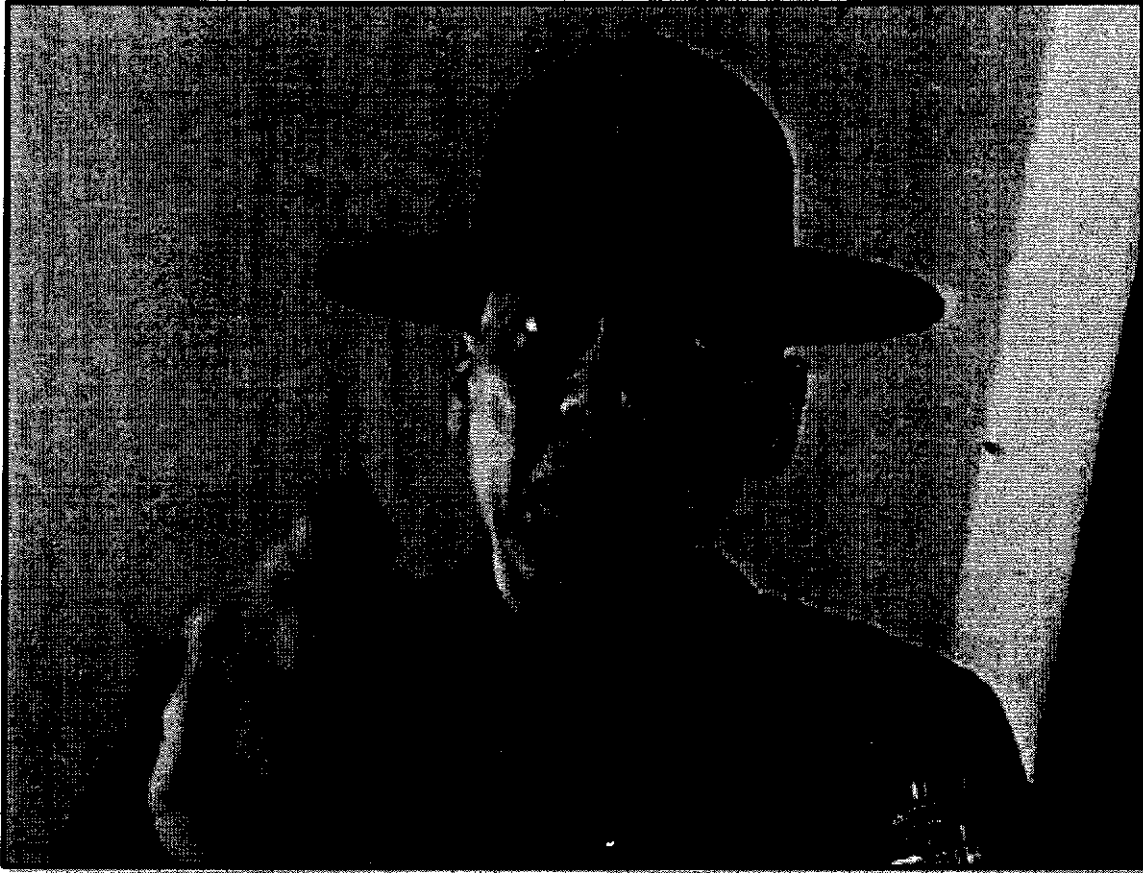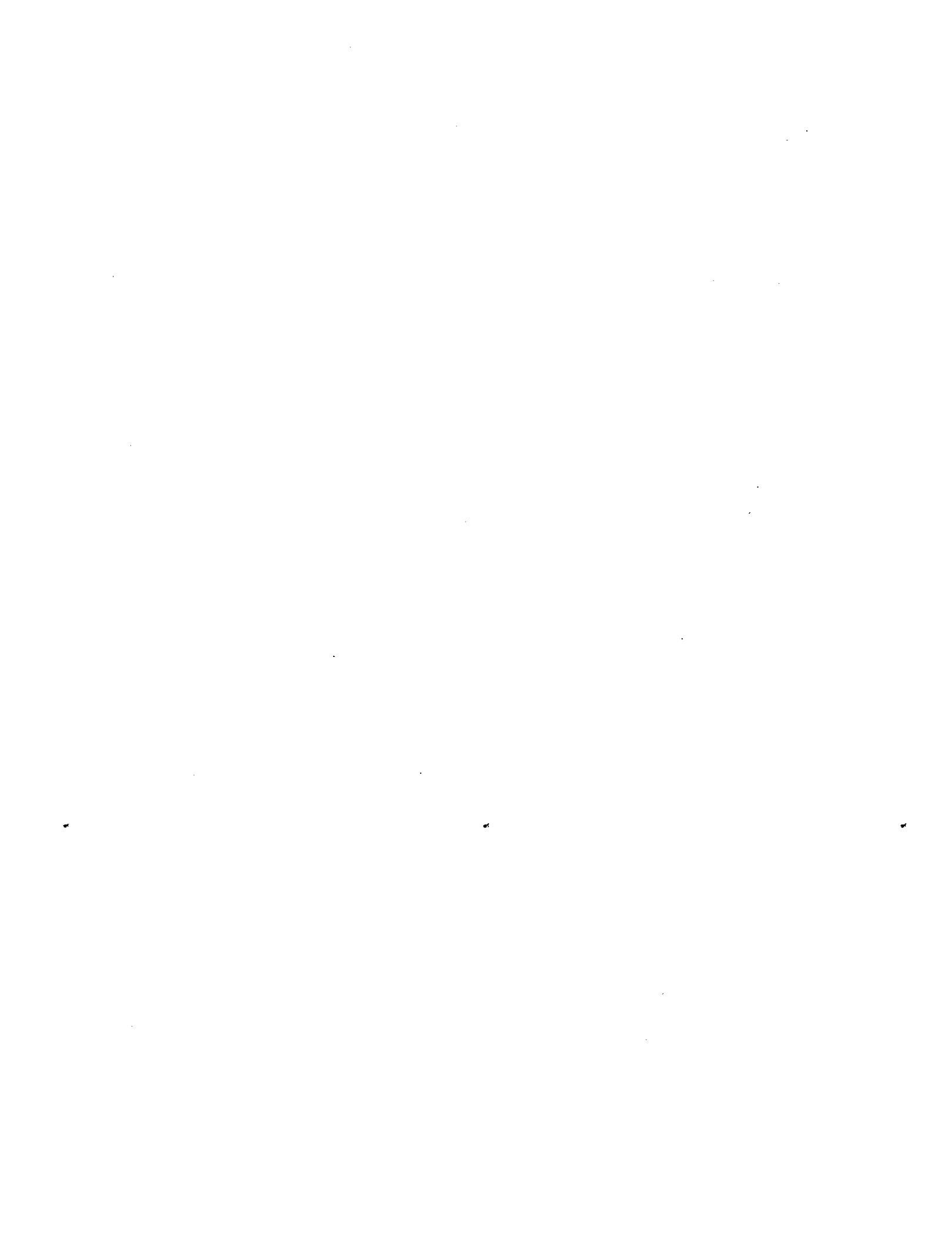# SAS® Training Manual
for New Recruits

