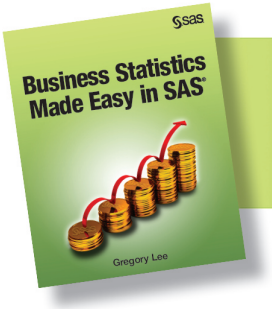
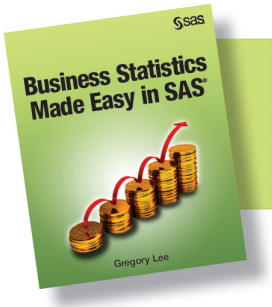
Business Statistics Made Easy in SAS®

Gregory Lee



Contents



6

Basics of SAS Programs, Data Manipulation, Analysis & Reporting

Introduction

This chapter begins the many sections of this book that teach the practical implementation of statistical techniques through SAS. We start in this chapter with an overview of SAS programs and programming, data manipulation, the basics of SAS statistical analysis, and different types of documentary reports in SAS.

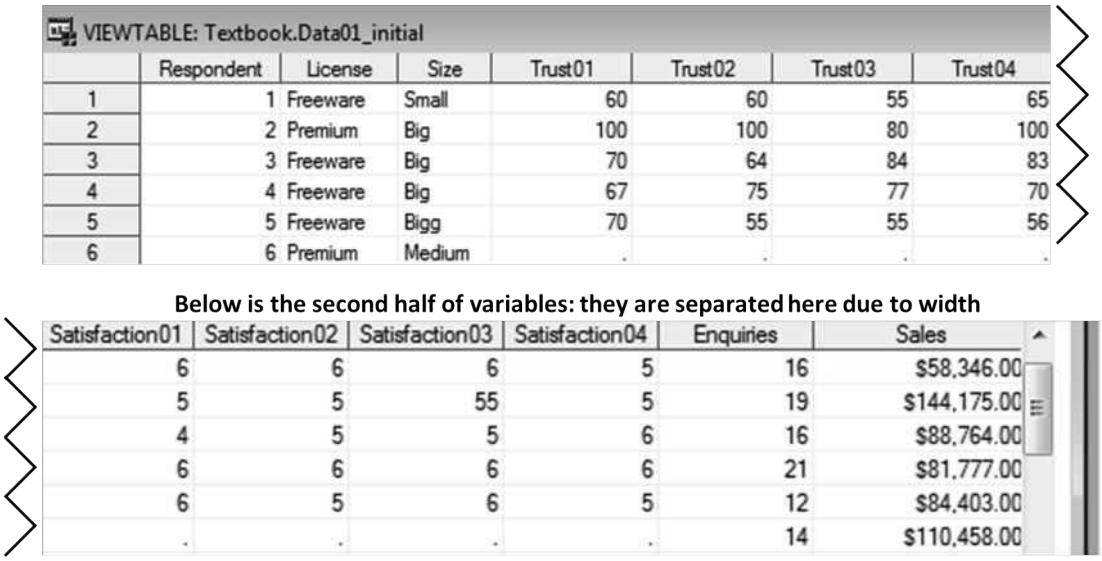
The Running Data Example

Reminder of the Main Textbook Case Study

To facilitate the discussion of the next few chapters, I will continue to work with the Accu-Phi case study from Chapter 1, specifically with the following variables (see Figure 6.1 on page 71 below for a reminder of the initial data format):

- *Sales*: Measured as actual services sales in dollars in the first year of sales.
- *License*: A description of what license the customer has (“Freeware” or “Premium”).
- *Size:* A description of the size of the customer by turnover, with the character values “Small,” “Medium,” or “Big.”
- *Trust:* The trust the customer has in your product and company. You have measured trust through four questions in an online survey, on a 0-100 point sliding scale.
- *Customer satisfaction:* Measured through four questions in an online customer survey, but from 1-7.
- *Enquiries:* The average number of enquiries about the core software product logged with the call center or online help by customers, per month, since starting use of the product.

Figure 6.1 *First lines of initial dataset*



VIEWTABLE: Textbook.Data01_initial

	Respondent	License	Size	Trust01	Trust02	Trust03	Trust04
1	1	Freeware	Small	60	60	55	65
2	2	Premium	Big	100	100	80	100
3	3	Freeware	Big	70	64	84	83
4	4	Freeware	Big	67	75	77	70
5	5	Freeware	Bigg	70	55	55	56
6	6	Premium	Medium	.	.	.	.

Below is the second half of variables: they are separated here due to width

Satisfaction01	Satisfaction02	Satisfaction03	Satisfaction04	Enquiries	Sales
6	6	6	5	16	$58,346.00
5	5	55	5	19	$144,175.00
4	5	5	6	16	$88,764.00
6	6	6	6	21	$81,777.00
6	5	6	5	12	$84,403.00
.	.	.	.	14	$110,458.00

The download available on the course website in the “Textbook Materials” folder, gives this initial dataset (“Data01_Initial”).

Reminder of Your Brief for the Case Example

Let us say that your CEO wants you to analyze the data and answer the following questions which are important to the company:

- How did the first-year sales go?
- Are our customers satisfied and to what extent do they trust us?
- How many enquiries do customers make?
- Do sales, satisfaction, trust or enquiries differ depending on whether the customer has a premium or freeware contract, and depending on customer size?
- What is the distribution of licenses between the levels of size?
- Is sales seemingly substantially associated with any of the other variables?

