1. Click on **Labels..**  
*The Define Labels: age dialog box will appear.*
2. If there is no text in the **Variable Label:** box, enter the text "Age Group" there.  
*There should be text there since we created the label when we did the recode.*
3. Go to **Value:** in the Value Labels section and type a 1
4. Press <Tab> once and type 0 to 10
5. 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.
6. Repeat steps 3 through 5 using the following information:
 

Value:	Value Label:
2	11 to 19
3	20 to 60
4	61 and older
7. Click on **Continue**
8. Click on **OK**  
*In the Data window you will see that age is now displayed as a single digit.*
9. Select **Variables...** from the **Utilities** menu.
10. Click on **age** to verify the changes you just made.
11. 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 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 wipe out 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** 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 **Statistics/Summarize/Crosstabs...** from the menus.
2. Use **age** for Rows and **ca2** for Columns.
3. Check the proper selections in the **Cell** section.
4. Paste the command and run it.

From this, you can see that 12% of heads of households are 61 years of age or older. Also, that of the people 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** earlier. To do this, we will explore SPSS's ability to switch between the **Syntax**, **Output** and **Data** windows.

To switch to the **Output Navigator**:

1. From the **Window** menu select Output1 - SPSS Output Navigator.
2. Scroll back through the 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 Syntax1 - SPSS Syntax Editor.
2. Scroll through the window with the scroll bars.

To switch to the **Data Editor**:

1. From the **Window** menu select C-q1a - SPSS Data Editor.
2. Scroll through the 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 95 taskbar, found by default at the bottom of the screen (the taskbar may be moved to any sides 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 Questionnaire, 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	RIBAUÉ	ANGOCHE	
Household size	1 to 4 members	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%
	5 to 7 members	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%
	8 or more members	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, you can see for Monapo for example, 34,8% of all 1 to 4 member households (group 1) are found within Monapo and that 60,7% of all households in Monapo have 1 to 4 members in a household.

Before exiting SPSS for Windows 7.5, we should save the contents of the Output Navigator. The output window contains all of the commands 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.

1. Make the Output Navigator active using its icon in the Windows 95 taskbar.
2. From the **F**ile menu select **S**ave **A**s...
3. Enter the filename **session1**  
*The .spo extension will be added to the name automatically.*
4. Click on **Save**

To exit SPSS for Windows:

1. From the **F**ile menu select **E**xit 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
3. Click on **Save** and give it a filename such as Module1.sps, then SPSS for Windows 7.5 will exit.

**SPSS for Windows SAMPLE SESSION**  
**Module 2 - Restructuring Data Files - Table Lookup & Aggregation**

**Short Course Training Materials**  
**Designing Policy Relevant Research and**  
**Data Processing and Analysis with SPSS for Windows 7.5**  
**3<sup>rd</sup> Edition**

**Department of Agricultural Economics, Michigan State University**  
**East Lansing, Michigan**  
**February 1999**

Some types of data analysis will require restructuring of the data files. The data from the four questionnaires—household, member, production and sales—are arranged in four separate data files because the data are at different levels. The household data is at the most general, or highest, level. The other three files contain more detailed data, which is usually thought of as being at a lower level. 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.

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				

Given the data in the current form, many transformations will be needed 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 in each of these:

- **C-Q1A.SAV**: This file contains data on household member characteristics. It is at the household-member level. The variables **ca3**(age) and **ca4**(sex), are of use to us in this exercise for computing the number of adult equivalents per household.

- C-Q4.SAV: This file has household-product level variables, some of which we will need for this exercise. The variables we need are the following:
  - a. **prod**, containing codes for the agricultural crop produced.
  - b. **p1a**, containing codes for the unit in which the production was measured (100 kg sack, 50 kg sack, etc).
  - c. **p1b**, the number of units produced this year.

*Note that the units of production are not standard units.* For example, a "100 kg sack", as the term is used in Mozambique, weighs 100 kg only when filled with corn. When filled with manioc root for example, 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* of conversion factors specifically created to deal with the problem of non-standard units. It contains conversion factors equal to the weight in kilograms of each product in each unit in which it was observed. 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 different from that for a 50 kg sack of cotton, which is different from that for a 50 kg sack of manioc root. The variables in this file are:
  - a. **prod**: This variable refers to the product.
  - b. **unit**: This variable refers to the unit.
  - c. **conver**: This is the conversion factor, and is equal to the number of kilograms corresponding to each combination of **prod** and **unit**.

Below, a sample of data from CONVER.SAV shows that, a 20 liter can (**unit**=8) of rice (**prod**=7) weighs 19 kg; a 50 kg bag (**unit**=24) of rice weighs 53 kg; a 20 liter can of beans (**prod**=30) weighs 17 kg; and a 50 kg bag of beans weighs 47 kg.

<b>prod</b> (Product)	<b>unit</b> (unit)	<b>conver</b> (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 converting kilograms of food into calories of food. It contains two variables:
  - a. **prod**, the product variable
  - b. **calories**, equal to the number of calories per kilogram of each of the foods recorded in the survey.

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 each household contained. We can generate a file with this information using data from the member file, C-Q1A.SAV.
3. We need to get the results from steps 1 and 2 together in a single file in order to compute calories produced per adult equivalent per day.

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

In 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 Table IV 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. Thus, after converting all production to calories, we must, for each household, sum the calories from each product to arrive at the household total.

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


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

We are only interested in the seven staple food crops (corn, nhemba bean, manteiga bean, manioc, rice, sorghum, and peanuts). Looking at **prod** in the questionnaire, we find that these products have codes of 47, 30, 31, 41, 6, 44 and 5. To make only these cases active we use **Select Cases**. **Select Cases** selects a subset of the cases based on particular criteria. **Select Cases** can either filter out the unselected cases or delete the unselected cases. If you delete the unselected cases you can return to the original file as long as you do not save the current working file under the same name as the original file. If you filter out the unselected cases (which we will be doing since it is a safer method) you can always unfilter the data which will activate all of the cases in the file.

1. From the **Data** menu select **Select Cases...**  
*You should see the Select Cases dialog box.*
2. Select the radio button next to **If condition is satisfied**
3. Click on  under **If condition is satisfied**

4. Click **in** the box, to the right of , **not** on the button itself .
5. Enter the following text (without returns):
 



```
PROD = 47 | PROD = 30 | PROD = 31 | PROD = 41 | PROD = 6 | PROD
= 44 | PROD = 5
```

*The "/" are symbols for the word OR. We are telling SPSS to select all cases with **prod** equal to 47 or 30 or 31...*
6. Click on 
7. Select the radio button next to Filtered
8. Paste the command
9. Select the text in the **Syntax Editor** from `USE ALL` to `EXECUTE` and run the command.





Only cases with these product codes will now be used. This subset of the data will be in effect until we open another file or Select All cases (unfilter the cases).

Let's first convert all production of these seven 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 **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**.

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 CONVER.SAV and the working data file. 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 
4. Select **p1a** from the list under New Working Data File: and click on 
5. Click on   
*This will allow you to rename **p1a** to **unit** to match the conversion file.*
6. Next to New Name: type **unit**
7. Click on 
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)
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   
*A dialog box will ask you if you want to save the contents of the data window. We do not want to save it, the new file can take its place, so...*
15. Click on 
16. Select and run the command. Be sure to include EXECUTE.


The above steps tell SPSS for Windows to merge the working data file and the CONVER.SAV file, (using CONVER.SAV as a table) to add the unit variable to our working data file. Since the key variables need to have the same names in both files we rename **p1a** ( the unit variable for our working file) to **unit** (**p1a** will remain **p1a** in 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. 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:** enter **qprod\_tt**
3. Label **qprod\_tt** here, or *Total production in kg*, if you wish.
4. From the list on the left select **p1b** and click on 
5. Type \* or select the button in the dialog box
6. From the list on the left select **conver** and click on 
7. Paste, select and run the command

Next, we need to look up how many calories per kilogram each product contains. This information is in the table-lookup file CALORIES.SAV. This file has two variables—product and number of calories per kilogram. The key variable is product. In order to add the calorie-conversion variable to the working data file we need to do another merge with keyed table lookup. This time the key variable only needs to be the product variable. The data file has already been sorted by product (see the previous merge), so we don't need to sort it again.

1. From the **Data** menu select **Merge Files** then **Add Variables...**
2. Select the file calories.sav, 
3. Check the **Match cases...** box
4. Check the **External file is keyed table** box

5. Put **prod** in the Key Variables: box
6. Paste the command
7. Clear the warnings as necessary
8. Select and run the command

The new working data file produced by the merge now contains the needed calorie variable, **calories**. We can now compute total calories produced.

1. Use **Transform/Compute...**
2. Use **kprod\_tt** as the Target Variable:, representing *Total Calories Produced*
3. Enter the equation **qprod\_tt \* calories**
4. Paste, select and run the command

This gives us a working data file with total calories produced per product for each household. Now, 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. In other words, we need to create a new household-level file from the current product-level file with one case per household.

To create the new household-level file, we use **Aggregate**. **Aggregate** will create a new data file with one case per household and **kprod\_tt** summed for each. It always uses the working data file as the file to be aggregated. We already have the production file open, so we're ready to aggregate.

1. From the **Data** menu select **Aggregate...**  
*The Aggregate Data window will appear.*
2. Select **district**, **vil**, and **hh**, respectively, for the Break Variable(s):
3. Select **kprod\_tt** as the Aggregate Variable(s):
4. Click on **Name & Label...**
5. Change the default name kprod\_\_1 to **kprod\_tt**
6. Enter the following label: **Calories Produced in Staple Foods**
7. Click on **Continue**
8. Click on **Function...**
9. Select **Sum of values** and click on **Continue**
10. Select **Replace working data file**
11. Paste the command
12. Click on **No** to not save the contents of data window NewData
13. 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, and would not have become our working file. We would have had to open the file to access it.

The **Break Variable(s)** 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) creates a new variable **kprod\_tt**, which we calculate by summing **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**).

The new working data file now contains what we need, total number of calories 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

This says that, on average, a female 10 to 19 years old needs only 84% as many calories as a male 10 years or older, and that children under 10 need only 60% as many calories as the typical male older than 10. Thus for example, a child (male or female) under age 10 gets 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 their adult equivalent.

**Compute.../If...** allows us to do this. The adult equivalent variable being created is **ae**.

1. From the **Transform** menu select **Compute...**  
*The Compute Variable window will appear.*
2. For the **Target Variable:** enter **ae**
3. In the Numeric **Expression:** box enter a **1**
4. Click on **If...**
5. Select the radio button for **Include if case satisfies condition:**
6. Enter the statement **ca4 = 1 & ca3 >= 10**
7. Click on **Continue**
8. Paste the command but don't run it yet.
9. 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 becomes much easier at this stage to simply copy and paste the same command within the Syntax editor itself and then changing the variables names. It is quicker. For those who cannot perform the copy/paste manoeuver here within the Syntax editor, simply repeat the steps above as indicated.

Numeric Expression	If... Statement
<b>.84</b>	<b>ca4 = 2 &amp; ca3 &gt;= 10 &amp; ca3 &lt;= 19</b>
<b>.72</b>	<b>ca4 = 2 &amp; ca3 &gt;= 20</b>
<b>.60</b>	<b>ca3 &lt; 10</b>

10. Select all of the **If** statements and run.

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

1. You will need to select **Statistics/Summarize/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, .72, .84, and .60— and no missing cases. You can see we have nine missing cases. This tells us that our data file is missing either the age or the sex for nine people. This is something that should have been identified during the cleaning process. At this point a researcher should go back to the original questionnaires and try to fix this. Since we can't do this, we will use an alternative method.

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. To find out the average adult-equivalent value for our sample...

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

This 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 as an illustration of SPSS commands, 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 **V**ariables:
3. Click on **Old and New Values...**
4. Select System-missing
5. Select **V**alue: in the New Value section and enter **.79** in the box
6. Click on **Add**
7. **Continue**
8. Paste, select and run

Now we need 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 **D**ata menu select **A**ggregate...
2. Move **d**istrict, **v**il, and **h**h to **B**reak Variable(s):
3. Move **ae** to **A**ggregate Variable(s):
4. Click on **Name & Label...**
5. In the **N**ame: box enter **ae\_tt**
6. In the **L**abel: box enter **Adult Equivalents**
7. **Continue**
8. **Function...**
9. Select **S**um of values
10. **Continue**
11. Select **R**ep<sub>l</sub>ace working data file
12. 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, do a **Descriptives** on **ae\_tt**.

1. **Statistics/Sumarize/Descriptives...**
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 Data Editor window 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.**

Now 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 case-by-case to get both sets of data in a single file. 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're joining two files at the same level, as we're 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** because without them you have no guarantee that SPSS will combine the right cases.* 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. Use 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 gives us 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**
5. **Continue**
6. Numeric Expression: enter **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 variable for each case, showing 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. **Rank Types...**
5. Unselect Rank
6. Select Ntiles: 4
7. **Continue**
8. Paste and run
9. Note the new variable name in the **Output** window; it should be `NKPROD_A`  
*We will need to know the new variable name for the next procedure.*

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 and generate a name for it, that will contain the rankings.

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

1. **Statistics/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 Ribae is 2517.4551. The output from **Compare Means** gives you the numbers necessary for the table, although they are not formatted exactly as we showed the table at the beginning of this section. In Section 3 you will learn how to produce the same results but in a nicer-looking table format.

Save this file as `HH-FILE3.SAV`.

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

You should now save the contents of the Syntax window 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 and by clicking on **OK** instead of **Paste**. Pasting commands into the **Syntax Editor** and then running them, rather than running them directly, gives you documentation on your work and enables you to run the exact same analysis over again at a future date. Documenting now can save many steps later. For example, if you find out that some data was entered incorrectly, you can change the data and then easily run an entire analysis over again on the corrected data, using the saved syntax file, without recreating the steps.