THE UNIVERSITY OF IOWA
SAS® Boot Camp*
August 17-18, 2010

* Event sponsored by UI Human Resources, the Institute for Clinical and Translational Science, and the Office of the Chief Information Officer

SAS® Training Manual
for new Recruits

This manual provides an introduction to basic programming operations and procedures of the SAS® system. It is designed specifically to help those new to the use of SAS who have a desire to learn how to apply the statistical analysis features of SAS to their research. The manual focuses on the simplest, most direct methods for reading in data, performing data transformations and conducting analyses. These materials have been organized around key concepts in a PowerPoint presentation that accompanies the manual.

Introduction   Slide 1

The SAS® System, originally Statistical Analysis System, is an integrated arrangement of software products that enables a user to perform:

- ✓ Data entry, retrieval, management, and mining
- ✓ Report writing and graphics
- ✓ Statistical analysis and operations research
- ✓ Forecasting and decision support
- ✓ Data warehousing
- ✓ Systems management and enterprise solutions

As a statistical software package, SAS allows the user to manipulate and analyze data in many different ways. Because of its capabilities, SAS is used in many disciplines, including medical sciences, biological sciences and social sciences. The SAS system provides a flexible framework for integrating clinical, sample and experimental data. Knowing the SAS programming language will help you not only in your own research, but also in the development of additional research skills.

When learning how to use SAS it is important to recognize that sometimes there are different ways to accomplish the same task in SAS. For example, there are literally dozens of ways to complete the task of reading in a data file for analysis. In this manual, we will focus on the simplest, most direct method for reading in the types of data files provided to you. Obviously if you encounter other types of data files in your own work, the methods outlined here might not be the most appropriate.

Tips and Tricks
A number of timesaving "tips and tricks" have been handed down by word of mouth from veteran SAS users to novices over the years. Veterans probably wished these tricks were known when they were learning how to program SAS for the first time. Some important tips for beginners are listed in enclosed boxes throughout this manual.

Section I. Learning the SAS Windows Environment

Section Overview

Learning Objectives

At the conclusion of this session, participants should be able to:

- demonstrate knowledge and understanding of how to access SAS on the UI campus
- identify various components that make up the SAS windows environment

Section Outline

- SAS Access at The University of Iowa
- Starting the SAS System
- Getting Familiar with the Terrain
- The Enhanced Editor Window
- The Log Window
- The Output Window
- The Explorer Window
- The Results Window
- Class Exercise

SAS Access at The University of Iowa

SAS in a "stand-alone" form can be operated in a PC Windows, Macintosh or Linux (running Windows) environment. Access to SAS is also available on the UI campus through the Virtual Desktop -- a web-based system which allows access to a number of software applications from virtually any computer with an Internet connection, on or off campus. The Virtual Desktop makes SAS available 24/7 from anywhere in the world. The only requirement is that you have a valid HawkID and that you install the Citrix client or plug-in on your computer. Currently SAS 9.1.3 and 9.2 are available on the Virtual Desktop in campus ITCs; SAS 9.2 is also available in a "stand-alone" form on individual computers in the Statistics ITC in Schaeffer Hall, the Hardin Library ITC, and College of Public Health.

Additional information about how to install the Citrix client or plug-in on your personal computer and how to access SAS using the Virtual Desktop can be found at http://helpdesk.its.uiowa.edu/virtualdesktop/.

Tips and Tricks

If at any point during the installation or after the client or plug-in has been launched you receive a message that gives you an opportunity to update the client or plug-in, you should select "upgrade later" as your option.