The Pre-Analysis Data Cleaning & Preparation Steps

Before actually analyzing data to answer questions such as the CEO's queries above, you will need to assess the data for integrity, clean any obvious errors and mistakes, and prepare the data for final analysis. These checks may include:

1. *Initial data assessment and cleaning*. Notice the following in Figure 6.1 on page 71:
    a. Size of the fifth respondent is captured as "Bigg," obviously a typographical error.
    b. The "Satisfaction03" score for Respondent 2 is captured as a "55", but this is supposed to be a 1-7 scale.
    c. These are data entry mistakes. While easy to spot in such a small with the eye, you'll not see this in a bigger table easily. Mis-entered data can seriously impact any analysis.
2. *Missing data*: There is missing data; we need to assess and possibly deal with this as discussed in Chapter 4.
3. *Multi-item scales assessing trust and satisfaction*: We need to assess and aggregate these into single measures of the variables if possible.

We need to pre-assess and clean our data. We usually do these sorts of assessments through basic descriptive statistics and variable associations. Therefore, the next four chapters will sequentially discuss the following:

- Chapter 6 discusses how to create, change and manipulate data, as well as give an overview of some other topics. To do things like create aggregated variables from multi-item scales, we'll need these skills.
- Chapter 7 discusses the essential descriptive statistics we use for single variables.
- Chapter 8 discusses basic measures of variable association.
- Chapter 9 discusses using these analyses in an initial set of steps for the purposes of data checking, cleaning, and preparation.

Overview of the Three Big Tasks in Business Statistics

Having been introduced to the SAS products in the previous chapter, we now turn our attention to a basic introduction to the three major types of tasks you may wish to perform in SAS:

1. *Data manipulation* tasks are those where you wish to change or add to your current data set. For instance, you may wish to sort your current dataset by some variable, or add a new column of data that is the sum of three other columns. Appendix A to this chapter gives you some lessons on how to do such tasks, including manipulating data and creating new datasets.
2. *Data analysis* involves generating representative numbers or pictures of the data that tell you something you wish to know about the data. This could range from an analysis as simple as the average of a variable to complex analysis of the relationships between many variables.
3. *Reporting* obviously means formatting your findings into a useful report that will be appropriate and engaging for the user.

The next sections introduce each of these major steps in greater or less detail, after an initial overview of SAS programming in general.

Basic Introduction to SAS Programming

Running SAS Tasks through Point-and-Click Windows

You can use various point-and-click windows to perform tasks in SAS. This method is relatively simple to use, and favored by many people. If you were using the point-and-click options you could open and use SAS products that work like this, such as *SAS Enterprise Guide* or *JMP*. *SAS Studio* also has a version of this sort of approach built in, called the ”Visual Programmer.”

Point-and-click has serious disadvantages, however, because there are often a great number of check boxs and options, and SAS does not remember your settings. Therefore, every time you re-start a certain section of SAS you have to re-enter many check box options. For this

reason, we will not use the point-and-click options very much in this book, as they are very slow and inefficient.

Doing SAS Tasks through Programming Code (Syntax)

Advantages of Programming Code

Instead of point-and click, SAS usually uses programming code in the SAS 9 Editor window or the SAS Studio Code window to input keywords that tell SAS what you want. Note the following about programming code in general:

1. *Programming is efficient*: The programming code input method is very efficient and advantageous. It is far quicker than using point-and-click. You can save programming code for later use more easily than you can in many point-and-click programs. Finally, point-and-click takes a lot of time to go through if you are in a classroom teaching situation, whereas opening and running a programming code file is quick.

2. *Saving and re-using programming code*: You can save your programming code files and re-use them time and time again (see for instance the programming code files in the "Textbook Materials" folder). Generally, once you have the programming files you like to use, the only thing you have to do is change the names of the datasets and variables.

3. *This book mostly uses programming code*: Because of the advantages of programming code, I will mostly use and teach this input method in this book. You will not have to learn what programming to use; the textbook comes with pre-written programming code files (see the "Textbook Materials" folder at http://support.sas.com/publishing/authors/lee.html). Each time we run an analysis, you will be directed to open and run a pre-existing file as described below.

First Lessons on SAS Programming

Programming can be a daunting task for many people. However, it is actually a very easy language simply composed of a few keywords, as well as a basic structure to which you need to stick.

For instance, take a look at Figure 6.2 on page 75, which shows an example of SAS code in either the Editor window of SAS 9 or the Code window of SAS Studio. Here, you can see various keywords and variable names that tell SAS what dataset to analyze, which variables to analyze, and what statistical analysis to do on these variables.

Figure 6.2 *Example of programming code in a SAS Editor or Code window*

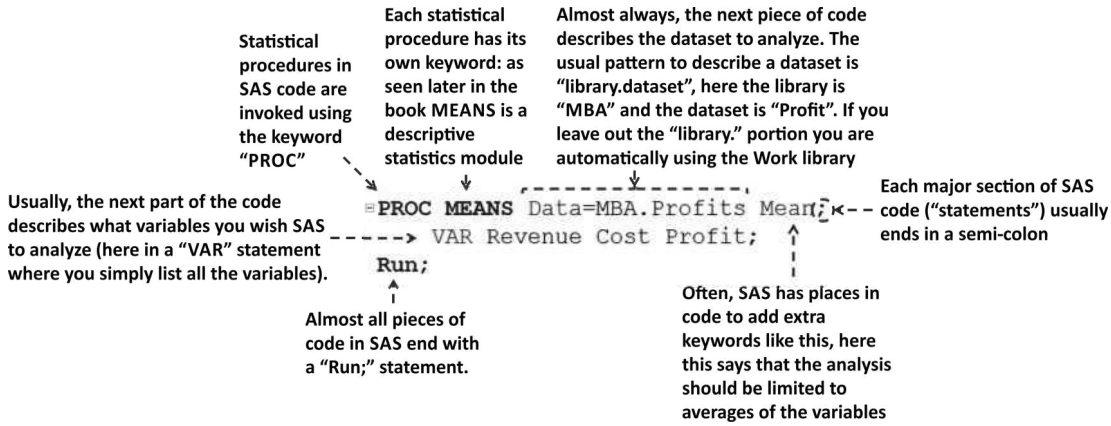


Figure 6.2 on page 75 is a specific type of code that runs a statistical analysis. We can see the following in this figure:

1. To run a SAS statistical procedure, you usually start with the keyword PROC followed by a specific keyword that identifies which particular statistical analysis you want. For instance, in Figure 6.2 on page 75 the keyword MEANS asks SAS to do basic descriptive statistics on variables, as described in later chapters.

2. When running procedures, we next usually identify the dataset to be analyzed by its library and then its dataset name, i.e. the general structure is "<Name of the library>.<Name of the dataset>." In Figure 6.2 on page 75, the dataset to be analyzed is the "Profits" dataset within the "MBA" library, as identified by the "*Data=MBA.Profits*" part of the code.

3. There are often extra keywords to identify further statistical options.

4. Usually, the middle section of SAS procedure code contains a description of the variables to be analyzed. In the simple example in Figure 6.2 on page 75, we simply list the variables to be analyzed after the keyword VAR. In more complex procedures that are mostly beyond the scope of this book, we sometimes also have to tell SAS how the variables are related.

There are also certain general SAS programming rules that can be seen in the example in Figure 6.2 on page 75:

1. *Capitalization of words in SAS code*:

    a. SAS programs usually do not care about capitalization of words. For instance, in Figure 6.2 on page 75, keywords such as "Proc Means" could easily be spelled "PROC means" or any combination or lower and uppercase, as can the dataset names.

   b Almost the only time that SAS cares about capitalization is if you are referring to specific text data within a dataset. For instance if "Gregory Lee" is a field in a dataset, then if you need to refer to this data in code, you must get the exact capitalization correct.

2 *Spacing, lines and tabs in SAS code*:

   a It does matter that you keep at least one space between different keywords of SAS programming (e.g. you can't put "PROCMEANS" above).

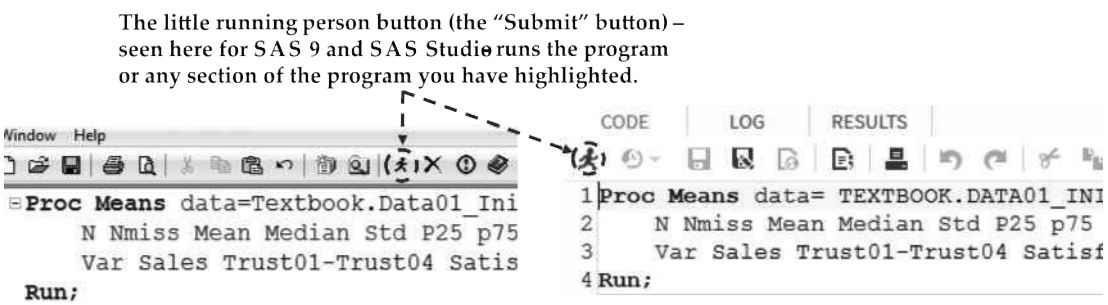
   b However, other than that, SAS does not mind where in the code or editor window you place code so long as the basic statements are in the right order. You can place different statements on different lines, run them together without line breaks, or use multiple spaces or tabs between pieces of code, etc.

3 *Semicolons as the key for endings of sections*: Sections of SAS programs end with a semicolon (";"). If you try to run a SAS program and find that it does not work, it is often because you have failed to add the semicolon at the end of a section.

4 *The Run command as the key for the end of a program*: SAS programs usually end with a "Run;" command.

5 *Running a SAS program*: To actually make the program run, you click the little running person icon in the SAS 9 or SAS Studio toolbar, as seen in Figure 6.3 on page 76 below.

Figure 6.3 *Running a SAS Program*



One cardinal rule is to always check the SAS log after running code to see if the program has worked and to determine if there are errors (e.g. misspelling the dataset name). In such cases, SAS will warn you in the log with red error sections. This is particularly easy in SAS Studio, which lists any errors at the top of the log section.

Finally, note that "PROC"-type code to invoke SAS statistical analyses are not the only form of programming. Notably, the very important DATA keyword is used to create and manipulate datasets, as described below in "Major Task #1: Data Manipulation in SAS" on page 77.

Opening Existing SAS Code Files

As I have discussed above, this book does not expect the reader to become a SAS programmer immediately. All the analyses taught in the book are given to you as pre-written programming code files that you simply have to open and run to get the results. As you work with these files, you will quickly see how the underlying programs work, and soon be able to apply them to your own datasets and variables with little change.

Even if you were to write your own programs from scratch, you would usually save the code files and then re-open and run them later when you wish to recreate the analysis.

To open existing programming code files like those in the "Textbook Materials" folder, do the following:

- *In SAS 9,* go to File > Open Program and navigate to where the file is stored on your hard drive.
- *In SAS Studio*, go to the Server Files and Folders section, and open the code file by double clicking on it (for instance, see the many code files in the "Textbook Materials SAS Studio" folder).

As mentioned in the chapter introduction, there are three big tasks in SAS, namely, data manipulation, data analysis, and report generation. The following sections discuss these steps further.