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 **Output window**.*
2. Save the **Output Navigator** as **session2**

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

1. **File/Open...**
2. Select **Syntax(\*.sps)** in the **Files of type:** scroll down menu, then select **session2.sps**
3. **OK**  
*The **Syntax window** !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'.
EXECUTE .
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-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:
- a. The procedure is very similar to the work that we just completed.
  - b. Sales come from `c-q5.sav`.
  - c. Check the file for the appropriate variable for the quantity of sold production. Note that the product codes are the same as for `c-q4.sav`. Also check for the variables by which to sort.
  - d. 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.
  - e. Computing the calories sold involves the same basic steps as computing the calories produced. (Step 1)
  - f. Merge this newly created file, (the file containing calories sold), with the file containing calories produced, `hh-file3.sav`.
  - g. Keep in mind that only 256 households sold products, but all 343 households produced and retained calories. If the calories-sold variable is missing, it means the household did not 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
DISTRICT          1  MONAPO                1098.8770  401.0378    84
DISTRICT          2  RIBAUE                1232.8030  350.2260    29
DISTRICT          3  ANGOCHE                912.7559   384.7468    28

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

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

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

Total Cases = 343

```

**SPSS for Windows SAMPLE SESSION**  
**Module 3 - Entering data from other sources into SPSS, opening and saving files**

**Short Course Training Materials**  
**Designing Policy Relevant Research and**  
**Data Processing and Analysis with SPSS for Windows 7.5**  
**3<sup>rd</sup> Edition**

**Department of Agricultural Economics, Michigan State University**  
**East Lansing, Michigan**  
**February 1999**

Data can be entered directly into SPSS for Windows 7.5, or read from a variety of sources. SPSS for Windows can access data stored as Lotus 1-2-3, Quattro Pro and Excel files (spreadsheets), dBase files (databases) and ASCII files (text files). Data stored in text files are read in any of three formats: fixed column, free field and tab delimited. In this module, an example will be provided for reading databases as it is the format under which many market information systems enter their raw data such as for the Senegal SIM (Système d'informations de marché). We will also discuss reading spreadsheets from Quattro Pro as we will be using such a file to import a price index to compute real or deflated prices in module 8 of the time series sample session.

### **Opening and saving data files**

The **Open** command from the **File** menu opens an existing data file in SPSS or in another format. To read unformatted data from a text or ASCII file, select **Read ASCII Data** from the **File** menu, then select a format, **Freefield** or **Fixed Columns**.

What to do? Select or enter the name of the formatted data file to open. The file list lets you choose the file to open. The **Directories** list allows you to choose a directory (**Look in:**). You can also select a drive by choosing one of the options from the **Directories** list. You can select the type of file that you want to open from the **Files of type:** list. You must select one of the alternatives to describe the type of data file. You may also type the fully-qualified filename into the **File name:** box. Available data file types are: **SPSS**, **SPSS/PC+**, **Systat**, **SPSS portable**, **Excel**, **Lotus**, **SYLK**, **dBASE**, **Tab-delimited**, and you may also open a **SPSS syntax file**, an **SPSS output Navigator document**, an **SPSS script** or if you wish, you may select to view all files in the current directory.

Similar controls are used to open or save a file in both dialog boxes: an edit field (drop down list) at the bottom initially contains an extension specification such as **SPSS (\*.sav)**. The main box above the edit field lists all files in the default drive and directory that match the extension. You can select a file from the list, or change the extension to see a different list of files.

When opening or saving data files or exporting charts, the file type selection in this list determines the file format that will be read or written. You must set the file type correctly to read a data file. For other files, this is simply a shortcut for the extension that governs which files are listed. You can override the file type by typing an extension at the top and pressing **Enter**.

The default drive and current directory are listed above, which contains a visual hierarchy of directories represented by folder icons (viewed by opening the drop down list). The last open folder

icon is the current directory. To display a different directory or drive, double-click it, or click it and press **Enter**. To open a file from or save a file onto a different drive, select the drive from this list.

## **How SPSS Reads 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). There are no "empty" cells within the boundaries of the data file. 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:

- Rows are cases, and columns are variables.
- Data type and width for each variable are determined by the column width and data type of the first data cell 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.
- 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.
- If you don't read variable names from the spreadsheet, SPSS uses the column letters (A, B, C, etc.) for variable names for Excel and Lotus files.

Keep this information in mind as we will need it in module 8, as we import a data file from a spreadsheet to merge a price index to deflate nominal prices.

## **How to Open Database Files and Enter the Data into SPSS?**

This sub-section will give you the tools necessary to read files from other sources. As you will be entering a dBase file from the Senegal SIM (Market Information System) on cereals for the 1995 year, first ensure that the **cer95.dbf** file is on your hard disk in your sample folder in your C drive (with the other files for this sample session).

1. Open you SPSS for Windows 7.5 software and go to the **File/Open...** menu
2. From the C:\Sample directory, click on the arrow on the **File of type** drop down menu, scroll to the **dBase** choice and click on it.
3. Click on **cer95.dbf** in the File Name list box
4. Click on the **Paste** button to place the command in the Syntax window. The Syntax Editor window will now become the active window and you will see the text  
GET TRANSLATE  
FILE='c:\sample\cer95.dbf'  
/TYPE=DBF /MAP .  
EXECUTE .
5. Place the cursor anywhere on the line containing the **Get** command and click on the Run  button on the Toolbar.

Now go to the **Data editor** and look at the data. You will see many price, quantity and other variables for various cereals as well as variables for regions, markets and dates. The first variable (**d\_r**) is the field used by dBase to mark cases for deletion and is not usually of interest when running analysis. It is worth removing this variable as it takes up space in the file and on your hard disk. You can do so from the **Data Editor** window by clicking once on the variable name to select it and then

pressing the delete button on your keyboard or choosing **Clear** from the edit menu.

Databases allow fieldnames to be longer (up to 40 characters) than SPSS variable names (8 characters). SPSS will truncate each long fieldname to 8 characters so it is always good to select fieldnames in dBase no longer than 8 characters if you wish to preserve the same variable names in SPSS. You can see in the **Output Navigator** how SPSS truncated over 17 variables! You can also see in the **Output Navigator** just below, the summary report of how the database fields are read by SPSS for Windows 7.5.

Now take a look at the **date** variable in the **Data Editor** window. See how the variable is presented in a **day-month-year** format. For presentation purposes, this variable may be widely used such as in tables or for labeling category axis in graphs and so on. Double click on the variable name, the **Define variable** dialog box will appear. Click on **Type**: see the multitude of ways in which to change the presentation of dates!

Unfortunately, this variable may not be used for analysis or for data manipulation and organization such as with aggregating or selecting data by time or date. And worse, this variable cannot be converted within SPSS to date variables like week, month or year, variables which would allow us to organize and aggregate data correspondingly to analysis needs. Hence, it is important to enter data in such a way to allow for analysis and organization of all the data, which are not possible at present. The first suggestion to come to mind is to go back to dBase and decompose this variable into at least three new variables: one for day, one for month and another for year. It is often appropriate to create a date variable for weeks as well. Thus, it is recommended that data being entered should include these four separate variables in the data set.

### **Converting Files Into Numeric Format.**

In most cases, to do analysis with SPSS, you must first convert the data into numeric format. To avoid recoding, cleaning and associating values to the old variables manually, you can use the **Automatic Recode** command. This command converts string and numeric values into consecutive integers. When category codes are not sequential, the resulting empty cells reduce performance and increase memory requirements for many SPSS procedures. Additionally, some procedures cannot use string variables, and some require consecutive integer values for factor levels. The new variable(s) created by **Automatic Recode** retain any defined variable and value labels from the old variable. For any values without a define value label, the original value is used as the label for the recoded value. A table displays the old and new values and value labels.

String values are recoded in alphabetical order, with uppercase letters preceding their lowercase counterparts. Missing values are recoded into missing values higher than any non-missing values, with their order preserved. 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. To recode string or numeric values into consecutive integers, from the menus choose **Transform/Automatic Recode...** and select one or more variables to recode. For each selected variable, enter a name for the new variable and click **New name**.

### **Exercise 3.1.**

Now that the dBASE file is open, try to recode automatically some of the string variables.

**SPSS for Windows SAMPLE SESSION**  
**Time series**

**Short Course Training Materials**  
**Designing Policy Relevant Research and**  
**Data Processing and Analysis with SPSS for Windows 7.5**  
**3<sup>rd</sup> Edition**

**Department of Agricultural Economics, Michigan State University**  
**East Lansing, Michigan**  
**February 1999**

As for past modules, the following time series modules are a self-paced training aid designed to introduce the commands needed for some typical survey analysis processes in SPSS for Windows 7.5. To use it most effectively, you will need a knowledgeable SPSS for Windows user to help you get started and to answer questions while you work independently through the session. It is nonetheless partly intended to be a completely stand-alone training tool, although it may also be used as a guide for classroom training.

Data used in this SPSS for Windows 7.5 session was taken from weekly time series data gathered by the Market Information System (SIM) in Mali. Multiple markets were chosen to represent various market types such as producer, wholesale and consumer (or retail) markets. Data for various grains are available in the data but for the purpose of this sample session, the focus will be on sorghum.

Section	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
QuattroPro file with price index data for 1995	INDEX.WK1
QuattroPro file with production data for growth rates	MALIPROD.WK1

These sessions also make the assumptions that:


- You know how to use Windows with a mouse
- The data files described above are stored in the directory C:\SAMPLE on your hard disk
- Preferences are set to list variables in the same order they are listed in the file
- Preferences are set to list commands in the output window
- You have an understanding of basic statistics ( $R^2$ , estimate of how well the model fits the data or proportion explained by the 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, and so on).

P.S. You can modify any of the **Preferences** using **Edit/Options...** from the menu system. Always remember to **SAVE** the changes to the data after each exercise and module, using a new file name. And remember to save your output as we will be using it in module 10. It is highly recommended to save the listing in the **Syntax Editor** window and the results in the **Output Navigator**. This allows you to review, document and interpret all the analysis you are carrying out.

## The Price and Quantity Data File and the Working File

The data in SPSS for Windows 7.5 are stored in *data files*. As in past modules, if we want to work with a set of data, we must open the corresponding data file or make it active, so that it is available to the program. Again, when a data file is opened, it is loaded from the disk into memory, making it the working file. Let's start with the data file that corresponds to the price and quantity data PQ\_8995.SAV. To open this data file and make it the working file:

1. From the **File** menu, select **Open...**
2. Change to the directory your sample session data is in and select the file PQ\_8995.SAV.
3. Click on the **Paste** button to place the command in the Syntax window.  
*The Syntax window will now become the active window and you will see the text*