Starting the SAS System

You can launch the SAS System by clicking first on the *Start* button, then clicking on *All Programs*, followed by clicking on the program group labeled *SAS* or *The SAS System*, and finally clicking on either *SAS 9.1.3* or *SAS 9.2*, dependent on which version of SAS has been installed on the "stand-alone" computer; or if using the virtual desktop, you need only to log in on the computer with your *HawkID*, click on *Start, All Programs*, then the version of SAS to which you have access rights. The Citrix client or plug-in has already been installed.

Getting Familiar with the Terrain

After you click and open SAS, three important windows are automatically opened: the Enhanced Editor, the LOG, and the Output. The Log and Enhanced Editor are immediately visible, while the Output window is hidden beneath these windows and will not become available until you actually have output. Each of these components will be explained in more detail in the coming sessions. On the left-hand side of the screen, there is also a window labeled Explorer. This window can be helpful with file management, but we will also cover that in a later session.
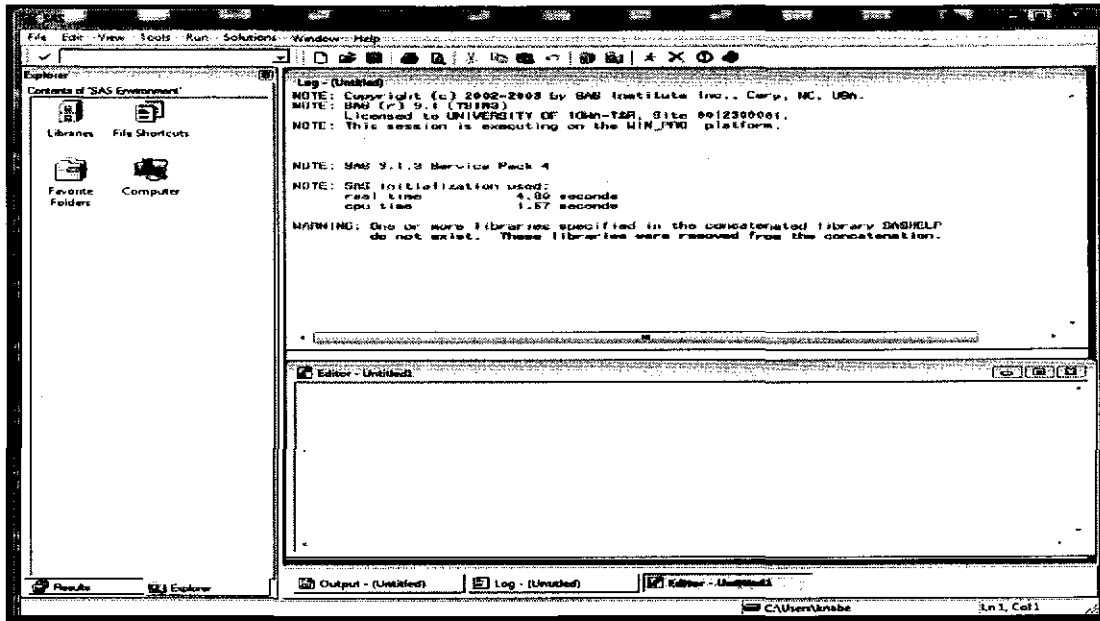
The Enhanced Editor Window

When the SAS System opens, the default active window is the Enhanced Editor window. You know the Enhanced Editor is the active window because it has a dark blue stripe at the top. The Log window has a light blue stripe, indicating it is inactive. The Enhanced Editor window is where you write programs you wish to run on the SAS System. These programs consist of a series of SAS commands that read in and manipulate data, as well as perform statistical analyses.
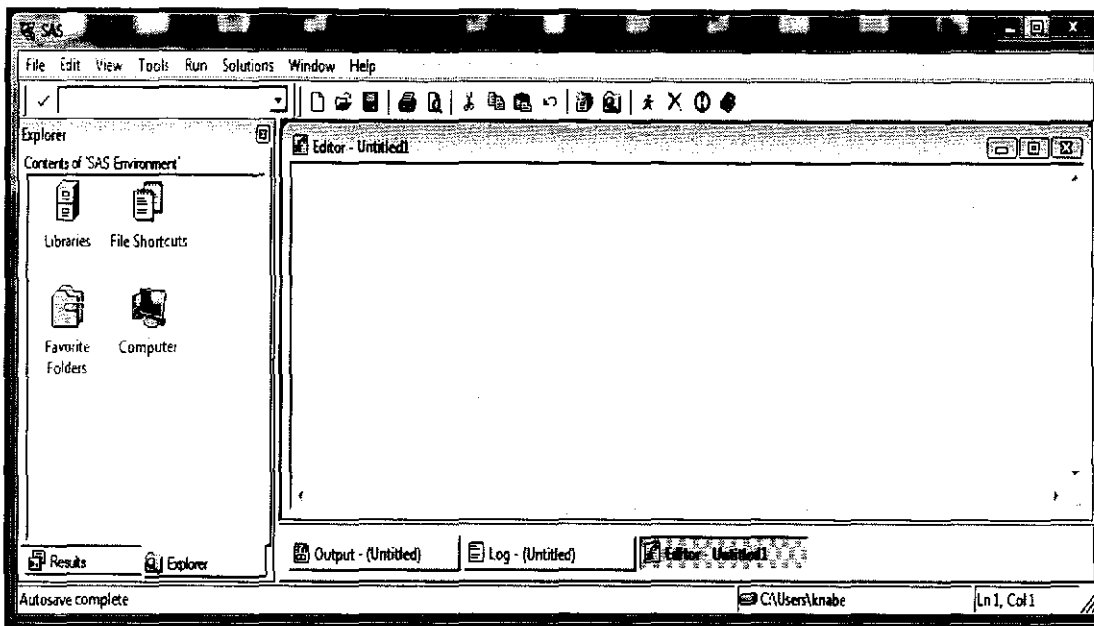
Once a program has been written and saved, the program name will appear next to the word Editor at the top of the window (replacing the text "Untitled1"). You may then recall the program at any time to edit it and rerun it. The Enhanced Editor works almost exactly like Microsoft Word; you can cut/copy/paste, move the cursor, etc. The Enhanced Editor will give you color-coded procedures, statements, and options that may help you find errors in your program before you even run it. An example of how to use this color coding for error checking will be provided in a later section of this manual.