Major Task #1: Data Manipulation in SAS

Introduction to Data Manipulation

Data manipulation – in other words, changing data or creating new data – is one of the most important tasks in practical business statistics. After capturing data, it is rarely the case that the initial sheet or database query is completely perfect for analysis. Often, changes need to be made, for various reasons such as:

- Imperfections in the original data that need to be fixed
- The need to add new data
- The need to combine multiple datasets

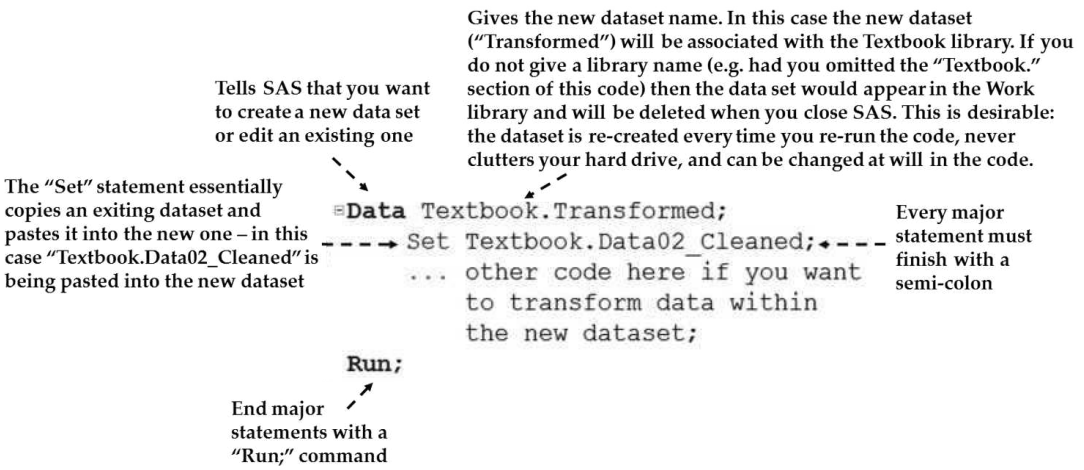
While you can manipulate data in more basic spreadsheet programs like Microsoft Excel, you can also do so in SAS, and far more simply, flexibly and reliably. This book cannot cover much of the SAS data manipulation universe, which is enormous and world-leading. The next few sections can cover only a few salient topics. For a broader introduction to these topics, the reader should consult texts such as Delwitch & Slaughter (2012).

Creating New Datasets in SAS

As a first topic, we often create new datasets in SAS programming code. This section discusses the basics of doing this.

Of course, one way to create new datasets in SAS is to import them from elsewhere, such as importing Microsoft Excel files. Chapter 5 describes how to do this. This chapter is more interested in dealing with data once it is in SAS. To create or manipulate data in SAS we use a "DATA" statement. Figure 6.4 on page 78 shows the outline of a data step for creating a new dataset SAS.

Figure 6.4 *Creating a new dataset in SAS*



As seen in Figure 6.4 on page 78, if you wish to create a new dataset you do the following:

1. *Start with the keyword DATA,* which tells SAS that you wish to create a new dataset.
2. *Name the new dataset*. Note the following:
   - **a** Specify the name of a library and a dataset name, separated by a period (e.g. "Textbook.Transformed" in Figure 6.4 on page 78). The new SAS dataset will appear in the physical folder you have associated with this library. Of course, you have to have associated this library name with the folder beforehand, as described in Chapter 5.
   - **b** There are basic rules for naming SAS datasets. This can be any name – in the code above we used the name "Transformed" – so long as it follows these rules:
     - **i** A SAS name can contain from one to 32 characters.
     - **ii** The first character must be a letter or an underscore (_).

   iii Subsequent characters must be letters, numbers, or underscores.

   iv Blanks cannot appear in SAS names. If you want to separate parts of the dataset name, use underscores, e.g. "Dataset_03."

c If you leave out the library name and give only a dataset name (e.g. the "Data Transformed;" line in Figure 6.5 on page 81 below) then the new dataset will be created in the special "Work" library. In other words, calling the dataset "MyData" is the same as calling it "Work.MyData". The "Work" library is automatically created as part of the SAS installation, and I explain it in more detail in the next section. Using this option is often desirable.

d If you choose the same name and library as an existing dataset, then you will overwrite (i.e. replace) the original version of the dataset.

3 *Populate the new dataset with initial data*. There are two main choices here:

a *Populate the new dataset with data from another dataset*. We frequently base the new dataset on the data from an existing dataset. Think of this as a copy and paste, i.e. you are copying data from an existing dataset into your new dataset. As seen in Figure 6.4 on page 78, we can do this by putting the line "*SET <name of existing dataset>;*" into a DATA step. In Figure 6.4 on page 78, we are using the SET statement to copy all the contents of the "Data02_Cleaned" dataset into the new "Transformed" dataset. In this code, both datasets are located in the "Textbook" library.

b *Enter raw data directly into SAS*. You can also enter data literally in SAS in a DATA step. This book will not cover this direct data input option. I personally advocate importing initial raw data from a spreadsheet program such as Microsoft Excel.

4 *If desired, manipulate the data.* In the DATA step, we can manipulate the data in a great number of ways. "Create New Variables or Manipulate Current Variables in SAS" on page 80 below describes more on such steps.

5 *Other programming notes*: As seen in Figure 6.4 on page 78, do not forget to place semicolons between major statements and add a "Run;" statement at the end before running.

Creating Temporary Datasets in the Work Library

The previous section noted that if you do not give a library name as part of a dataset name then you are automatically linking the dataset with the special "Work" folder (so specifying "Profits" is the same as saying "Work.Profits").