```
GET
  FILE='C:\SAMPLE\PQ_8995.SAV' .
EXECUTE .
```
4. Remember, you can place the cursor anywhere on the line containing the "GET" command and then click on the Run button  on the Toolbar.

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

As in the first module, it is important to know the data file you are working with and the variables it contains. We can find this out by looking at the data file as well as by using the **Variables...** command on the **Utilities** menu using the following steps:

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

This display shows on the left, the list of variables found in the data file which cover location, markets, time, prices and quantity data. And by choosing a specific variable, the display also shows the value labels for each of the variables, the type of variable, the display width in characters, the number of decimal places (if type is Numeric), and any missing values. Take your time to look at the number of markets, of commodities, the types of prices, and so on.

Choose the **p4c** variable which represents 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 explaining any missing price data! The reasons are shown just below in the display. It is important to know the reasons why data are missing. Again, these codes are not prices but have been put in the missing values category of the variable for a purpose: so SPSS for Windows will avoid including and using these values in any analysis it will perform.

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 will execute immediately. The Output window will become active and will contain a listing of all the variables with their definitions.*



This information is now included in your **Output** window. Displaying and saving this information about your files provides you with a way to document your data files.

Click on the **Close** button when you have finished viewing the data.

Other types of errors may be found in the data. Select the file and take a few moments to view the data. To view the nature of any specific variable, you may double-click on the variable name at the top of the table in the data editor window. Looking at the data, you will notice many missing values which have been replaced by a period. These are system missing values, also called a sysmis. It is important to understand the data correctly. Thus, the reason why much data seems to be missing is that the file brings together data from several different types of markets, for example, producer markets do not include wholesale market data and so on. Data may seem missing, but in actual fact, there is no missing data since there are no prices for non-existing markets! Having said this, it is possible to find a sysmis in data when values should have been collected and entered. This is a real error and should not be allowed. Or else, why is there a missing value and where is the real value? This must be explained and is not permissible.

Now, are there other types of errors once all the data has been entered? Yes. Many types of errors may occur when entering the data. Depending on the type of data and analysis one wishes to perform on the data, and to avoid any errors, cleaning exercises are carried out to ensure the data is ready and will not distort the results. This is the focus of the following module.

**SPSS for Windows SAMPLE SESSION**  
**Module 4 - Data cleaning and verification (Time Series)**

**Short Course Training Materials**  
**Designing Policy Relevant Research and**  
**Data Processing and Analysis with SPSS for Windows 7.5**  
**3<sup>rd</sup> Edition**

**Department of Agricultural Economics, Michigan State University**  
**East Lansing, Michigan**  
**February 1999**





Before performing any analysis on given data, it is always good to verify the data for errors. These errors may come from wrongly entered data or mistaken observations from the field or markets. Another type of error is logic of market prices. Cleaning exercises allow us to verify the data and for this sample session, we will verify the logic of market prices for sorghum.

Since the data is in French, here are three key translations to help you understand the levels of prices: *Prix à la production* is Producer Price; *Prix au regroupement* is the Assembly (Wholesale) Price; *Prix à la consommation* is the Consumer or Retail Price.

**Logic of market prices - producer, wholesale and retail.**

The producer's price (**p4p**) means the price of the commodity that was paid by merchants or assemblers to the producers in a given market. Wholesale price (**pr8**) means the price received by a merchant for a certain commodity from a buyer of that commodity in the market place (called assembly level in the data). Retail price (**p4c**) means the price paid by consumers to retail merchants (e.g. per kilogram). Local units of measurement may vary in size and from place to place, and it would have been necessary to convert these measurements into kilograms or metric units. As you have completed such an exercise in module 2, and for the purpose of this session, the local units have already been converted into metric units. And in Mali, the enumerators do this conversion in order to minimize the length of the data transmissions over the radio.

As we want to clean the data for sorghum, we must choose the values pertaining to this commodity. To do this, we can use the **Select** command.

1. First, make the data editor window active. If not, you may do so by clicking the  button on the Toolbar. You should have the PQ\_8995.SAV file open.
2. From the **Data** menu select **Select Cases** and select the radio button next to **If condition is satisfied**
3. Click on  under **If condition is satisfied**
4. Click on **cer** in the left column, then on , click on the equal sign "=" and then click once on the value 8.
5. The text in the box should look like: CER = 8
6. Click on 
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. To do so:

1. Go to the Data Editor window and from the **Transform** menu select **Compute...**
2. Under **Target Variable:** enter **clean**
3. Enter the value 0 in the Numeric **Expression** box
4. Paste, select and run the command

We must now identify the types of cleaning errors possible. There are three sets of prices so we must calculate three different errors, (1) the producer price is superior to the consumer price, (2) the producer price is superior to the wholesale price and (3) the wholesale price is higher than the consumer price. To program this in SPSS, you must:

1. From the **Transform** menu select **Compute...**
2. For the **Target Variable:** the word **clean** should already appear, if not, enter it.
3. Click on **Type&Labels..**  
*A dialog box will appear. We could have typed the variable label when we computed clean as 0 but we have just learnt now what the variable represents. We must also add the value labels of the three types of errors which we will do further ahead.*
4. In the **Label:** box, enter the text "Price logic error types" there. Click on **Continue**
5. In the Numeric **Expression** box, enter a **1**
6. Click on **If...**
7. Select the radio button for **Include if case satisfies condition:**
8. Enter the statement **p4p > 0 & p4c > 0 & p4p > p4c**
9. Click on **Continue**
10. Paste the command but **DO NOT** run the command yet.
11. You do not need to use the menu system at all times, you may copy and paste the syntax within the Syntax editor yourself, then replace the variables names. Or if you are familiar with the syntax you may type it directly. So, on a new line, just below **If (...)** **clean=1** statement in the Syntax editor, type two other **IF** commands with the following variables and numeric expressions:

Numeric Expression	If... Statement
2	<b>p4p &gt; 0 &amp; pr8 &gt; 0 &amp; p4p &gt; pr8</b>
3	<b>p4c &gt; 0 &amp; pr8 &gt; 0 &amp; pr8 &gt; p4c</b>

12. We want to add **Value Labels** for each of the errors. One way we can do this is to use the **Define Variable...** command from the **Data** menu in the Data Editor or double click the variable name in the **Data** editor as well. You will notice that once the function has run, the command is printed in the Output navigator, syntax which you may use to paste to the Syntax editor and run again when necessary.
13. From the Data Editor window, double-click on the variable **clean** and click on **Labels..**

14. Go to **Value**: in the **Value Labels** section and type "1"
15. Press <Tab> once and type "Producer price is superior to the consumer price"
16. Click on **Add**, or press the "Alt" and "A" keys on your keyboard
17. Repeat steps 15 through 17 using the following information:
 

Value:	Value Label:
2	Producer price is superior to the assembly price
3	Assembly price is higher than the consumer price
18. Click on **Continue**
19. Click on **OK**. If you check the Output Navigator, you will see the output is printed for this command.
20. Make the **Syntax** editor active and select all of the `IF` statements and run.

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


1. You will need to select **Statistics/Summarize/Frequencies...**
2. Use **clean**
3. Paste and run

Go to the **Output Navigator** and look at the results: 8 errors were found! In all, there were no errors where the producer price is superior to the consumer price, 7 errors where the producer price is superior to the assembly price and 1 error where the assembly price is higher than the consumer price. A logical step here is to go back and look at the questionnaires and also talk with the enumerators.

Many reasons are possible for the presence of these errors and we cannot modify the data to fit - the simplest thing to do is to avoid using these values in further analysis. One way to do so is to set up a filter which will exclude all values of the **clean** variable above 0. The method for programming the filter is the same as when we selected out sorghum for this cleaning analysis.

Now lets take a closer look at these errors. Using the **Case summaries** command, we can view the prices, markets and the case number for each of the cleaning errors. But we want to view the cleaning errors only so we must specify not to list all the cases. We can do this by replacing the filter with a new one to include the cleaning errors. To do this, you must:

1. From the **Data** menu select **Select Cases** and select the radio button next to **If condition is satisfied**
2. Click on **If...** under **If condition is satisfied**  
You will notice that `cer = 8` can still be found in the box. Click next to the value 8.
3. Click on **&** first and then on **clean** in the left column, then on **▶**, click on the superior sign ">" and then click once on the value 0.
4. The text in the box should look like : `CER = 8 & CLEAN > 0`
5. Click on **Continue**
6. Select the radio button next to **Filtered**
7. Paste the command.
8. Select **Summarize** from the **Statistics** menu
9. Select **Case Summaries...**
10. Select **mar cer mois date p4p p4c pr8 clean** from the list on the left and click

on the  next to Variable(s): (Hint: You can hold down the "Ctrl" key to select all the variables at the same time).

11. Click on the box next to **Show case numbers**
12. Paste and run both the commands

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<sup>a</sup>

	Case Number	Marché	Céréale	Mois	Jour	Prix moyen calculé à la consommation	Prix moyen calculé à la production	Prix au regroupement pondéré	Price logic error types
1	25691	KOUTIALA	Sorgho	Février	18	48,00	40,00	39,70	Producer price is superior to the assembly price
2	25693	KOUTIALA	Sorgho	Mars	4	48,00	40,00	39,80	Producer price is superior to the assembly price
3	25705	KOUTIALA	Sorgho	Mai	27	57,00	48,00	47,60	Producer price is superior to the assembly price
4	25739	KOUTIALA	Sorgho	Février	10	41,00	38,00	43,40	Assembly price is higher than the consumer price
5	25758	KOUTIALA	Sorgho	Juin	23	60,00	45,00	44,90	Producer price is superior to the assembly price
6	25775	KOUTIALA	Sorgho	Octobre	20	54,00	44,00	43,73	Producer price is superior to the assembly price
7	27020	ZANGASSO	Sorgho	Mai	14	,	40,00	32,00	Producer price is superior to the assembly price
8	27086	ZANGASSO	Sorgho	Août	19	,	40,00	33,20	Producer price is superior to the assembly price
Total	N	8	8	8	8	6	8	8	8


<sup>a</sup>. Limited to first 100 cases.

## Exercise 4.1.

Repeat the same cleaning exercise using maize as the commodity. Use all cases and then use filters to select maize. You 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 of number 2 type and 5 errors of number 3. (Hint: Maize is commodity number 9).

## Outliers

We could use the **Frequencies** command to view outliers but it produces long lists and we do not get as 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 in the first module. The use of this descriptive statistic will help us find the largest and smallest values in the data. Also we may view the distribution of the data as we can produce a stem and leaf plot using the same command.

0. Replace the current filter by selecting anew, sorghum as the commodity for analysis and run the **Explore** command on the three price variables (**p4p**, **pr8** and **p4c**) using the following steps:
  1. From the **Statistics** menu select **Summarize/Explore....**
  2. Select **p4c** from the list on the left and click on the  button next to **D**ependent List.
  3. In the lower left corner of the dialog box is a box called **Display**. Click on the radio button (circle) next to **B**oth.  
*This will give us statistics and plots.*

4. Next click on the **Statistics...** button.  
*This will bring up the Explore: Statistics dialog box.*
5. Click once on the square next to Outliers to put an X in the box.  
*You will notice there is already an X in the box next to Descriptives.*
6. Click on the **Continue** button.  
*This will bring you back to the Explore dialog box.*
7. Next click on the **Plots** button.
8. Click on the radio button **None** in the Boxplots box display.
9. Click on the **Continue** button.
10. Click on **Paste** to put the command in the Syntax Editor and run the command.
11. Repeat for the producer and wholesale prices separately.

#### Extreme Values

			Case Number	Value
Prix moyen calculé à la consommation	Highest	1	1057	775,00
		2	13295	150,00
		3	13293	150,00
		4	6453	150,00
		5	13294	, <sup>a</sup>
	Lowest	1	26994	,00
		2	26844	30,00
		3	26845	30,00
		4	25528	34,10
		5	26846	, <sup>b</sup>

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

If you look at the Output navigator, you will see descriptive statistics and the stem and leaf distribution of sorghum prices as well as the five highest and five lowest values occurring for each price. So you can tell if you have any extreme *outliers*. They will be identified by their case numbers. For sorghum there was one outlier found: the value 775. As all other values are under 150 FCFA, most likely, the person entering the data typed 7 twice, so we may ignore this value. Another recommended procedure is to check the questionnaires and see that the real value is 75

FCFA, then you may write 75, replacing 775. (Please refer to pages 171-177 of the SPSS Base 7.5 for Windows User's Guide for further explanations of the **Explore** command.)

1. A way of eliminating these extreme prices can be done using the **Missing Values** command by double-clicking on the variable name in the table (**P4C**) or using the Data/Define Variable... menu in the Data editor.
2. Once in the dialog box, click on the **Missing Values** button. There you can choose to enter Discrete (specific) missing values, a Range plus one discrete missing value, select this option and enter 997 for *L*ow: 999 for *H*igh: and 775 for *D*iscrete value: , or simply a Range of values.
3. (click continue and then on Ok). This is how the codes 997 to 999 were entered earlier. This will allow not to include these extreme values in any further analysis.
4. **SAVE** your new data set to module4.sav and the syntax to module4.sps

## **Exercise 4.2.**

Run the same exercise to view extreme values for corn. You may also run a **Frequencies** command to compare. 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 entered the missing values, run the **Explore** command again: you will notice the price average has changed! Although no outliers will be found, you may also run these exercises on producer and wholesale prices. Make sure to save your changes to the data.

**SPSS for Windows SAMPLE SESSION**  
**Module 5 - Basic Analysis (Time Series)**

**Short Course Training Materials**  
**Designing Policy Relevant Research and**  
**Data Processing and Analysis with SPSS for Windows 7.5**  
**3<sup>rd</sup> Edition**


**Department of Agricultural Economics, Michigan State University**  
**East Lansing, Michigan**  
**February 1999**

Basic analysis gives us the tools to interpret the data simply and quickly. Using elementary commands we can prepare graphs and tables allowing us to represent, analyze and diffuse market information rapidly and effectively. The purpose of this module is to teach you how to use these tools and to prepare reports.

The following exercises are some examples of how market information maybe presented and prepared by basic analysis. Although there are many ways to represent data and statistical analysis, the following exercises include: a table and graph of weekly percentage changes, aggregation to weekly and monthly levels, graphing mean price, and adjusting values for aggregated seasonal index in order to graph by agricultural season or production year.



### Percentage change from previous week

To be able to view percentage changes between weeks, we must first calculate the difference in prices and then compute the percentage. We can then represent these changes in a graph or a table. To calculate the difference and percentage for consumer prices, follow these steps:

0. If your syntax window is still open from the last module, close it so we can start a new one for this module. If the Data editor window is not already open, make it active select the **Module4.SAV** file, paste and run. We want to work on sorghum so you must set a filter like in module 4 (Run **Select Cases...** from the **Data** menu, **If... cer = 8**, and so on).
1. Create a new variable using the **Transform/Create time series...** command.
2. Select **p4c** from the list on the left and click on the  button next to **New Variables(s)** box. You will notice **p4c\_1=DIFF(p4c 1)** appear in the box. This will calculate the difference in prices between two consecutive cases.
3. Paste and run the command.
4. If you look at the Data Editor window you will notice the values are off by one case in time. This means we must adjust the values to the corresponding weeks. The way to do this is to use the same **Create time series** command using the **LEAD** function to replace the correct value in time (upwards one case in time - see explanation below for LAG and LEAD).

Open the dialog box. Change the function by clicking on the arrow facing down next to the **F**unction dialog box with the word **Difference** within. Choose **Lead**. First select **p4c\_1** to return to the column on the left. Then select **p4c\_1** from the list on






the left and click on the  button next to **New Variables(s)** box again (this will change the function). Change the name of the variable to **p4c\_1\_1** to **p4c\_d** (for difference) and click on the  button.

5. Paste and run the command.

**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.

**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.

Now we want to compute the percentage change between two weeks. We must then compute a new variable.

1. From the **Transform** menu select **Compute...**
2. For the **Target Variable:** enter **pctchg**
3. Click on  and enter a label (e.g. Percentage change).
4. Click on .
5. In the Numeric **E**xpression box, enter **(p4c\_d / p4c) \* 100**
6. If there is something next to the **IF...** button below, a condition still exists from module 4 that you must replace. Click on the **IF...** button, select the radio button next to *Include all cases*, click on .
7. Paste and run the command.

Now let's look at given percentage changes for three markets over four weeks. Another way to select specific data, without using solely the **Select** command (for filters or for 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 next procedure. It will be in effect until the next command that reads the data. Thus it may include data manipulation like in the example below (i.e. the transformations will apply for selecting data and for another command such as a table, graph, or regression and so on, commands that actually read the data and not only modify it; see the annex for further information). This command is not available through the menus so we must type it in the syntax editor. Make the syntax editor active and type and run the following command:

```
Temporary.  
Select if (sem = 20 or sem =21 or sem =22 or sem =23).  
Select if (mar = 3 or mar = 10 or mar = 41).
```

We have now temporarily selected data which we will depict in a table. Once the table will be executed the **Temporary** command will no more be in effect. Using **Tables** you can calculate various statistics and present them in a variety of ways that are completely under your control. Unlike other SPSS for Windows procedures, **Tables** allows you 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 (their denominator) so that they add to 100% across rows, columns, sub-tables, or whole tables.
- display up to 60 characters for variable labels and value labels.

To customize the table using the **Basic Table** command, go to:

1. **Statistics/Custom Tables/Basic Tables...**
2. Move **pctchg** to **S**ummaries:
3. Move **mar** to **D**own:
4. Move **sem** to **A**cross:
5. Click on **Layout...**
6. In the summary variable labels box, click the radial button next to **A**cross the top
7. **Continue**
8. **Titles...**
9. In the **T**itle box type: **Weekly price changes in percentages for three Malian markets over four weeks**
10. **Continue**
11. Paste and run

**Weekly price changes in percentages for three Malian markets over four weeks**




			Semaine			
			20	21	22	23
PCTCHG	Marché	BADALA	0	-1	4	2
		DIBIDA	-1	2	1	2
		KOUTIALA	2	3	4	3

Look at the table. We can see the changes in prices from one week to another. See how the changes vary by market! Let's now analyze producer markets using a graph. We can select two different producer markets by typing in the **Syntax Editor**:

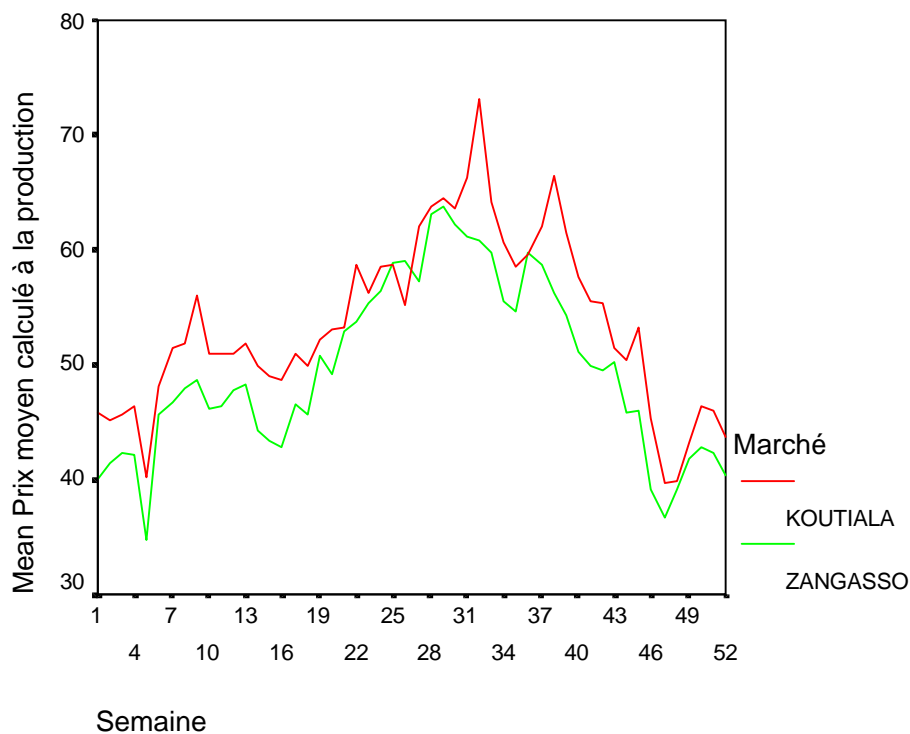
Temporary.  
Select if mar = 41 or mar = 75.

Now you can plot a graph by following these steps:

1. **Graphs/L**ine...
2. Click on the square next to **M**ultiple
3. Click on **Define**
4. Click in the radio button next to **O**ther **s**ummary function in the **L**ines **r**epresents box.

5. Now choose the **p4p** producer price variable in the left column and click on the  next to **Variable:** and 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 
9. 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
10. 
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 Output Navigator using the filename Output5.spo



### Exercise 5.1

Take a look at the consumer markets. To do so, you can select any of the consumer markets by typing the **Temporary (Select if)** command like shown above. For this example, choose markets 3-10-20-21-51-56-70 (do not run the command yet). Now paste the graph command but replace the producer price variable with the consumer price variable **p4c**. Paste and then run both commands together. What do you see?

We can observe that prices follow about the same trends over time. And we can also notice how all the curves are bunched together. One way to avoid this is to separate the curves using the **Split file** command like so:

1. Go to **Data/Split File...**
2. Click the radio button next to **Compare groups**  
You may also choose **Organize output by groups** which will perform the same function as for **Compare groups** but the difference is in the output: in this example, the split-file groups are presented together for comparison purposes here. Organizing output by groups will display separate output for each split-file group, not as graphs displayed together in one single output.
3. Choose **mar** from the left column and put in the **Groups based on:** box
4. Click on the radio button next to **File is already sorted**
5. Paste but do not run the command

Now paste the graph command. But we want to view each graph separately so make sure to select the box next to **simple**. Use the following steps:

1. **Graphs/Line...**
2. Click on the square next to **Simple**
3. Click on **Define**
4. Click in the radio button next to **Other summary function** in the **Lines represents** box.
5. Now choose the **p4c** consumer price variable in the left column and click on the **▶** next to **Variable:** and the mean value of producer prices will be graphed.
6. Choose **sem** for the **Category Axis:**
7. Click on **Options...**
8. 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
9. **Continue**
10. Paste the command
11. Go to the syntax editor and look at the commands. The **Split file** command will remain until we remove this function. As we wish to affect this command to the graph function only, we may either type **SPLIT FILE OFF.** on a new line following the end of the graph command in the Syntax editor or select **Analyze all cases, do not create groups** from the **Split file** command menu and paste it to the Syntax editor.

Now pick the commands to run with your mouse, by selecting **Temporary, Select if** to the **Split file off.** command. Run the command and look at the new charts produced. See how each individual sorghum market has its own graph: one for producer, one for assembly and one for retail markets.

### Aggregation to monthly levels and graphing mean price

Another way to represent data or group data for analysis is by aggregating data to other levels. For example, if we wanted to view a mean price for Bamako which is composed of many markets, we can look at the prices by “localité” which groups markets together allowing us to obtain a mean value for the area. Lets look at the Bamako mean price.

1. Go to the Syntax editor and type:  
Temporary.  
Select if loc = 3.
2. Then paste the command for a simple line graph for consumer prices by week.

**Note** For those of you who are comfortable in 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 modules of the sample session. You will find this becomes quicker as you become familiar with the syntax. Remember to change the variables in the command and be careful to copy the correct choice of the three possible graph options (see annex for further information on these options).

## Exercise 5.2

Repeat steps 1 and 2 but replace graphs by week for graphs by month (i.e. mois). Practice using the **Split file** command by localit  (loc). Remember to type and run SPLIT FILE OFF. once you have finished using the **Split file** function (you may also paste this from the menu). As well, you may practice these commands choosing other commodities such as maize for example. Save any graph that is of interest to you. The /Missing Report portion of the command is unnecessary as it only displays groups defined by missing values which is of little interest here.

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

The following sub-section will allow us 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. But first, we should compute a variable by month and year together for identifying each period of time represented in the graphs. Follow these steps:

1. From the **Transform** menu select **Compute...**
2. For the **Target Variable:** enter **date\_m**
3. Click on **Type&Labels...** and enter 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. Type in the Syntax editor **FORMATS date\_m (MOYR8).**
9. Run the compute and formats commands together.

Follow these steps to compute the season month and production year variables:

1. From the **Transform** menu select **Compute...**
2. For the **Target Variable:** enter **c\_month**
3. Click on **Type&Labels...** and enter a label (e.g. Production month).
4. Click on **Continue**

5. In the Numeric Expression box, enter a **mois + 2**
6. Click on **If...**
7. Select the radio button for Include if case satisfies condition:
8. Enter the statement **mois <= 10**
9. Click on **Continue**
10. Paste and run the command.
11. Repeat steps 2 through 10 replacing the previous information with the following:

Numeric Expression	If... Statement
<b>mois - 10</b>	<b>mois &gt;= 11</b>

12. We must create a variable as well for the production year **c\_year**. Repeat steps 1 through 10 but for **c\_year** (use label 'Production year or agricultural seasons').
13. Repeat steps 2 through 10 for the two following statements:

Numeric Expression	If... Statement
<b>annee + 1</b>	<b>c_month &lt;= 2</b>
<b>annee</b>	<b>c_month &gt;= 3</b>

14. Run these commands. Now we want to add **Value Labels** for each of the new months and production years. To do this we can use the **Define Variable...** command from the Data menu or in this case, we will type directly in the Syntax editor the following:

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'.
EXECUTE .
```

15. Now run this command.
16. **SAVE** the data. **IT IS VERY IMPORTANT THAT YOU DO NOT FORGET TO SAVE THE DATA AS IT WILL BE USED IN THE MODULES AHEAD!** Use the filename **Module5.sav**, paste and run.
17. Save your syntax file as **Module5.sps** and close it.

### Exercise 5.3

Now let's see what the new data looks like. Graph the consumer price by production month (i.e. **c\_month**) and compare with the graphs computed beforehand using 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!

**SPSS for Windows SAMPLE SESSION**  
**Module 6 - Seasonal Analysis (Time Series)**

**Short Course Training Materials**  
**Designing Policy Relevant Research and**  
**Data Processing and Analysis with SPSS for Windows 7.5**  
**3<sup>rd</sup> Edition**

**Department of Agricultural Economics, Michigan State University**  
**East Lansing, Michigan**  
**February 1999**

Seasonal analysis of weekly, monthly and yearly historical price or quantity data series for food and agricultural commodities constitute an important part of market analysis (see Goetz and Weber 1986, IDWP 29, Michigan State University, for further details time series analysis). This analysis seeks to sort out repetitive 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.

### **Merging and aggregating files and data**

As we want to perform seasonal analysis on the Market Information System data from 1982 to 1995, we need to merge data from two different files: pq\_8995.sav (now module5.sav) and p\_8289.sav. For the purpose of this module, will select data for Bamako and sorghum.

1. Start a new syntax window, and open module5.sav. Paste and run.
2. Verify all filters and split files functions are off.
3. Now, as in past modules, please select out data for Bamako and sorghum by using **Data/Select Cases...** and specifying **loc = 3** and **cer = 8**. Make sure to click on the radio button next to Deleted in the Unselected cases are display box and then paste and run the command.

Different from the past module, we will be using the **aggregate** command to aggregate data to a new level, in this case the mean for Bamako. To do so, use the following steps:

1. Go to **Data/Aggregate...**
2. From the left column, select **annee mois date\_m c\_month c\_year loc** and put the variables in the **Break Variable(s)** box
3. Select the consumer price variable **p4c** and put in the **Aggregate variable(s)** box.
4. Click on **Name & Label...**, enter "Mean consumer price" as the label, and change the name from **p4c\_1** to **p4c**.
5. **Continue**
6. Click on **File...** and change the name of the file to AGGBKO1.sav
7. **Save**
8. Paste and run the command
- 9a. Now open the data file p\_8289.sav using **File/Open...**
- 9b. **Select sorghum and Bamako** in the same manner as above for the 1989 to 1995 data.