COLOR	COMMAND TYPE	EXAMPLE
BOLD BLUE	Major SAS commands	DATA
ROYAL BLUE	Sub commands, and recognized SAS words	INFILE STUDENT
PURPLE	Words within quotes such as filenames or titles.	'C:\My Documents\DATA.DAT'
BOLD GREEN	Numbers	1-20
GREEN	Commented out commands	*PLOT;
RED	Errors	TALBE
CALORIES	All user defined words such as variable names	CALORIES RESDAT1

**Table 1. SAS Enhanced Editor Color Codes**

Tips and Tricks
The Enhanced Editor color-codes your program for you as you type. This feature is invaluable! Even though a separate Program Editor window is also available in SAS, we highly recommend you use the Enhanced Editor to create your programs. It will help you avoid being frustrated by small syntax errors that can prevent your programs from running properly.

The Log Window

The Log window opens directly above the Editor. Once you have run a SAS program, important information is automatically displayed in the log window. This window will contain a running list of all of the SAS commands that have been executed, and will include information about any errors that occurred during the program's execution and the reason(s) for the errors. For example, when reading in a data file, the Log window will display how many records were read in, and how many variables were contained in each record, allowing you to confirm that the data were read in correctly. It is always a good idea to inspect the Log window to confirm that your program ran as expected before you are tempted to believe the results of your analyses. The Log window is extremely important if you hope to figure out what is wrong with your program.
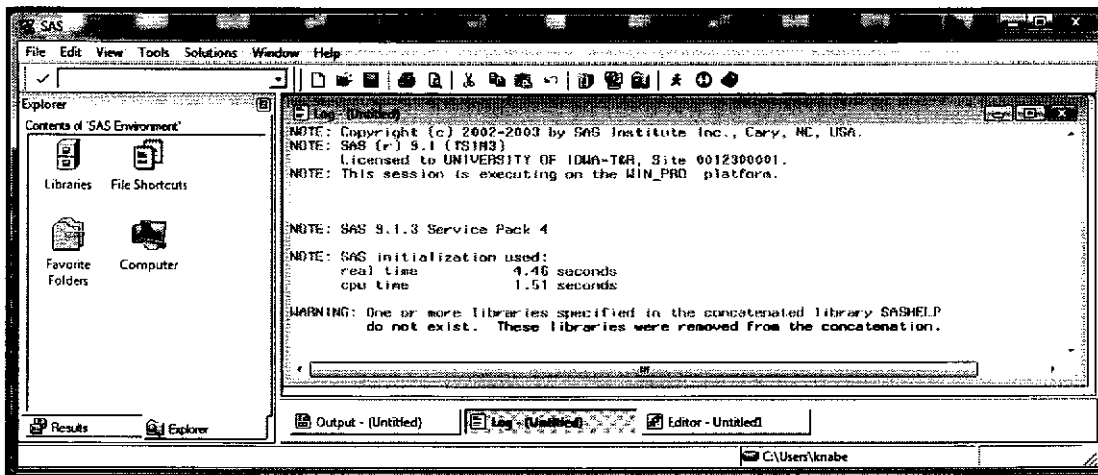
The Output Window