The Work library has a special property: all datasets contained within it are deleted when you close SAS. This is desirable in many cases for two major reasons:

- Datasets created in the Work folder do not clutter your hard drive or server, as they are deleted once you close SAS. However, because you can save the code used to create

them, these datasets can be re-created every time you re-run the code. Programming code takes up far less space on a computer than data.

- If you keep your original data and copy it to a Work library dataset, then changes you make to the new dataset do not affect the original data, which means you are never at risk of harming your original dataset.

This method of creating datasets out of programming code only for the duration of your session – and analyzing the temporary data as you need - is highly efficient and often used by SAS analysts.

On the other hand, giving a SAS library name other than Work causes the dataset to be stored permanently in the folder associated with that library. This is, of course, desirable in cases where you do wish to maintain a permanent copy.

Create New Variables or Manipulate Current Variables in SAS

There are many situations in business statistics where you wish to create a new variable that is, in effect, a transformation of an existing variable's data. Here are some initial examples:

- Creating an index such as a financial ratio (such as creating a price/earnings ratio from two columns containing price and earnings data, respectively).
- Creating mathematical transformations of variables, such as a new variable that is the square root or log of another variable.
- Using the birthdates of people to create a new column that, on a consistently updating basis, calculates their ages.

In addition, you can change and manipulate existing variables in SAS.

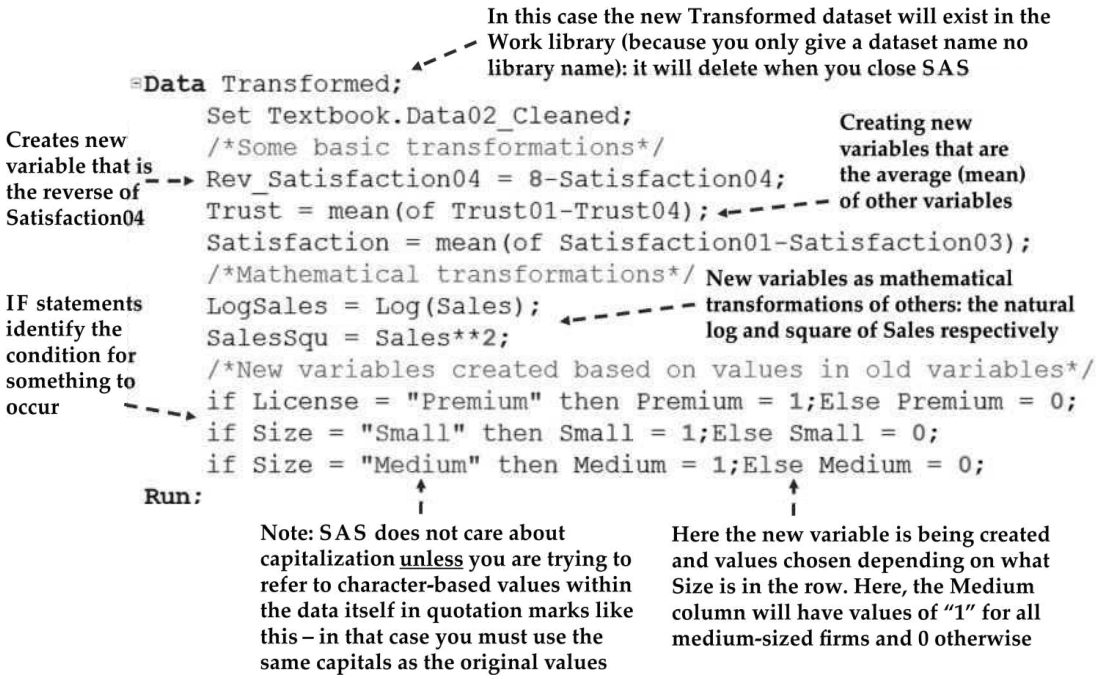
In our main textbook example, so far we have two major types of such tasks:

1. *Creating new variables that reverse the data of reverse-worded survey questions*. Specifically, Satisfaction04 is a reverse-worded survey item (see Chapter 4 and Chapter 9 for more on this), which required us to create a new variable that reverses its data.
2. *Creating two new factor variables, which are the aggregation of multi-item scales*. Trust and satisfaction ultimately needed to be created as factors which are an average of the individual multi-item scores. (Of course, we can't do this step without having assessed internal reliability. Again, see Chapters 4 and 9).

One of the many things SAS is brilliant at is data manipulation. You can manipulate data by using the SAS point-and-click interfaces like SAS Enterprise Guide, but it is quicker and easier to use code in programs like SAS 9 or SAS Studio. The DATA step in SAS not only creates new datasets or edits existing ones, but manipulates data columns or rows.

Figure 6.5 on page 81 shows a sample SAS data step in which the new dataset is created based on an existing dataset (specifically, we create a dataset called "Transformed" in the Work library because no library is specified, and we copy and paste everything from the Textbook.Data02_Cleaned dataset using the SET statement).

Figure 6.5 *Example of creating new variables in the SAS DATA step*



Then, each subsequent line creates a new variable:

- We create a new variable called "Rev_Satisfaction04" that takes the data from the existing variable Satisfaction04 and reverses it using the principles discussed in Chapter 4.
- We create new variables called "Trust" and "Satisfaction" that are averages of some of the individual currently existing multi-item scale columns. Note the way the average works. Also, note here that I have only averaged the values for Satisfaction01-Satisfaction03; see Chapter 9 a little later for why.
- We create two new mathematical transformations of the Sales variable, one the natural log and one for the square (each Sales number to the power of two).
- We create several *conditional* variables using the IF-THEN concept, where the new variable only takes on a certain value if a given condition is true. In the first of these, we create a new variable called "Premium" that will have the value 1 whenever the currently existing License variable contains the value "Premium" in a row, and takes the value 0 for all rows where License is not "Premium."