- 9c. We do not need to aggregate the data in this file as we can simply save this data in its present format. To do this, you must:
10. Go to **File/Save As...**
11. Give it the file name AGGBKO2.sav and paste the command
12. Look in the Syntax editor, you will notice the save command written followed by the word */compressed*. But in fact, the **Save** command syntax (which you can look up or search in the SPSS help tools) looks like:

```
SAVE OUTFILE=file
[/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 obtain many of the possible syntax functions from the menus for many of the commands, we can in any case type the various functions, add syntax, in the Syntax editor as SPSS will recognize and execute these commands. To do so, first:

13. So above the line with */compressed* in the syntax file (after the line with Save outfile...), type the following:  
     /KEEP annee mois date\_m c\_year c\_month loc p4c
14. Run the save outfile command.

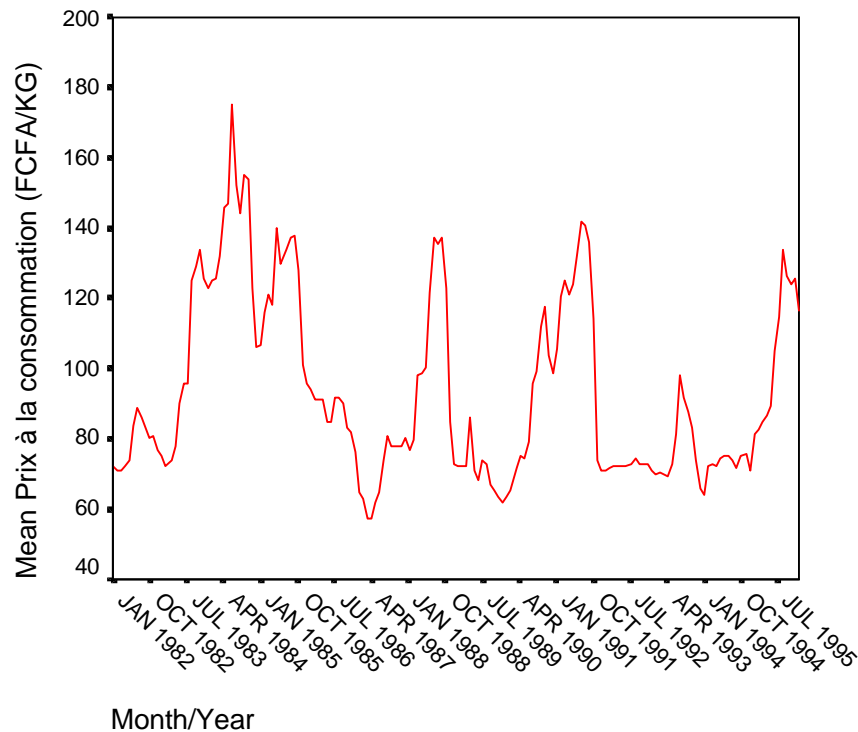
Now that both files are similar in data, we are ready to merge them together.

1. If not already active, paste the command and open the aggbko2.sav file
2. Go to **Data/Merge File/Add Cases...**
3. Choose aggbko1.sav and click on **Open**
4. Paste and run the command
5. Save the file to aggbko3.sav. Paste and run the command and then save your new syntax file to Module6.sps

Take a look at the data. You will notice that price data for the years 1982 to 1989 are now joined to those of 1989 to 1995! Now lets 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 Output navigator 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** simultaneously. 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 Output6.spo (you can eliminate previous output by deleting the selections from the left column).

## Computing moving average and graphing nominal prices

A moving average helps in 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 value observation is larger or smaller, the averaging procedure will bring that value more in line with the other values in the series and fluctuations are thus eliminated. Removing these short term fluctuations in the time series, the analyst is allowed to focus on important longer term patterns like cycles and trends.

To compute the moving average, we need to use the **Create time series** command using the following steps. This command will produce a new series as a function of an existing series, in this case prices.

1. Create a new variable using the **Transform/Create time series...** menu.
2. We need to use the Moving average function. You can change the function by clicking on the arrow facing down next to the **Function** drop-down box with the word **Difference** within. Choose **Centered moving average** (see explanation below).
3. Then just below, you must enter 12 in the **Span:** box (representing the 12 months)

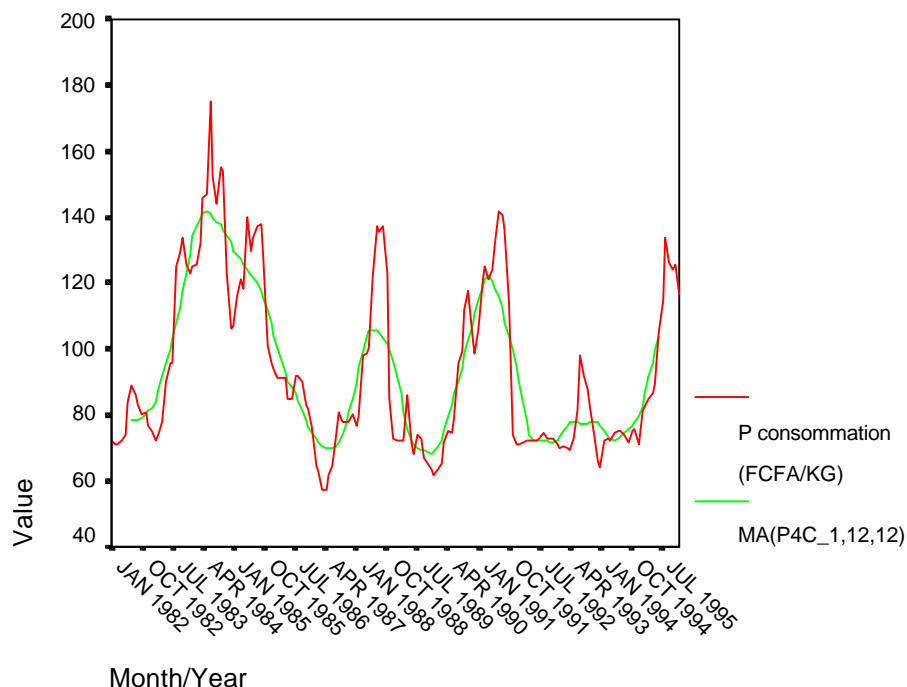
4. Select **p4c** in the column on the left and click on the  button next to **New Variables(s)**. Change the name of the variable to **p4c\_1** to **p4c\_ma** (for moving average) and click on the **Change** button.
5. Paste and run the command.
6. You may also give the variable a label by typing in the syntax:  
VAR LAB p4c\_ma "Centered price moving average".
7. 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.

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, paste and run the command

Now view the graph: see how the moving average presents the overall trend and eliminates the short term fluctuations!



## Seasonal index

Another good indicator of seasonal analysis is seasonal index. Seasonal movements in price and quantity time series are particularly prevalent in agricultural crops. Ecological processes and cycles exert strong influences on agricultural production processes and are reflected via quantities in the behavior in prices over time. Seasonal price movement can reflect storage costs and can even effect non-storable commodities. The index can be calculated thanks to the moving average: the seasonal index is the nominal price divided by the moving average. To calculate the seasonal index, use the following steps:

1. From the **T**ransform menu select **C**ompute...
2. For the **T**arget Variable: enter **season**
3. Click on **T**ype&Labels... and enter a label (e.g. Seasonal index).
4. Click on **C**ontinue
5. From the Numeric **E**xpression: box, enter **p4c / p4c\_ma\*100**
6. Paste and run the command.

Now graph the seasonal index with the nominal prices by calendar month (**date\_m**) or by production or marketing month (**c\_month**) using the same steps for the moving average graph above. For example, the syntax for this command should look like this:

```
GRAPH  
/LINE(MULTIPLE)=VALUE(SEASON p4c) BY date_m.
```

Take a look at the graph: see how the index is centered around 100? And do you notice a periodic trend (e.g. annual peaks)? Further research would show that the seasonal trend-prices start to decrease a little in September and October when people have an idea of whether it is a good year or not. At the harvest in November, December prices drop. Prices increase throughout the dry season from January to June and reach their peak in the rainy season in July and August. You may save the output as output6.spo (you are appending the new output to the output6.spo file).

This would be a good time to save the data as well. It is always good to give a file a name which helps us define the type of data we are working with or that is contained in the file.

1. Save outfile as **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, it is important to aggregate data to other levels for analysis. In this sub-section, we will aggregate from a weekly to a monthly level and see the distribution of the standard deviation. First lets aggregate by production month using the following steps:

1. Go to **D**ata/**A**ggregate...
2. From the left column, select **c\_month** and put it in the **B**reak Variable(s) box
3. Select the variable **season** and put in the **A**ggregate variable(s) box, then give a label such as “Mean seasonal index”

4. Select the variable **season** and put in the Aggregate variable(s) box for a second time. Click on **season\_2 = MEAN (season)** within the box, once you have given it the label "Seasonal index SD", click on **Continue**
5. Click on **Function...** and select the radio button next to **Standard deviation**
6. **Continue**
7. Click on **File...** and change the name of the file to AGGSEAS.sav
8. **Save**
9. Paste and run the command

Now open the file `aggseas.sav` (paste and run the command) and take a look at the Data editor window: 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. 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:

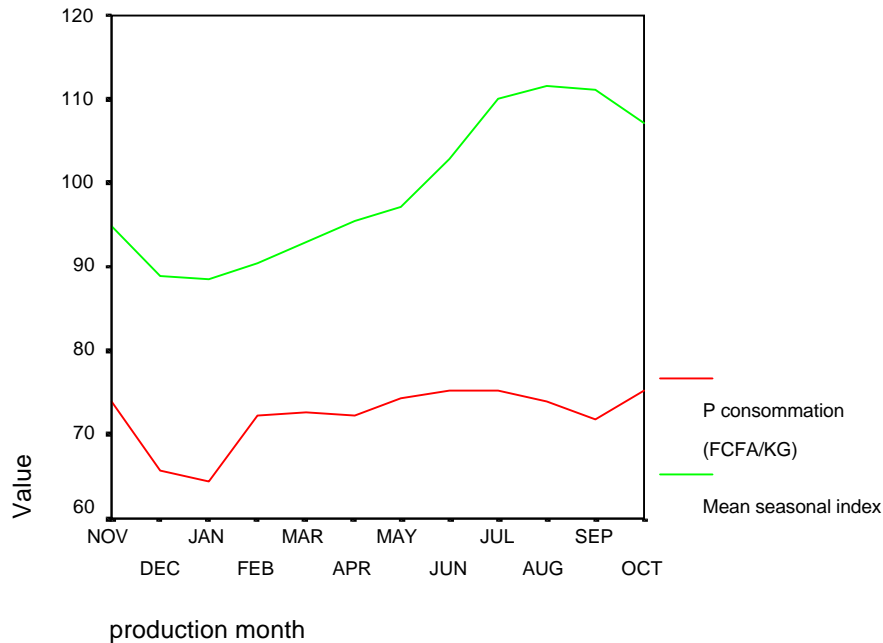
```
COMPUTE SDPLUS = season_1 + season_2 .
EXECUTE.
COMPUTE SDMINUS = season_1 - season_2 .
EXECUTE .
```

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. Viewing the graph, we can observe how the trends for all three follow closely one another.

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. Lets take 1994 production year for this example.

0. Make the seasonal data active by opening the `season.sav` file (paste and run the command). Select the data for **c\_year = 1994** and then merge the data from the aggregated seasonal index with the standard deviation file, using the **Merge** command. We may do so following these steps:
  1. Go to **Data/Merge File/Add Variables...**
  2. Select the filename `aggseas2.sav` and click on **Open**
  3. Select **annee mois date\_m loc season season\_2 sdminus sdplus** from the list under **New Working Data File:** and click on **◀**
  4. Check the box next to **Match cases on key variables in sorted files**
  5. Click on radio button next to **External file is keyed table**
  6. Select **c\_month** from the **Excluded Variables:** list
  7. Click on **▶** next to **Key Variables:** (bottom, right)
  8. Paste the command
  9. Click on **OK**, then **NO**
  10. Select and run the command. Be sure to include `EXECUTE.`

Now lets 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 **Define**
5. Put **p4c** and **season\_1** in the **Lines represent** box.
6. Click on the radio button next Variable: in the **Category Labels** box
7. Select **c\_month** and put it as the choice of the variable in the **Category Labels** box
8. 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! Why is this? Save a copy of the chart for further analysis. 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 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).

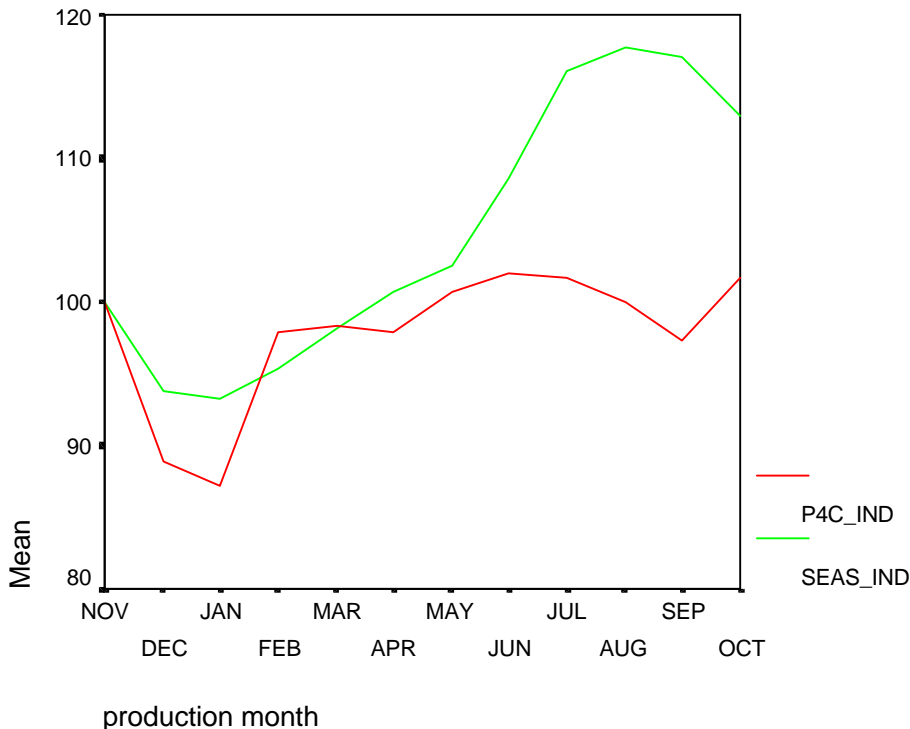
What are the consumer price and seasonal price values for 1994? If we look up the Data editor window we can see the values for the month of November which are 73.85 and 94.91 respectively. Now to get the same base value or 100 in November we must divide all of the values throughout the production year 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: enter **p4c\_ind** and enter a label if you choose
3. In the Numeric Expression box, enter **p4c/73.85\*100**
4. Paste the command.

5. Return to the menu **Transform/Compute...**
6. For the **Target Variable**: enter **seas\_ind** and enter a label if you choose
7. In the **Numeric Expression** box, enter **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  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, we can observe that 1994 was an odd year as it did not follow the seasonal trend. Prices were, for one particular month, higher than the seasonal average but then remained pretty much around the same price. Can you explain? 1994 was the period of the devaluation. 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 interpret results better, it would be good to bring in production data, do formal interviews and so on. Write your ideas in your Output navigator and compare with your colleagues. Don't forget to save your results (in output6.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. Enter 1982 as the year for the First case is: box
5. Click on Ok. (Note: The command will be posted in the Output navigator).

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

1. Go to **Statistics/Time Series/Seasonal Decomposition...**
2. Choose the consumer price **p4c** in the left column and put it under Variable(s):
3. In the Moving Average Weight box, select the radio button next to **Endpoints weighted by .5** (see explanation of choice of models below).
4. Paste and run the command.

### **Two types of models to choose from: Multiplicative or Additive.**

#### **Multiplicative Seasonal Adjustment.**

The seasonal component is a factor by which the seasonally adjusted series is multiplied to yield the original series. In effect, 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.**

The seasonal adjustments are added to the seasonally adjusted series to obtain the observed values. This adjustment attempts to remove the seasonal effect from a series in order to look at other characteristics of interest that may be "masked" by the seasonal component. In effect, Trends estimates seasonal components that do not depend on the overall level of the series. Observations without seasonal variation have a seasonal component of 0.

Look at the results in the Output navigator and then in the Data editor. The seasonal model is:

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

And for example, for January of 1982 we have:



$$\begin{aligned} P(72) &= \text{STC}(80.101) \times \text{SAF}(.893) \times \text{ERR}(1.006) \text{ and} \\ \text{SAS}(80.593) &= P(72)/\text{SAF}(.893) \end{aligned}$$

The various seasonal variables computed by SPSS seem to correspond to the model. Make sure to save the new data `season.sav` file.

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 by graphing 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 **T**ransform/**C**ompute...
2. Enter **nov** as the target variable.
3. In the Numeric **E**xpression box, enter a **p4c**
4. Click on **If...**
5. Select the radio button for Include if case satisfies condition:
6. Enter the statement **mois = 11**
7. Click on **C**ontinue
8. Paste and run the command.
9. In the Syntax editor, type  
Temporary.  
Select if c\_year > 1982.
10. Now use the **A**ggregate command and go to **D**ata/**A**ggregate...
11. From the left column, select **c\_year** and put it in the **B**reak Variable(s) box
12. Select the variable **nov** and put in the **A**ggregate variable(s) box.
13. Click on **F**ile... and change the name of the file to AGGNOV.sav
14. **S**ave
15. Paste and run the command with the Temporary 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 **D**ata/**M**erge File/**A**dd **V**ariables...
2. Select the filename aggnov.sav and click on **O**pen
3. Select **c\_year** from the **E**xcluded Variables: list 
4. Check the box next to Match cases on key variables in sorted files
5. Click on radio button next to **E**xternal file is keyed table
6. Select **c\_year** from the **E**xcluded Variables: list
7. Click on  next to **K**ey **V**ariables: (bottom, right)
8. Paste the command
9. Click on **O**K, then on **N**O
10. Select and run the command.

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

1. From the **T**ransform menu select **C**ompute...
2. For the **T**arget Variable: enter **p4c\_ind** and a label of your choice
3. From the Numeric **E**xpression: box, enter **p4c / nov\_1\*100**
4. Paste and run the command.
5. 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 **Split file** command by production year and then graph as with 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 (Module6.sps) to add the syntax from the most recent exercises in the module. Close it so we can open a new one for the following module.

### **Exercise 6.1**

This will be an exercise which may take time as it calls for you to repeat all of the different types of seasonal analysis performed in this module. 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 module, but of course you may return to the module 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 window 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 corn will be used in exercises of modules ahead.

- i) First you need to aggregate and merge the same two data files **p\_8289.sav** and **module5.sav**. Select Bamako and maize such as was done in the module. 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 corn 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...)
- v) 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).

**SPSS for Windows SAMPLE SESSION**  
**Module 7 - Trends (Time Series)**

**Short Course Training Materials**  
**Designing Policy Relevant Research and**  
**Data Processing and Analysis with SPSS for Windows 7.5**  
**3<sup>rd</sup> Edition**

**Department of Agricultural Economics, Michigan State University**  
**East Lansing, Michigan**  
**February 1999**

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 contains a series of monthly consumer sorghum prices for Bamako from 1982 to 1995. For the purpose of this module, we will focus on a shorter period of four years, from January 1992 to December 1995.

Thus you will need to create a new file from the price series for the 1992-1995 period. As you have done before, open the **Season.sav** file (paste and run the command), use the **Select** command to eliminate the data before 1992 (Hint: `annee ≥ 1992`), paste and run the command. Save this new file under a new name (call it **Trends.sav**), in order to keep the data from 1982 to 1995 in the original file. Now we can study sorghum price trends in Bamako from 1992 to 1995 using a regression analysis. Use the following steps:

1. **Statistics/Regression/Curve Estimation...**
2. Select the consumer price variable **p4c** in the left column and put it the **Dependent(s):** box
3. Click on the radio button next to **Time** in the **Independent** box
4. 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.
5. At the bottom of the dialog box, check the radio box next to **Display ANOVA table**
6. Paste and run the command.

Among other things, the results window provides us with:

- The adjusted R Square. This statistic gives us an empirical measure of the strength in the linear relation that can exist between the dependant and independent variables in the sample. It measures the proportion of the variation of the dependant 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%;

- the coefficients of regression (**B**) that gives the linear relation between the dependant 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;



1. Go to **Statistics/Regression/Linear...**
2. Put **p4c** in the Dependent: box
3. Put **time** in the Independent(s): box
4. **Statistics...**
5. Click on the Durbin-Watson box in the Residuals box
6. **Continue** then paste and run both the command

Go to the Output navigator. You will notice that the coefficients are the same as those of the previous values and the Durbin-Watson (D.W.) coefficient is equal to 0,20. When we compare this coefficient to the corresponding 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:

**Model Summary<sup>b</sup>**

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Durbin-Watson
1	,665 <sup>a</sup>	,443	,431	13,380	,198

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

1. Go to **Statistics/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 both the command

Note. Notice that the command begins the analysis by the “ordinary” method of least squares, the same syntax which used at the beginning of the module.

The Output navigator reveals that the D.W. coefficient has increased to 1,37 but it remains in the interval were the positive auto-correlation is considered. However, 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, long periods of time with many observations have more difficulties in satisfying the theoretical values of D.W. which increase with the number of observations. Even more, during a long period, we are generally confronted with trend reversals, making the relation between the dependant and the independent variables problematic. Thus, it is necessary in this case to subdivide the period on objective facts. For example, we could chose the devaluation 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. Select the data prior to 1994 (`annee < 1994`) by filtering the non-selected observations that will be used later for the second period (described above);
2. Calculate the regression with the **Autoregression** command by executing the five steps described above.

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

1. Select the proper period the way we selected the 1993 period;
2. Since the time variable for this period starts with 25, you must create a new variable `time_2`, starting by 1:
  - i. From the **Transform** menu select **Compute...**
  - ii. For the **Target Variable:** enter **time\_2** (add a label of your choice)
  - iii. From the **Numeric Expression:** box enter: **time - 24**
  - iv. **If...** and select the radio button next to **Include if case satisfies condition:**
  - v. Enter: **annee > 1993**
  - vi. **Continue**
  - vii. Paste and run the command
3. 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. This is caused by the analysis procedures which round numbers upwards, which can render a number above 1, a number that can theoretically 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%, 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 Module 6). We would need to “deseasonalize” **p4c** for example and use the new **sas\_1** variable as the dependant 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 Module7.sps and close it.

### **Exercise 7.1.**

Open the file with the maize price data you prepared in exercise 6.1 in the last module. Then repeat the various trend analysis (through regression) you have just learnt in module 7. Interpret the results.

**SPSS for Windows SAMPLE SESSION**  
**Module 8 - Real Prices and Price Graphing and Tables (Time Series)**

**Short Course Training Materials**  
**Designing Policy Relevant Research and**  
**Data Processing and Analysis with SPSS for Windows 7.5**  
**3<sup>rd</sup> Edition**

**Department of Agricultural Economics, Michigan State University**  
**East Lansing, Michigan**  
**February 1999**

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 learnt in the past module 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 module, we will create real prices by importing price index data for 1988 to 1995 from Quattro Pro (Spreadsheet software) as it was explained in module 3.

**Import consumer price index from Quattro pro to compute real or deflated prices**

As we want to obtain data from a Quattro pro file, the extension will be \*.wk1 which is equivalent to \*.w\*. Follow these steps to import the price index data:

1. **File/Open...**
2. Go to **Files of type:** and select **Lotus (\*.w\*)**
3. Click on the file index.wk1 which will have appeared above and click on paste
4. An options window will open, click on OK.
5. Run the command

Make the Data editor is active. You will see three columns: the first is the year, the second is the month and the third is the price index which we want to keep. You may save this file, as it is always recommended to do so but for the purpose of this module, we will simply perform some data manipulation.


1. Click on “c” to select to whole price index column.
2. We want to copy this column to another file so go to **Edit/Copy**
3. **File/Open...**
4. Change the three variable names from **a** to **annee**, **b** to **mois** and **c** to **pindex**. Then save the file to **pindex.sav**
5. Paste and run the command
6. Go to **Files of type:** and select **SPSS (\*.sav)**

7. Select **season.sav** then paste and run the command
8. Make the Data editor window active
9. Go to the column after the season variable column
10. Go down to cell number 73 (where data starts at January 1988)
11. Now paste the data from the price index file here by going to **Edit/Paste**
12. The same values will appear from cell 73 downwards while the values upwards are replaced by `sysmis`. You will also notice that the column has a new title called : VAR00001 which you must change. Double click on VAR00001: the **Define Variable** dialog box will appear.
13. Replace the **Variable Name:** with **pindex**
14. Click on **Labels...** and enter Consumer price index as the **Variable Label:**
15. **Continue**
16. Click on Ok

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

1. From the **Transform** menu select **Compute...**
2. For the **Target Variable:** enter **real** and a label of your choice (e.g. real prices)
3. From the **Numeric Expression:** box, enter **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 the two prices compare by graphing the nominal and real prices together!

1. As 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  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 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 (Hint: you do not have to import the file only copy the pindex variable to the price data on maize). 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 in the Basic Analysis module, 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). As the data is only available after 1993, we should also select out the data accordingly. Follow these steps:

1. **File/Open...**
2. Select **Module5.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. Enter **mar = 75 and cer = 8 and annee >= 1993** in the box on the right
6. Click on **Continue**
7. Select the radio button next to **Deleted**
8. Paste and run the command.


So we can fit prices and quantities on the same graph we must first convert the quantity data from kilograms to tons. To compute the data into tons, follow these steps:

1. From the **Transform** menu select **Compute...**
2. For the **Target Variable:** enter **q6\_t** and a label of your choice (e.g. Quantities bought in tons)
3. From the **Numeric Expression:** box, enter **q6 / 1000**
4. Paste and run the command (and save the data under a new name e.g. zan9395.sav).

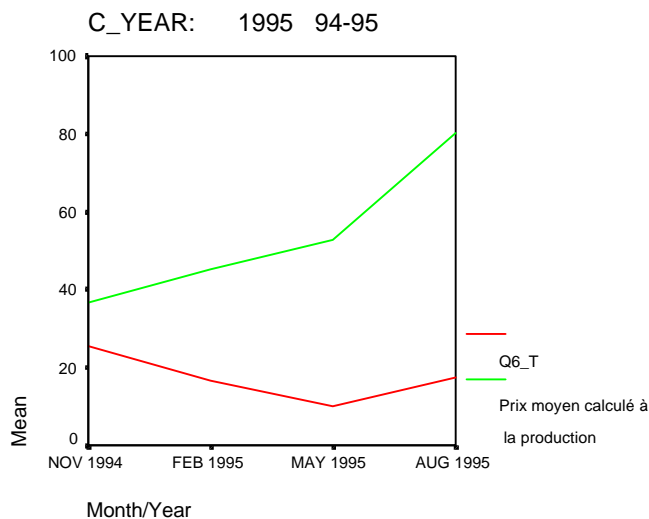
Now you are ready to plot a number of graphs to look at and analyze the data. Lets try to graph the quantities against prices by quarter and the by production year by following these steps:

1. As we want to graph by quarter we must first select four months using the Temporary command. Type but do not runt the following text in the Syntax editor:  
Temporary.  
Select if c\_month = 1 | c\_month = 4 | c\_month = 7 | c\_month = 10.

P.S. For those who have become familiar with copying and pasting the commands within the Syntax editor, you may do so at all times. I invite those who are not sure to try copying and pasting the commands. Bypassing the menu system is quicker but make sure to select the correct syntax which corresponds to the sample session, like the following graph:

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  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 with the **Temporary** statement
12. Repeat steps 2 through 11 but use the **Split File** command to take a closer look at each production year (remember to type/select from the menu **SPLIT FILE OFF**). The syntax for this last statement should look like this:

```
SPLIT FILE
  BY c_year.
GRAPH
  /LINE(MULTIPLE)=MEAN(q6_t) MEAN (p4p) BY date_m.
SPLIT FILE OFF.
```



What can we say observing the graphs? The relationship between price and quantity is unclear but looking at each of the marketing years, it seems 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. hibernage - the rainy season from June to September), the 93/94 season is a good example. The reason for the drop is due to soudure. But we can also observe increases in quantities at harvest. Save your charts in the Output navigator to Output8.spo. It has been some time since we saved your Syntax editor documenting all the work you have done! Save the syntax to session8.sps.

As we have done in earlier modules, 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 be choosing the 1994 calendar year and we will look at monthly price changes. As you have become familiar with using these commands, make the **module5.sav** file active, then you will need to create two separate **Select If** commands like so: **annee = 1994 or c\_year = 1994** and the second **cer = 8** (make sure to select the radio button next to **Deleted**) and then save the data under **TABLE94.sav** filename (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 not, 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\Module5.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 and 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**
7. **Save**
8. Paste and run the command
9. Now make the **aggtab94.sav** file active.
10. Create a new variable using the **Transform/Create time series...** command.
11. Select **p4c\_1** from the list on the left and click on the **▶** button next to **New Variables(s)** box. You will notice **p4c\_1=DIFF(p4c\_1 1)** appear in the box.
12. Change the name of the variable to **p4c\_1\_1** to **p4c\_p** (for price change) and click on the **Change** button. Paste and run the command.
13. Always recommended, if you want to give the new variable a label e.g. Price change, you can do so by double clicking on the name in the Data editor window.

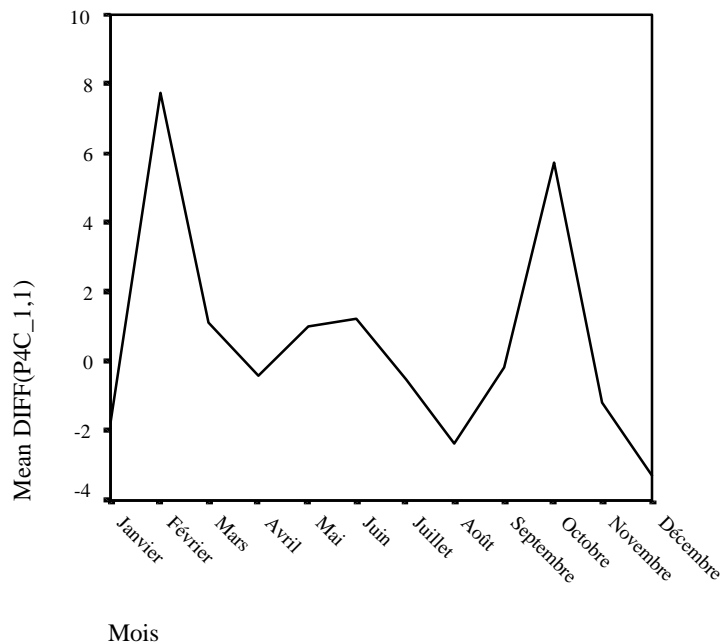
Now we are ready to graph the price changes for any market for any commodity during the 1994 year. For an example, let's select sorghum for Bamako by typing in the Syntax editor:

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

Now use the **Graph** command for 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 our ideas in the Output navigator 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 and mois = 12.
```

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

1. **Statistics/Custom Tables/Basic Tables...**
2. Move **p4c\_1** to **S**ummaries: and move **p4c\_p** to **S**ummaries:
3. Move **mar** to **D**own:
4. Move **cer** and then **mois** to **A**cross:
5. Click on **Layout...**
6. In the summary variable labels box, click the radial button next to **A**cross 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 Output navigator: 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, namely in market information systems, is to calculate growth rates for production and other types of variables.