As you might expect, when SAS successfully executes a program file, the results of your analyses are displayed in the Output window. As you submit future programs, the results will be added to the end of the Output, giving you a running history of the results for each SAS session. You can, of course, clear the contents of the Output window, or any other active window, at any time (which is often useful if there were errors in your program that make the results invalid). You can also cut/copy/paste, etc. from the log and output windows.
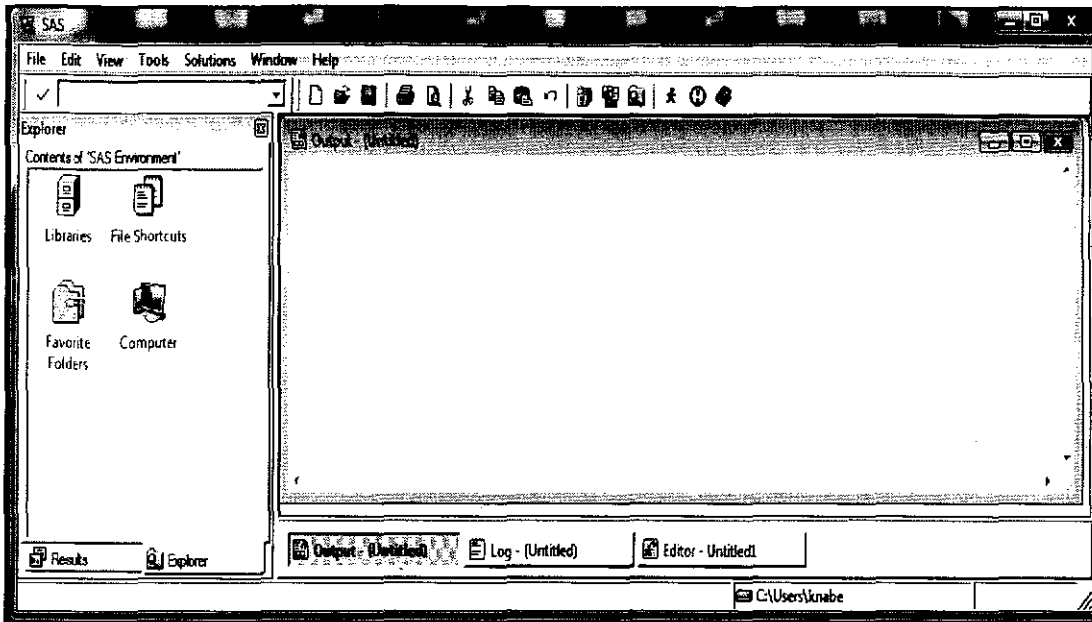
Tips and Tricks:
To clear the contents of any active window, place the cursor in the window and press CTRL-E. This shortcut is quite useful, especially if you run a program that produced incorrect output. You'll want to clear both the output window of the incorrect output, and clear the log of the error messages.

The Explorer Window

 Slide 9

With the explorer window, you can open\view data you have read into SAS. Click on libraries, then the work folder, and this will show you any datasets you have read into or created in SAS for that session.
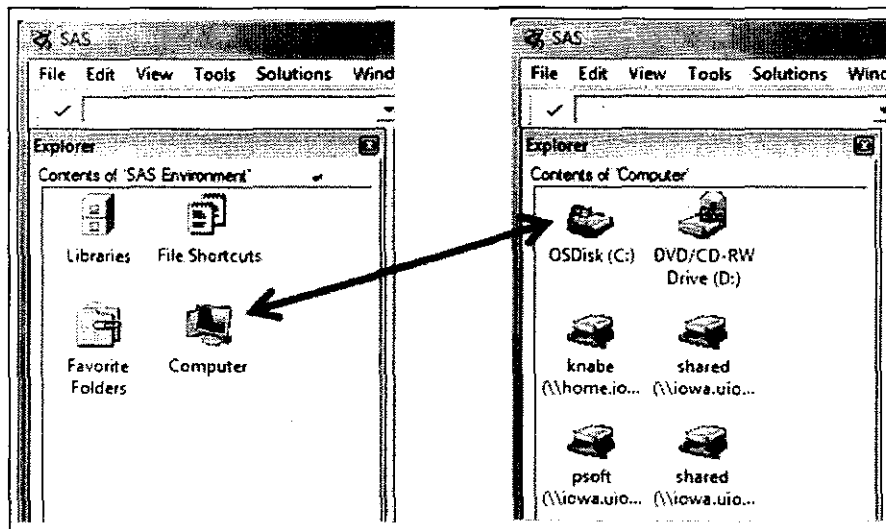
The Results Window Slide 10

The Results window provides a table of contents for your output. It lists each procedure that produces output in outline form and can be expanded to show each part of the procedure output. This window can be very helpful when you have a lot of output and you wish to view a particular section or component of the output.

Class Exercise

Section II. Understanding SAS System Workflow

Section Overview Slide 11

Learning Objectives

At the conclusion of this session, participants should be able to:

- demonstrate understanding and knowledge of basic SAS operations and procedures
- develop a step-by-step approach to conducting a SAS analysis

Section Outline

- Becoming a SAS Programmer
- Writing a SAS Program
- A Simple SAS Program Explained
- Data Value, Variable, Observations, and Dataset
- Rules for SAS Names
- Rules for SAS Statements
- Required Structural Components
- Class Exercise

Becoming a SAS Programmer Slide 12

Though the SAS system does provide a menu-driven interface (SAS Enterprise Guide), most interactions with the SAS system, particularly in research applications, is done through the writing of SAS programs. SAS programs provide a high level of flexibility to the user. Also, platforms like Unix and Mainframe do not provide a menu driven interface for SAS. To use SAS it is necessary that you become acquainted with the SAS language. SAS programs are written using the SAS language to manipulate, clean, describe and perform data analysis.

Writing a SAS Program

 Slide 13

To become proficient in the use of SAS you must learn how to write a SAS program. This requires familiarity with a special scripting language. Writing a SAS program can be easy if you understand the basic requirements or have prior programming experience.

A Simple SAS Program Explained

 Slide 14

Let's start with a simple example to see how SAS reads and understands data and then performs the request set of operations. Below is a table that provides some information about 10 accounts holders of a credit card company. Information includes Account Number, Account Open Date, Account Status Code and current Credit Limit. Now let us look at some data features.