Take note of the following programming notes about this sort of programming:

- Take another look at the IF-THEN statements in Figure 6.5 on page 81. Note here that this is the only situation in which capitalization counts in SAS. Take the example of the *if License = "Premium"* section of the code. Here, we are asking SAS to go look in the dataset for all rows where this exact condition is true *including the exact capitalization of*

"Premium," and then apply the result only in those rows. If there are also entries in the License column spelled "premium" then the above condition will not identify these rows. So, be careful of capitalization in these situations only.

- As always, note that all statements are separated by semicolons and the entire set ends with a "Run;" statement.

You could do so much more. For instance, you could create a new variable that is the sum of other variables (replace MEAN in the above code with SUM). You can identify rows to delete based on certain rules. SAS has an almost endless set of possible variable manipulations – see the SAS helpfiles (notably SAS/STAT 13.2 User's Guide) for more.

Once you have told SAS what you want to do, submit the code using the Run button as seen above. Once you have done so, always check the log for errors and always open the new dataset to check that it is right. (And then close it: an open dataset in SAS cannot be replaced).

You can see the code from this section in the textbook resources files, under "Code06 Manipulating data example."

Combining Datasets

Often in the business world, we need to combine two or more datasets together. You can combine datasets side-by-side, one on top of the other, merge them based on a match in a certain variable, and so on.

Let us look at one of the most common examples: match merging. Imagine you are an organization with the following two datasets:

1. A database of customer account data, where each customer is identified by a unique customer number.
2. A different database of customer satisfaction survey data. Again, each customer's survey responses are identified by the customer number. Typically, only a limited subset of customers would have filled in the survey.

Now, let us say that you wish to combine these two datasets so that you can link the data. Each row needs to be matched up by customer number. You can do this in SAS using the MERGE statement. See the following example:

Example Code 6.1 *Example of merge matching data in SAS*

```
Data Customers.Merged;
    Merge Customers.Accounts Customers.Survey2016;
    By Customer_ID;
Run;
```

There are many nuances and complexities to combining datasets – for instance, to match merge by a common variable as I show above, both datasets must be sorted by the common matching variable (i.e. you would have to sort both of the above datasets by Customer_ID). For more on combining datasets, reference the SAS helpfiles or books such as Delwitch & Slaughter (2012).

This basic understanding of SAS data manipulation will help us in various parts of the rest of the book, since data manipulation is frequently required in statistical analysis.

Major Task #2: Data Analysis

"Basic Introduction to SAS Programming" on page 73 above discussed the basics of programming a PROC step in SAS, which is the foundation of SAS statistical analyses. The rest of this book gives various examples of core SAS statistical analyses in the context of business.

Just a few more general points apply to thinking about SAS data analyses:

- Knowing which analysis is the appropriate one for your situation is obviously critical. This book discusses many introductory analyses to help you begin this journey. However, especially when you are entering into more complex modelling, you should first carefully investigate the general ideas behind what the correct analysis is. Thereafter, you can read up on how SAS implements that specific analysis through code.
- You can easily find prior examples of SAS code for your desired analysis in the SAS helpfiles, online through SAS User Group articles or the like, or in books like this one. Then, you can copy the code developed in those sources and simply change the names of the dataset and variables for your particular analysis. In a similar vein, SAS Studio has pre-written code in the Tasks section.
- Often, in the same SAS program, we will first manipulate data and then – immediately below the DATA step – place the PROC step that references and analyses the dataset created above. We can then run the set together, change the data or analysis steps again if required, and so on. Example Code 6.2 on page 83 is an example.

Example Code 6.2 *Example of running DATA and PROC steps together*

```
Data Transformed;
    Set MBA.Profits;
    LogRevenue = Log(Revenue);
Run;
Proc Means data=Transformed;
    Var LogRevenue Cost Profit;
Run;
```

Major Task #3: SAS Reporting through Output Formats

Introduction

In the early days of its development, SAS reproduced statistical reports in very simple, old-fashioned listing type format, which was designed for line printers. How times have changed!

Now, modern SAS technologies work with their proprietary Output Delivery System (ODS) system, which allows you to tell SAS to output reports like tables and graphs in multiple different formats. For instance, SAS can put output into:

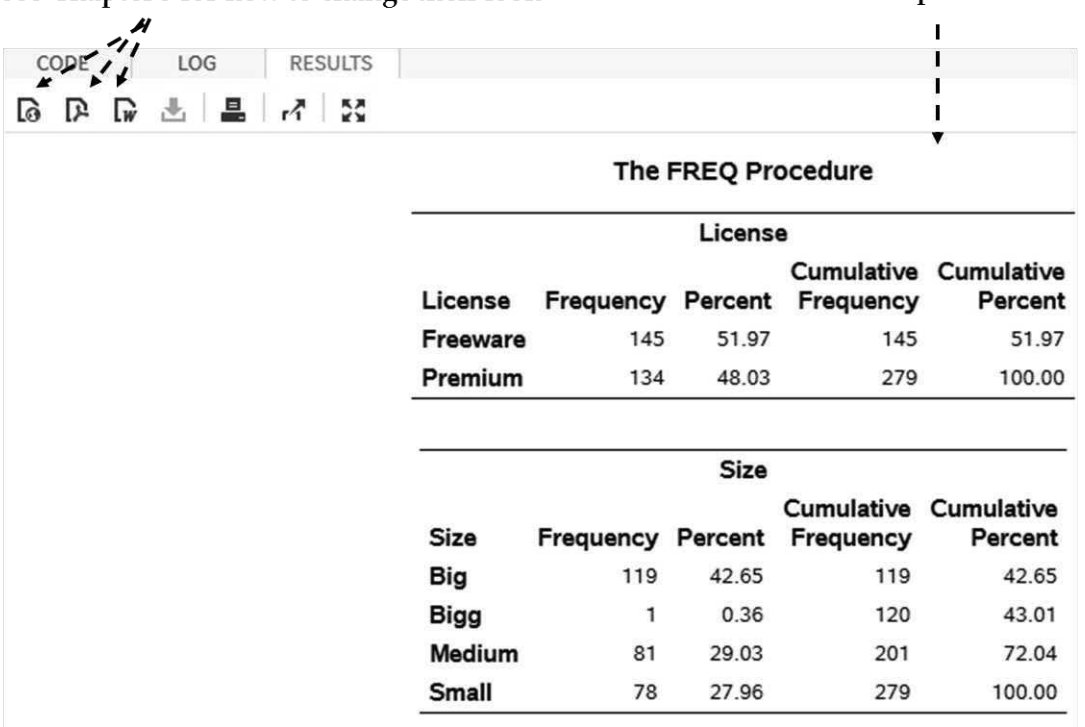
1. *Attractive HTML files.* This is set up as the default in newer SAS versions, and we have already seen in Chapter 5 how to change the automatic settings of how this output will look. You can save the automatic output as an HTML file in either SAS 9 or SAS Studio.
2. *Rich Text Files*, which will open as Microsoft Word or similar files.
3. *PDF files*, which will open in Adobe Acrobat or other PDF readers.
4. *Datasets created from output.* These, in turn, can be exported to spreadsheet or database programs such as Microsoft Excel or Access.
5. Several more.

These output delivery options are incredibly flexible, easy, and attractive. How to get these different formatted outputs differs between SAS Studio and SAS 9.

Different ODS Outputs in SAS Studio

In SAS Studio, you can download results in HTML (web browser), PDF (Acrobat or similar) or RTF (Microsoft Word or similar) formats at the click of a button, as seen in Figure 6.6 on page 85.

Figure 6.6 *Downloading ODS results in different formats in SAS Studio*



This is a major advantage for SAS Studio. Note also that the PDF output contains a menu allowing you to navigate between different sections of a longer report.

Different ODS Outputs in SAS 9

In SAS 9, you need to program the ODS outputs. Luckily, this is mostly very easy. For instance, say you wish to create a rich text output of various tables and graphs that you have created with SAS. Then, merely enter code like that in Example Code 6.3 on page 85 which will open your default program for processing rich text (like MS Word) and create a new file containing your SAS output (as you can see, you stipulate a filename and location for it to be saved to):

Example Code 6.3 *Example: Output in a rich text format that will open in MS Word or similar*

```
ODS RTF file='c://Output.rtf';
<Insert SAS code here to create output like statistical tables & graphs>
```

```
ODS RTF close;
```

The ODS formats need to be studied by the dedicated user, but they all mostly work as simply as the above example. The following are further examples:

Example Code 6.4 *Example of changing HTML output style*

```
ODS HTML style =HTMLBlue;
<Insert SAS code here to create output like statistical graphs>
ODS HTML style = Journal2;
```

The above example changes the HTML output style – which will usually open in SAS when you run anything – to a specific style called HTMLBlue. I set your style to Journal2 above because it produces clean black-and-white tables, however, in Chapter 10 later we will do graphing which is often best done in color. In the above code, you change to HTMLBlue which allows color output, then change back to Journal2.

Example Code 6.5 *Example: Writing SAS output to a PDF that will open in Acrobat or similar*

```
ODS PDF file='c://Output.pdf';
<Insert SAS code here to create output like statistical tables & graphs>
ODS PDF close;
```

Once again, this will save and open a PDF file of your output.