1. **File/Open...**, and go to **Files of type:** and select **Lotus (\*.w\*)**
2. Click on the file **maliprod.wk1** which will have appeared above and paste
3. The options dialog box will appear, click OK. Then run the command
4. In the Data Editor, type in the different variables names like so: Campaign, year, productn (tons) - the commodities are millet, sorghum, maize, rice and fonio.
5. Now let's calculate the growth rate using the **Curve Estimation** command. Go to **Statistics/Regression/Curve Estimation...**
6. Select the consumer price variable **productn** in the left column and put it the **Dependent(s):** box
7. Click on the radio button next to **Time** in the Independent box
8. In the **Models** box, you will see **Linear** is already chosen. Select the **Growth** and **Exponential** models as well.
9. At the bottom of the dialog box, check the radio box next to **Display ANOVA table**
10. 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 **Module8.sps** and close the window.

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:

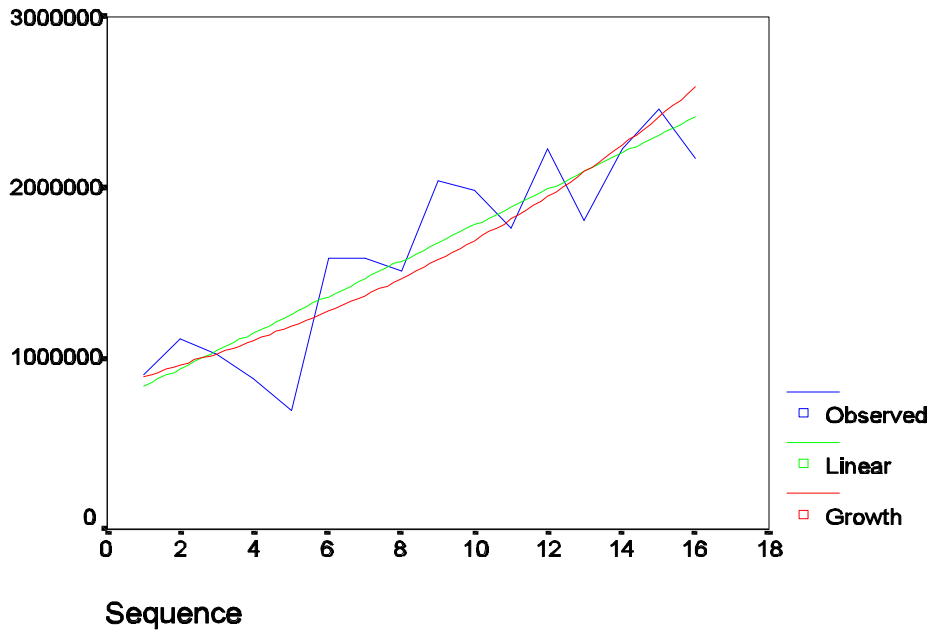
	DF	Sum of Squares	Mean Square
Regression	1	1,7054863	1,7054863
Residuals	14	,5883807	,0420272

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

### PRODUCTN



**SPSS for Windows SAMPLE SESSION**  
**Module 9 - Margin Analysis (Time Series)**

**Short Course Training Materials**  
**Designing Policy Relevant Research and**  
**Data Processing and Analysis with SPSS for Windows 7.5**  
**3<sup>rd</sup> Edition**

**Department of Agricultural Economics, Michigan State University**  
**East Lansing, Michigan**  
**February 1999**

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 detailed information). 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. Production and input costs, compensation for efforts and storage costs incurred, transfer and processing costs, risk and so on. Marketed margins allow the analyst thus to assess the extent of which costs and profits are normal and 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. Thus, 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**), and 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 1993 onwards.

### **Data manipulation, aggregating and merging files for margin analysis**

What you will do in the following lesson 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. Make the **module5.sav** file active (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. Enter **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. Go to **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. Filter off, then change the filter by repeating steps 2 through 7 to select **loc = 33, cer =8 and annee >=1993**
20. Repeat steps 8 through 12 replacing the variables in the **Aggregate variable(s)** box by **pr10**, giving it the name **pr10\_kou** and a label such as "Mean most common 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. Filter off, then change the filter by repeating step 19 to select **loc = 3, cer =8 and annee >=1993**
25. Repeat step 20 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...**
2. Select *marga.sav*, paste and run the command
3. Go to **Data/Merge File/Add Variables...**
4. Select the filename *marb.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 c\_year c\_month sem** from the **Excluded Variables:** list
8. Click on **▶** 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 under *margin.sav* as the filename




You will have noticed in the Data Editor window for some cases the same week (**sem**) is repeated twice because the two cases come from separate files. So we need to merge them to eliminate the sysmisses and false duplicates. We can do this by running an **aggregate** command here. Type and run in your syntax window the following commands:

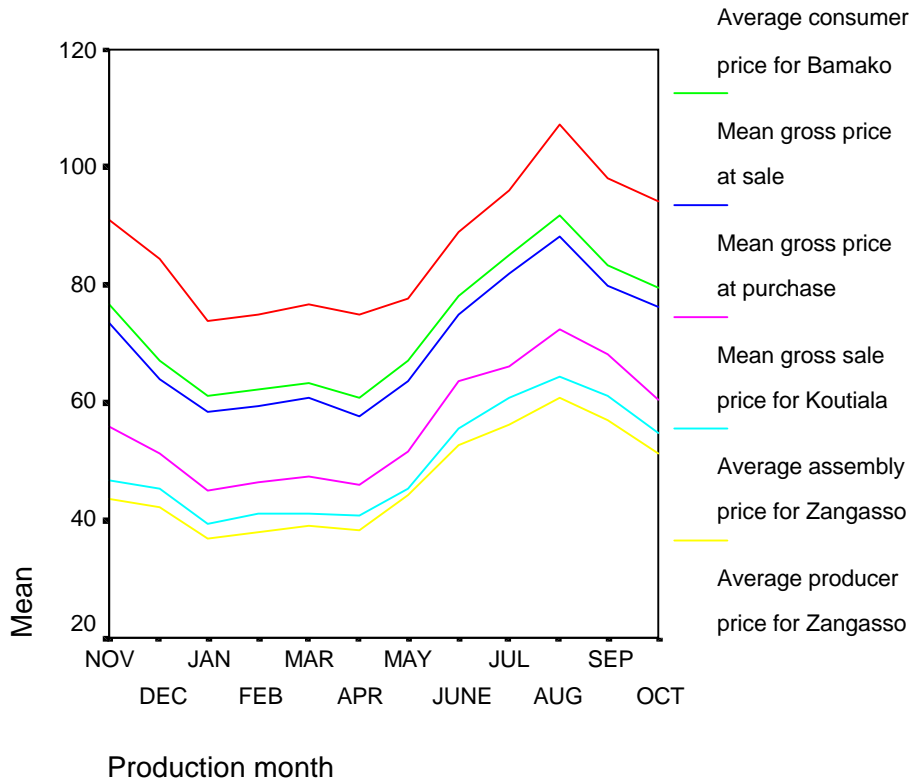
```
AGGREGATE
  /OUTFILE=*
  /BREAK=cer c_year sem
  /annee = MEAN(annee)
  /mois = MEAN(mois)
  /c_month = First(c_month)
  /p4p_zan "Average producer price for Zangasso" = MEAN(p4p_zan)
  /pr8_zan "Average assembly price for Zangasso" = MEAN(pr8_zan)
  /pr10_kou "Mean most common gross sale price for Koutiala" =MEAN(pr10_kou)
  /p4c_bko "Average consumer price for Bamako" = MEAN(p4c_bko)
  /gal3_bko "Mean gross price at purchase" = MEAN(gal3_bko)
  /gv16_bko "Mean gross price at sale" = MEAN(gv16_bko).
SORT CASES BY
  c_year (A) c_month (A) sem (A) .
SAVE OUTFILE='C:\Sample\margin.sav'
  /keep cer annee mois c_year c_month sem p4p_zan pr8_zan pr10_kou p4c_bko gal3_bko gv16_bko
  /COMPRESSED.
```

### Graph to verify price logic

As it was explained in module 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  next to **Category Axis:**
7. Click on 
8. Click on the radio box next to **Display groups defined by missing values**
9. 
10. Paste and run the command



Take a look at the graph (printed on next page), the curves allow us to see if any of them cross over another curve. This would mean that the price logic would not be respected e.g. a producer price higher than an assembly price in Zangasso. It is not the case so we may pursue further analysis. The graph also allows us to see 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)

Module 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 **T**ransform menu select **C**ompute...
2. For the **T**arget Variable: enter **prodshare** and a label of your choice (e.g. Producer share)
3. From the Numeric **E**xpression: box, enter **p4p\_zan / p4c\_bko**
4. Paste and run the command

- 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
<b>ass1shar</b>	pr8_zan / p4c_bko
<b>ass2shar</b>	pr10_kou / p4c_bko
<b>wh1shar</b>	ga13_bko / p4c_bko
<b>wh2shar</b>	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 enter **wh2\_shar wh1\_shar 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:

- From the **T**ransform menu select **C**ompute...
- For the **T**arget Variable: enter **marg1** and a label of your choice (e.g. Collection)
- From the **N**umeric **E**xpression: box, enter **pr8\_zan - p4p\_zan**
- 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
<b>marg2</b>	<b>Assembly</b>	<b>pr10_kou - pr8_zan</b>
<b>marg3</b>	<b>Rural Gross</b>	<b>ga13_bko - pr10_kou</b>
<b>marg4</b>	<b>Urban Gross</b>	<b>gv16_bko - ga13_bko</b>
<b>marg5</b>	<b>Retail</b>	<b>p4c_bko - gv16_bko</b>

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? The variations are very numerous and complex to understand. To get a better understanding of the data, a good analysis to run is performing descriptive statistics as was done in module 1.

### Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation
Collection	146	-8,00	12,60	3,0148	2,3457
Assembly	134	-21,20	15,50	6,4396	4,0355
Rural gross	129	5,43	44,77	13,5557	4,8645
Urban gross	136	,27	7,03	3,1623	1,0570
Retail	131	5,09	26,90	13,6574	4,4620
Valid N (listwise)	123				

1. From the **Statistics** menu select **Summarize/Descriptives....**
2. Select **marg1 marg2 marg3 marg4 marg5** from the list on the left and put them in the **Variable(s):** box
3. 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 margins 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 Output navigator (Output9.spo) and discuss them with your colleagues. Before we take a look at mean annual margins in various graphics, save the data (Module9.sav) as we have calculated many variables and would like to keep them for further analysis.

### 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 learnt in the first section of module 5 to lag prices over two weeks. Then compute and graph lagged margins with what you have learnt on gross margins. What do you observe?

### Graphing pie and bar chart for market margins at different levels

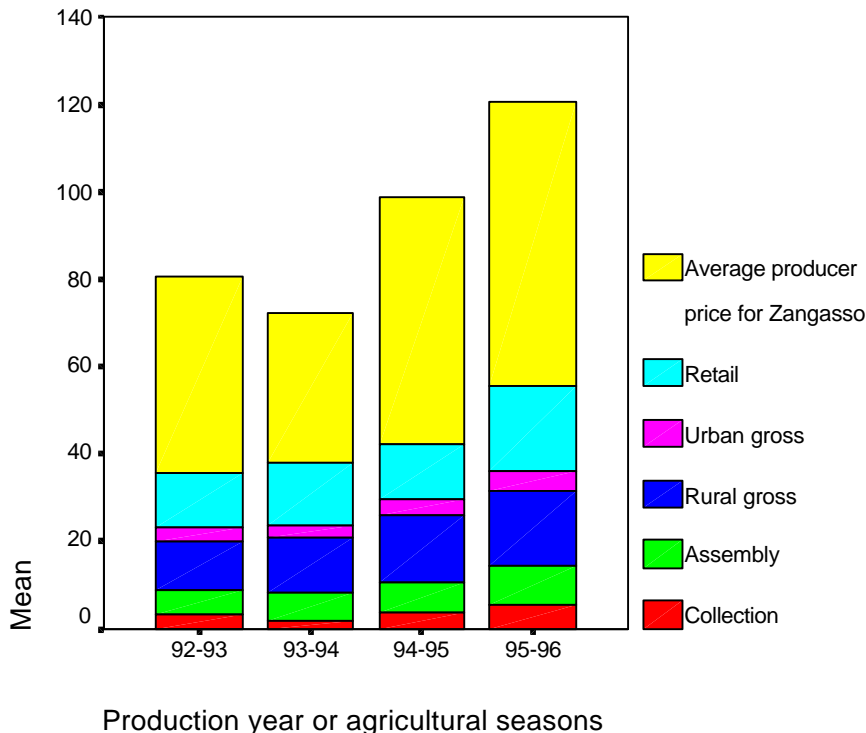
Mean annual margins presented in pie.

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. 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 **▶** 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



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 what levels of the system are absorbing price changes. Hence, before we aggregate data, make the marg.sav file active if it is not already open.

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 variable **marg1** and put in the **Aggregate variable(s)** box.
4. Select the variable **marg1** again and once in the **Aggregate variable(s)** box, click on it again, then click on **Function...**
5. Click on the radio button next to **Standard deviation**
6. Click on **Continue**
7. Repeat steps 3 to 6 but replace the variable **marg1** with **marg2 marg3 marg4 marg5** and **p4p\_zan**, but hold down the left mouse button to copy all of the variables to the **Aggregate variable(s)** box at the same time, keep them highlighted, then click **Function...** to change them all to **Standard deviation**.
8. Click on **File...** and change the name of the file to **MARGPLUS.sav**
9. **Save**
10. Paste and run the command

Now to compute, make the margplus.sav file active and then follow these steps:

1. From the **Transform** menu select **Compute...**
2. For the **Target Variable:** enter **cv\_p4p** (and add a label of your choice)
3. From the **Numeric Expression:** box, enter **p4p\_zan\_2 / p4p\_zan\_1**
4. Paste and run the command
5. Repeat steps 1 to 4 replacing the target variable and numeric expression by the following information:

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

With a simple graph, let's look at the mean coefficient of variations for the various 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 1994 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.