Account #	Open Date	Status Code	Credit Limit
1234670	11-Sep-04	Z	2000
1234671	12-Sep-04		3000
1234672	13-Sep-04	Z	2500
1234673	14-Sep-04	T	3200
1234674	15-Sep-04		8000
1234675	16-Sep-04	D	2000
1234676	17-Sep-04		4000
1234677	18-Sep-04	S	6000
1234678	19-Sep-04	T	8000
1234679	20-Sep-04	T	2000

Data Value, Variable, Observation, and Dataset

 Slide 15

Data Value. Data value is the basic unit of information. In the field containing information about the account holder's credit limit, the DATA VALUES are 2000, 3000 etc. The DATA VALUES in the field 'status code' are 'Z','D', 'S' and 'T'.

VARIABLE. A set of data values that describes a given attribute makes up a VARIABLE. Each column of data values is a VARIABLE. For example, the first column in our data set is reserved for the VARIABLE we'll call Account #. It has all the account numbers of the sample we have for the credit card holders

SAS variables are of two types - numeric and character. Values of numeric variables can only be numbers or a period (.) for missing data. Character variables can be made up of letters and special characters such as plus signs, dollar signs, colons and percent signs, as well as numeric digits.

In the sample data above, account number and credit limit are numeric variables and status code is a character variable. Open date is a field that deserves special mention. In SAS, dates are stored as numeric but displayed in various format using SAS formats.

OBSERVATION. All the data values associated with a case, a single entity, a subject, an individual, a year, or a record and so on, make up an OBSERVATION. Each row of the data table (or Matrix) represents one OBSERVATION. The row below represents all the data values associated with OBSERVATION #1.

Account #	Open Date	Status Code	Credit Limit
1234670	11-Sep-04	Z	2000

DATA SET. A DATA SET is a collection of data values usually arranged in a rectangular table (or matrix).

A SAS DATA SET is the special way that SAS organizes and stores the data. For example, if we convert our sample into a SAS dataset we will have a data set with 4 columns (fields) and 10 rows with 3 numeric fields and 1 character field. Why three numeric fields? SAS stores date as a number.

The DATA step creates the SAS data set and the PROC steps are instructions indicating how the SAS data set is to be manipulated or analyzed. There are certain procedures where the outcome of their execution results in creation of one or many datasets.

Rules for SAS Names   Slide 16

Among the kinds of SAS names that appear in SAS statements are variables names, SAS data sets, formats, procedures, options, and statement labels.

1. Many SAS names can be 32 characters long; others have a maximum length of 8.
2. The first character must be a letter (A, B, C. . ., Z) or underscore (_). Subsequent characters can be letters, numeric digits (0, 1. . ., 9), or underscores.
3. You can use upper or lowercase letters. SAS processes names as uppercase regardless of how you type them.
4. Blanks cannot appear in SAS names.
5. Special characters, except for the underscore, are not allowed. In file reference, you can use the dollar sign ($), pound sign (#), and at sign (@).
6. SAS reserves a few names for automatic variables and variable lists. For example, _N_ and _ERROR_.

Rules for SAS Statements   Slide 17

1. SAS statements may begin in any column of the line.
2. SAS statements must end with a semicolon (;).
3. Some SAS statements may consist of more than one line of commands.
4. A SAS statement may continue over more than one line.
5. One or more blanks should be placed between items in SAS statements. If the items are special characters such as '=', '+', '$', the blanks are not necessary.

Required Structural Components   Slide 18

Every SAS program you write will have at least two fundamental components:

- **DATA step** -- reads the data from the raw data file where it has been stored into the program and carries out any variable manipulation(s) that have been requested.
- **PROC step** -- performs the particular analyses that have been selected; SAS does not work directly on the original data file; the PROC step uses data that are stored in a temporary SAS (.sas) data set created by the DATA step.

The DATA step is that part of the program where the structure for the data to be analyzed is created. Variables corresponding to the various elements of the data set are defined, and the data are assigned to the Variables. The data may be input manually in the body of the program, or they may be read in from a file. Additionally data may be extracted from a data warehouse where it is being stored.

The PROC (PROCedure) step provides access to a series of procedures, or PROCs, each of which is dedicated to a particular form of data manipulation or statistical analysis to be performed on the data set(s) created in the DATA step. Some examples of PROCs and the operations they perform are:

PROC PRINT: Prints the contents of a data set and create reports.
PROC FREQ: Produces frequency and cross tabulation tables on the variables specified.
PROC MEANS: Computes means, standard deviations and other summary Statistics.
PROC TTEST: Computes the 2-sample t-test for comparing the means of 2 treatments.
PROC REG: Performs regression analysis using the method of least squares.
PROC GPLOT: Constructs plots of the data as specified by the user.

Stated in very simplistic terms, a DATA step deals with data cleaning and data formatting, and a PROC step performs some required statistical analyses and/or graphic representation of the results. Data steps are important for several reasons. First, the dataset may not be in a SAS compatible format, although this is usually not the case for the datasets in class examples or exercises. Second, sometimes you need to extract some of the variables or some of the observations from the dataset to perform analysis. Third, different procedures may require the same dataset in different format. A data step is needed to transform the dataset into the appropriate format for a procedure. Hence a SAS program may have more than one DATA step and more than one PROC step.

In the sessions that follow, you will have an opportunity to become more familiar with DATA and PROC steps and various features and options that are available with each one of them. Examples and code samples will also be provided to give you a better understanding of how these steps work together to produce the desired result.