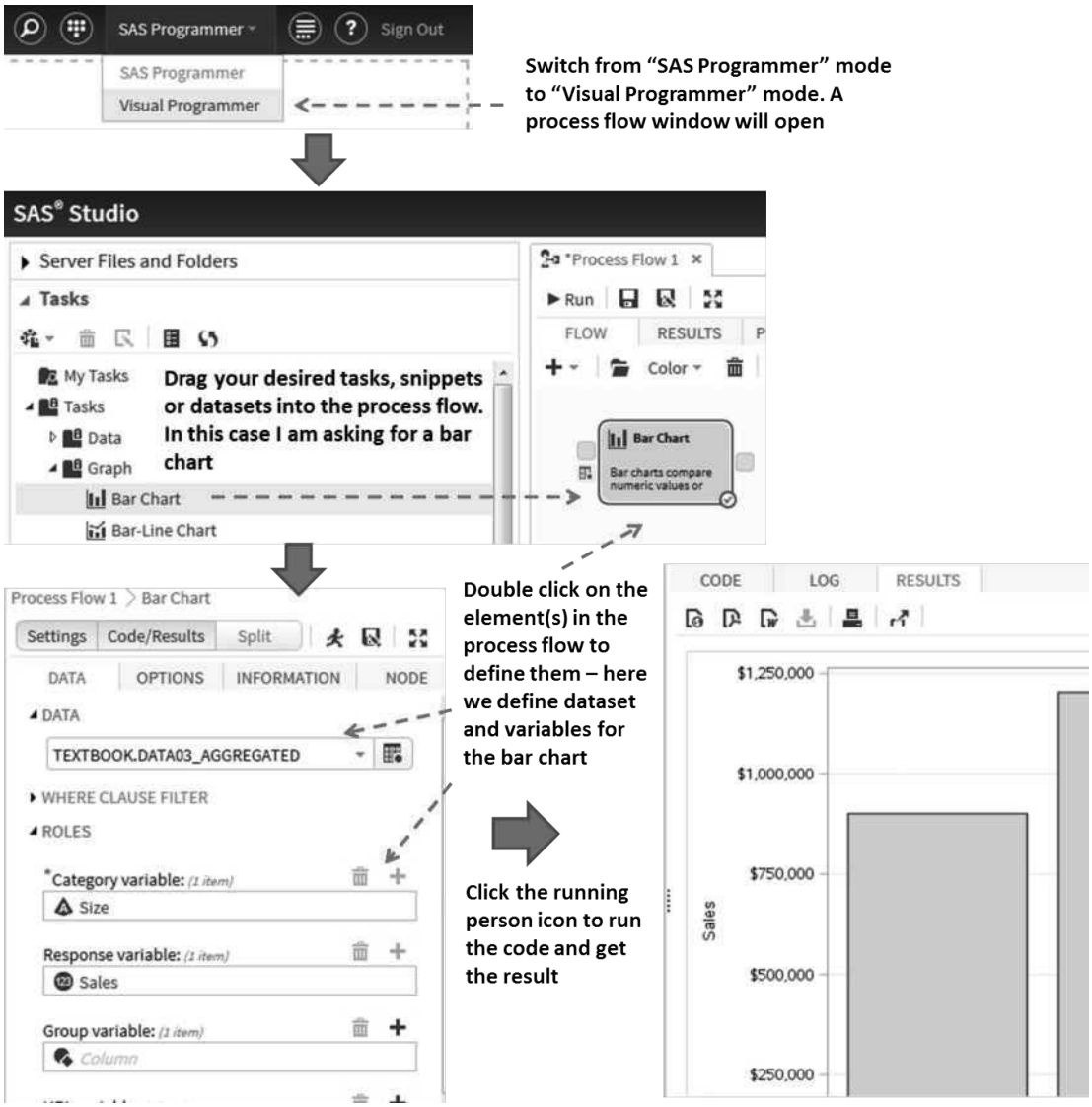
SAS ODS is an incredibly powerful system for crafting your SAS output. Any time you want to say "*hey, I'm creating such-and-such analysis in SAS and I would want it to look like that and come out in such-and-such a format,*" then ODS can usually do it for you.

The Visual Programmer Mode in SAS Studio

So far, I have demonstrated programming in SAS. As much as I have argued for using programming as the most efficient way of achieving analysis and teaching statistics in many cases, SAS Studio has created a clever way of generating your programs that allows you the comfort of a point-and-click type approach that works with SAS programming. This is known as the Visual Programmer mode.

In the SAS Studio Visual Programmer mode, you can define your dataset, task and variables for SAS Studio using easy-to-understand drag-and-drop methods. As an example of the use of this mode, see Figure 6.7 on page 87 below.

Figure 6.7 *Example of using the Visual Programmer mode in SAS Studio*



In this example, I have generated a bar chart simply by doing the following easy steps:

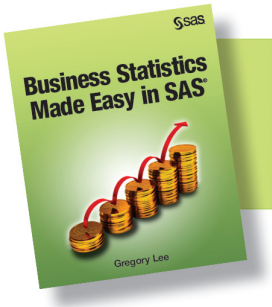
- Initiate SAS Studio Visual Programmer mode by switching from SAS Programmer mode at the top right. This opens a process flow window.
- Drag a pre-defined task from the Task window (in this case the Graphs > Bar Chart task) to the process flow.
- Double click the resulting Bar Chart process piece gives the settings. Here I define the dataset and variables using easy drop-down fields.

- Click the "running person" icon to get the results. Note: To see the graph in color, switch to the HTMLBlue results style in Preferences.

There are many other Tasks and what are called "Snippets" (pieces of code that can be used in various places). You should browse through these – perhaps after reading the book and acquainting yourself with the field of basic statistics – to see what Visual Programmer has to offer. It is an intuitive and pleasing way to generate simple tasks, but has other disadvantages of point-and-click modes, such as lack of the full functionality SAS programming can offer.

Conclusion

This chapter has introduced data manipulation in SAS, the absolute basics of analysis, and it has shown us how to create results in various formats. The rest of this book discusses a variety of analyses and principles that – used correctly - will launch you on a productive and profitable business statistics path.



Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

About the Author

Professor Gregory Lee is currently the Research Director and an Associate Professor in Research Methodology and Decision Sciences at the AMBA-rated Wits Business School.

He has published prior books on human resources (HR) metrics, and has many article publications in the international arena such as the *Human Resource Management Journal*, *European Journal of Operational Research*, *Scientometrics*, *Journal of Business-to-Business Marketing*, *The International Journal of Human Resource Management*, *International Journal of Manpower*, *Review of Income & Wealth*, *Journal of Human Resource Costing & Accounting* and many others.

He focuses on issues in human resource management, notably HR metrics (in which he has established himself as a leading expert) and other areas such as training, employee turnover and the employee-customer link.

He has served in many capacities within the international academic field. He has sat on the Graduate Management Admissions Council (GMAC©) advisory council, the editorial boards of the *Journal of Organizational and Occupational Psychology*, and engages in frequent reviewing for many journals.

In addition, he is a well-known consultant, writer and speaker in the corporate and practical management arenas, notably in the area of HR metrics, but extending to other areas such as human resources strategy and foresight.