As we will use marg.sav to aggregate to a monthly level and deflate margins by inflation rate, and as we are presently at a yearly level, save this file under a different name so you may use it for further analysis.

## Deflating margins by inflation rate

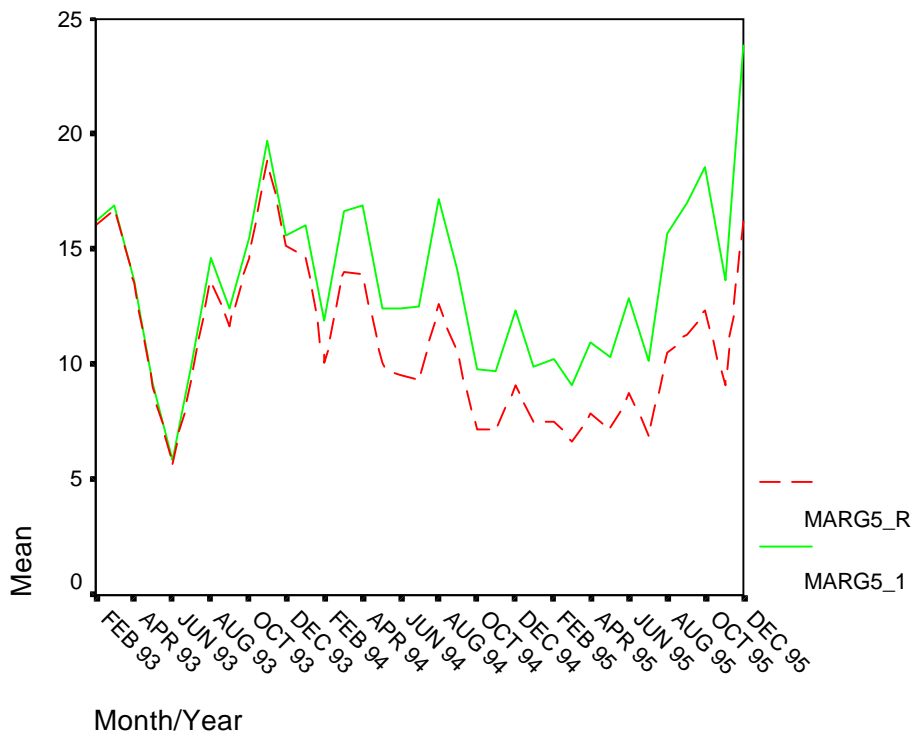
Refer to module 8 for a brief explanation of deflating prices. Open the **margin.sav** file (paste and run the command), then aggregate by **annee** and **mois** following these instructions:

1. Go to **Data/Aggregate...**
2. From the left column, select **annee** and **mois** and put them in **Break Variable(s)**
3. Select the variable **marg1** and put in the **Aggregate variable(s)** box.
4. Repeat steps 3 for **marg2 marg3 marg4 marg5** and **p4p\_zan** all at once by holding down the control key and clicking on the variables
5. Click on **File...** and change the name of the file to **MARGIN.D.SAV**
6. **Save**
7. Paste and run the command
8. We want to bring in the consumer index file we saved to deflate the retail margin so we need to make the **pindex.sav** file active (paste and run the command). Once open, select the data for 1993 upwards (paste and run the command).
9. Now we can merge the two files.
10. Go to **Data/Merge File/Add Variables...**
11. Select the filename **margind.sav** and click on **Open**
12. Check the box next to **Match cases on key variables in sorted files**
13. Click on radio button next to **Both files provide cases**
14. Select **annee mois** from the **Excluded Variables:** list
15. Click on **▶** next to **Key Variables:** (bottom, right)
16. Paste the command
17. Click on **OK**
18. Click on **NO** and then run the command.

Now that we have our working file, we can compute the real retail margin. As before, you need to use **Transform/Compute**, then for the **Target Variable:** enter **marg5\_r** and a corresponding label, and in the **Numeric Expression:** box, enter **marg5\_1 / pindex\*100**. Take time to 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 such month and year. As we have done in the past, we can use the **DATE.MOYR(month,year)** function in the **Compute** command like so (if not already present in data):

1. From the **Transform** menu select **Compute...**
2. For the **Target Variable:** enter **date\_m**
3. Click on **Type&Labels...** and enter 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 and run the command.



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.

As 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. Hence, 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 like so:

1. From the **T**ransform menu select **C**ompute...
2. For the **T**arget Variable: enter **grossmarg** (and add a label of your choice)
3. From the Numeric **E**xpression: box, enter **marg1\_1 + marg2\_1 + marg3\_1 + marg4\_1 + marg5\_1**
4. Paste and run the command
5. We want to discard the missing data for January of 1993 so use the **S**elect command for **grossmarg > 0** and then save the data file (grossmarg.sav).



To use a linear regression, follow these steps:

1. Go to **Statistics/Regression/Linear...**
2. Put **grossmarg** in the Dependent: box
3. Put **pinde**x in the Independent(s): box
4. Paste and run the command

**Coefficients<sup>a</sup>**

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: Gross margin

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).

#### REGRESSING GROSS MARKETING MARGIN BY QUANTITY MARKETED TO EVALUATE IF SCALE EFFECTS IN MARKETING SERVICES

As we have insufficient data, it will be hard to differentiate quantities by market destination. For this example, we will be using rice as the commodity and selecting the Niono-Bamako marketing channel. Make the MODULE5.sav file active. Then select and aggregate rice for Bamako and Niono as follows:

1. From the **Data** menu select **Select Cases** and select the radio button next to **If condition is satisfied**
2. Click on **If...** under **If condition is satisfied**
3. Enter **loc = 3 and cer =4 and annee >= 1993**
4. Click on **Continue**
5. Select the radio button next to **Filtered**
6. Paste and run the command
7. Go to **Data/Aggregate...**
8. From the left column, select **annee mois** and **sem** and put them in the **Break Variable(s)** box
9. Select the variable **p4c** and put in the **Aggregate variable(s)** box.
10. Click on **File...** and change the name of the file to **REGRICE.SAV**
11. **Save**
12. Paste and run the command
13. Repeat steps 1 to 6 but replace **loc = 3** with **loc = 43** under **If condition is satisfied**

14. Repeat steps 7 to 9, replace aggregate variable p4c by p4p and q6 and then click on the radio button next to **Replace working data file**
15. Paste, click **NO** and run the command
16. Go to **Data/Merge File/Add Variables...**
17. Select the filename regrice.sav and click on **Open**
18. Check the box next to **Match cases on key variables in sorted files**
19. Click on radio button next to **Both files provide cases**
20. Select **annee mois sem** from the **Excluded Variables:** list
21. Click on  next to **Key Variables:** (bottom, right)
22. Paste the command
23. Click on **OK**
24. Click on **NO** and then run the command.
25. You will notice from the data that for the same week (**sem** column), we have two different months (e.g. sem = 13, mois = 4 mois = 5) which gives us the impression that there are missing or duplication of data. One way to fix this is to aggregate by week again eliminating the month like so:  
Open Module5.sav. Replace the same filter as above and go to **Data/Aggregate...**
26. From the left column, select **annee** and **sem** and put them in the **Break Variable(s)** box
27. Select the variable **p4c q6** and **p4p** put them the **Aggregate variable(s)** box. Rename them from **p4c\_1 q6\_1** and **p4p\_1** to **p4c, q6** and **p4p**, and relabel them
28. Click on the radio button next to **Replace working data file**
29. Paste and run the command
30. Remember to save the new data file (such as mod9scal.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 (the numeric expression is **p4c - p4p**) and then calculate the linear regression like so:
2. Paste the **Split file** command by **annee** (make sure the **Compare groups** and the **File is already sorted** radio buttons are selected before pasting)
3. Go to **Statistics/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).

## 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 **Statistics/Regression/Linear...**
2. Put **p4c** in the Dependent: box
3. Put **p4p** in the Independent(s): box
4. Paste and run both 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.

Lets looks at the producer/retail price relationship in the opposite way: does farm level price change in response to retail price changes?

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

It would seem so, the fit is fairly good and is again significant - and the value of  $\beta$  is high as well, close to 1. Save your syntax to Module9.sps and close it.

## **Exercise 9.2**

This will be an exercise which may take time as it calls for you to repeat all of the different types of margin analysis performed in this module. You must use maize (**cer = 9**) as the commodity for this exercise. Repeat each of the individual analysis referring yourself as you go to the steps in the module for each of the analysis. 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 editor for documentation and any graphs or output that you may choose for further analysis.

- i) First, you need to aggregate and merge files for margin analysis. Use the same “localités” (loc) and time periods as in the module. Prices for maize are available at the various marketing levels.
- ii) Graph to verify price logic and modify or recode if necessary.
- iii) 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.
- v) deflate margins by inflation rate. Use the **pindex** variable and apply it to corn prices.
- vi) 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 module.

**SPSS for Windows SAMPLE SESSION**  
**Module 10 - Graphs, tables, publications and presentations,**  
**how to bring them into word processor**

**Short Course Training Materials**  
**Designing Policy Relevant Research and**  
**Data Processing and Analysis with SPSS for Windows 7.5**  
**3<sup>rd</sup> Edition**

**Department of Agricultural Economics, Michigan State University**  
**East Lansing, Michigan**  
**February 1999**

The objective of this module is to give you the tools necessary to prepare reports, e.g. market bulletins, and learn how to move SPSS results into other applications. While it is possible to move SPSS text output, tables, charts and other graphics, into word processors and spreadsheets under Windows and DOS, this module will uniquely focus on a chart or table as an example. The methods used in this example would be quite similar for other SPSS results.

This module will not look at how to prepare presentations or publications. An additional module should be developed to look at these questions more precisely and should include such concepts as titles, text boxes, image insertion and watermarks among other basic concepts. Nonetheless, the tools presented in this module, showing how to transfer SPSS results to a word processor, is paramount and key to diffusing and disseminating market information.

The method is simple: once the SPSS results such as a chart or a table are produced (it is always better to save the output as well), it can be printed or incorporated into reports prepared using word processors or publishing programs. Incorporating graphs and charts from SPSS can be done using a simple copy and paste procedure (the **Copy** command must be made from a Output navigator). Unfortunately, if the original chart or table file changes (say you were preparing a monthly bulletin with a graph for real prices, another graph for annual changes and a third for historical trends which include new monthly data collected from the market), to update your report you must erase the old graphs and paste new copies. Proceed with a simple copy and paste procedure of an SPSS graph or table of your choice that you have saved in the SPSS time series sample session modules 5 to 9.

1. Go to **File/Open...** in SPSS for Windows 7.5
2. Select any of the files in the sample folder where you saved your output from the sample session (\*.spo extension - called Navigator document)
3. Click on **O**pen
- 4a. To display an SPSS Output, click once to select an object. Double-click to activate an object for editing. If the object is a pivot table, you can obtain detailed help on items within the table by right-clicking on row and column labels after the table is activated. Once you have selected a table or a chart, you may double click on it (or right click and select the last option in the dialog menu - **SPSS ... O**bject) to open the SPSS pivot table or the SPSS chart editor.
- 4b. When you have finished making adjustments to the table or graph, exit the edit box to return to the Output navigator and right click on the object, select **C**opy or go to **E**dit/**C**opy through the menu system. Another option is to right click and select **E**xport. Note: It is highly recommended to make all modifications to the graphs or

tables within SPSS, NOT in the word processor as within the latter, the graph or table are considered a picture. Making changes to the “picture” within a word processor are allowed but drastically changes the image and does not retain the original format or layout. Unless the it is a windows metafile format (\*.WMF) which will retain all the font characteristics and border styles of the items at the time you copied them.

5. Now open your word processor software if it is not already open
6. Select a spot in your document where you would like to place the graph or chart.
7. Go to the Edit menu and select either **Paste** or **Paste Special** (or if you have selected Export and saved the file to \*.wmf - you simply need to open the file at the selected spot of your document).
8. Selecting **Paste**, the graph or table will be pasted directly on your document. Adjust the size the graph as you wish. Using **Paste Special**, a window will appear and you can select the type of paste/link you would like to set up. Look at the choices and then select **Picture**. Adjust the size of your graph in your document.
9. Save your word processor document.
10. There is something you do not like on the graph or table? You would like to change the axis, the title, a column? Double click on the graph. Go to SPSS and change as you see fit any part of the graph or table, and select **Copy**. Now return to your word processor document and see the changes you just made! You may also modify the graph or table within your word processor but this is recommended only if under a windows metafile format. Else, simply delete the graph or table in your word processor and copy over a revised version of it from SPSS like was done above. It is recommended to make all modifications in SPSS - as the imaging and formatting possibilities within SPSS are numerous and are one of the better achievements and changes in the SPSS 7.5 version. Remember to save your changes at all times.

### **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 sample session output notebook.

## Annexes

Short Course Training Materials  
Designing Policy Relevant Research and  
Data Processing and Analysis with SPSS for Windows 7.5  
3<sup>rd</sup> Edition

Department of Agricultural Economics, Michigan State University  
East Lansing, Michigan  
February 1999

The following annexes were prepared for users of the sample session to have 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 Output navigator.

### 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, a transformation). 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** command allows you to make selections that determine the type of chart you obtain, simple, multiple and drop-line. In the menu, select the icon for the chart type you want, and select the option under **Data in Chart Are** 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 it 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 missing values or sysmisses.

### **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 for Windows 7.5

Numerous modules could be dedicated to working with the Output navigator. 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 Output navigator. Your results have never looked this good! Easier and faster data exploration and to ability to drag icons in the navigator outline and content panes on the left, expand and collapse the outline - 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. This will open a separate window called a pivot table. Then scroll down to see the output in whole. 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.