Class Exercise

Section III. Getting Data into SAS

Section Overview

Learning Objectives

At the conclusion of this session, participants should be able to:
- demonstrate knowledge and understanding of how to read data into SAS
- provide an operational view of how SAS uses DATA, INFILE, and INPUT statements to perform data selection
- characterize the basic structural components of the DATA and PROC steps

Section Outline
- The SAS Data Step
- Reading Data into SAS
- Raw Data Files
- Writing an INFILE Statement
- Writing an INPUT Statement
- Using a CARDS or DATALINES Statement

- Importing Data from External Sources
- Handling Missing Data
- The SAS PROC Step
- PROC Syntax Commonalities
- PROC Statements and Options
- PROC PRINT
- PROC UNIVARIATE
- PROC FREQ
- Creating a Permanent SAS Dataset
- Important Programming Tips
- Class Exercise

The SAS Data Step

The DATA statement signals the start of the data step. The key word DATA is required, and should be followed by a name you choose for the temporary data set that SAS creates and uses as your program runs. The DATA statement has as its sole function to read and modify data. The DATA step can include DO loops, IF and IF-THEN/ELSE statements, and an assortment of numeric and character functions. DATA steps can also combine data sets by using concatenation and match-merge operations. A DATA step ends when SAS encounters a RUN statement or a new step (marked by a DATA or PROC statement).

A typical SAS program starts with a DATA step to input or modify data and then passes the data to the PROC step which analyzes the data, performs utility functions, or prints reports; but that is certainly not the only pattern for mixing DATA and PROC steps. DATA steps execute line by line and observation by observation.

Examine the program given below. In this example, the temporary data set is named CYRIL.

Examine the following simple SAS program code based on the infamous *Cyril.dat* file on twin IQ provided by Dr. Susan Graham at Harvard University:

```
DATA CYRIL;
   INFILE 'c:\my documents\S-052\CYRIL.DAT';
   INPUT ID FOSTIQ OWNIQ;
   LABEL ID      = 'id number'
         FOSTIQ = 'IQ of twin raised in foster family'
         OWNIQ  = 'IQ of twin raised in own family';
RUN;

PROC UNIVARIATE PLOT;
   TITLE1 'Descriptive statistics for OWNIQ and FOSTIQ';
   VAR FOSTIQ OWNIQ;
   ID ID;
PROC PLOT;
   TITLE1 'Plot of FOSTIQ vs OWNIQ';
   PLOT FOSTIQ * OWNIQ;
RUN;
```

Other optional statements that allow you to create new variables, add labels, and execute other functions can be added to the data step.

The SAS data set that is created is what is called a "working data set" and will only last as long as the current SAS session. When you quit SAS, the working data set will be erased, but the raw data file from which it was created will, of course, remain saved and unchanged. To recreate your SAS data set, you must simply rerun the SAS program file that created it.

Reading Data into SAS

SAS is capable of reading data files created by many different programs and converting them to SAS data sets. The most common format is "raw" data files. By that we mean that the data are simply entered and stored in a text file (sometimes referred to as an ASCII file). In it, the data are organized either by columns (with column numbers) or by being separated by a particular delimiter (like a space) or several delimiters. In either case, the file contains only the data; it does NOT include any variable names, variable or value labels, or any other formatting. These files are the simplest way to store data, and many programs can convert files to this simple format, allowing you to use the programming methods described here to analyze data from a wide variety of sources.

The most straightforward method for creating a raw data file from scratch is to enter it into any word processing program (Word, Notepad, etc.), using either the **column input format** or the **list input format** that were mentioned above. Both methods require that the variables are entered in order, and the records are separated by a return. Depending on the data format you use, the commands that you will use to read in the data set are slightly different, as you will notice below.

Below are five records of a data set called JUNK.DAT with data corresponding to different types of junk food. First the data were entered using column input format, and then using list input format.

```
Big Mac 563 26.0 8.25 53 1010 1.43 1
Whopper 670 27.0 9.50 51 975 1.60 1
Wendy's Double 560 41.0 8.25 54 575 2.05 1
McDonald's Hamburger 255 12.0 2.50 35 520 0.54 1
BK Hamburger 310 16.0 3.00 35 560 0.70 1

BigMac 563 26.0 8.25 53 1010 1.43 1
Whopper 670 27.0 9.50 51 975 1.60 1
Wendy'sDouble 560 41.0 8.25 54 575 2.05 1
McDonald'sHamburger 255 12.0 2.50 35 520 0.54 1
BKHamburger 310 16.0 3.00 35 560 0.70 1
```

In the first data set, columns 1-20 were assigned to the variable ITEM, which represents the name of the item. The number of CALORIES is in columns 22-24, PROTEIN is in columns 26-29, FAT in columns 31-34, FAT_PC in 36-37, SODIUM in 39-42 and COST in 44-47. The information contained in column 49 represents the TYPE of food (burger, chicken, fish, potatoes, shakes, breakfast, or specialty). It is coded using a numeric code for each of these categories.

In the second data set, these same variables appear in the same order. However, each time there is a space SAS knows that it is to put the value in a new variable. To accomplish this, none of the Item names could have spaces, since SAS would interpret this as a sign to move on to the next variable.

In the sections below, we will outline how to read in data sets in both of these formats using a Data Step.

Raw Data Files

Most common raw data files are flat-files, where each row contains all the data for a particular case. Raw data files are usually organized in variables (columns) x cases (rows) format. There may be a header line that

contains variable names that can be read or omitted. Raw data files may be internal or external to the SAS program.

To create a SAS data set from a raw data set, you need to create a SAS program with at least the following three SAS statements: a DATA statement; an INFILE statement; and an INPUT statement.

To read a raw data file using SAS you need to create a SAS program file (see below). To read our raw data file called JUNK.DAT, we created a SAS program file called JUNK.SAS. These names were assigned to help us remember what the program was for and the data set with which it went. These names do not have to match, and you can assign whatever program names you feel are appropriate for your purposes.

The program begins with a DATA step. The first line of the DATA step is the DATA statement. The DATA statement tells SAS that you would like to create a SAS data set with a specific name (in our example, JUNK). This is a temporary name that you choose; oftentimes it is convenient to use the same name as that of your data file, but it is not necessary to do so.

DATA JUNK;

    INFILE 'C:\WINDOWS\DESKTOP\JUNK.DAT';

    ITEM $ 1202224 PROTN26-29 FAT 31-34
    INPUT- CALORIES -EI FAT_PC 36-37 SODIUM 39-42 COST 44-47 FOOD 49;
RUN;

> **Tips and Tricks:**
> Data set names **must** begin with a letter, and must not contain any spaces. SAS can use names that are longer than 8 characters, but we would advise that you keep your names short and meaningful.

Writing an INFILE Statement

The INFILE statement gives SAS the name and physical location of the file that contains the data you wish to use. The physical location must begin with the root directory and include specification of each level [folder] required to reach the desired file. Current versions of SAS (9.1 and 9.2) can accommodate the long folder names often used to organize files.

The INFILE statement follows the DATA statement and must precede the INPUT statement. In some operating environments, SAS assumes data files have a maximum record [number of characters, including spaces, in a data line] length of 256. If SAS is not reading all of your data, you can add the LRECL= option to the INFILE statement to specify a record length at least as long as the longest record in your date file. Check the SAS log window to see if the record length is sufficient to read all your data.

```
INFILE 'c:\MyRawData\kunk.dat' LRECL = 2000;
```

Using an INFILE statement will bring data in from a different drive (i.e., drives A, H, C, F) and make them available for the entire SAS session. For example, if your data were saved on your system hard drive, the commands would be:

DATA a1;

INFILE 'c:\filename';

INPUT x1 x2;

You also need to name your dataset. You can name it anything you like; here it is named a1. The *input* statement identifies the variables in your dataset so you can use them for analysis. They can also be named whatever you would like; here they are named x1 and x2.

```
Tips and Tricks:
-- Use the colors to be sure that your program is written correctly! For example, if the word INFILE does
   not appear in royal blue, you've forgotten your semi-colon for the preceding command.
-- Check your log file! If your path and file name are not specified correctly, your log will produce an error
   that reads "Physical file does not exist." This will alert you that SAS did not find the file in the location
   that you specified.
```

Writing an INPUT Statement

The INPUT statement specifies the location of each variable in the raw data set. The INPUT statement begins with the word INPUT followed by the names of the variables in the data set in the order in which they appear. The INPUT statement allows you to read data in one of four formats:

```
Format                          Example
List            INPUT Item $ Calories Protein Fat Fat_PC;
Column          INPUT Item $ 1-20 Protein 26-29 Fat 31-34;
Informat        INPUT Item $20. +5 Protein. @31 Fat.;
Mixed           INPUT Item $ 1-20 Calories Protein @31 Fat +1 Fat_PC.;
```

The List format requires at least one space between each data value. The $ indicates that the data value is character. The Column format does not require spaces between data values, missing values can be left blank, character data can have embedded spaces, and you can skip unwanted variables. The Informat format name ends with a period (.) but it is not a requirement.

The INPUT statement lists the variables in the data set in the order in which they appear there, separated by a space. Be sure to include all of the variable names in the EXACT order in which they appear in the raw data file (even if you do not plan to use all of the variables in your analysis).

The INPUT statement specifies the location of each variable in the raw data set. Input statements begin with the word INPUT, followed by the names of each variable in the data set in the order in which they appear. The program above is written for the data set entered using the column input format, and therefore includes the column numbers where SAS can expect to find the variables. If we were reading in the data set entered using the list input format, we could eliminate these column numbers.

SAS distinguishes between two types of variables: character variables and numeric variables. Character variables contain information that includes a combination of letters, numbers and/or symbols. Numeric variables contain only numbers and conform to certain SAS conventions. Variable names must begin with a letter or underscore (_), and contain no blank spaces. In V8, SAS can accommodate variable names that are longer than 8 characters. However, to make programming easier, we recommend that you keep your variable names as short as possible.

Using a CARDS or DATALINES Statement

Another option is to type or paste your data set directly into the SAS editor. This generally works best when you have a small data set (25 lines or less). Often used when testing SAS program on a data subset of a large file. Use the CARDS or DEATALINES statement to inform SAS that your data is internal rather than external. The DATALINES statement must be the last statement in the DATA Step. It is very import for the last semicolon to appear on the next line after all your data has been listed. It you forget it, your last observation could be deleted because SAS reads all characters as data until it encounters a semicolon (;).

Tips and Tricks:

Reading in data @@. You've already learned that when you input your dataset after a CARDS or DATALINES statement, every observation needs to be on an individual line. In case you want to make better use of the window and want to have more than one observation per line, @@ is the syntax that tells SAS where the end of one observation is. For example:

```
data b1; input x y z @@; cards;
1.1 2.2 3.3 4.4 5.5 6.5 ;
```

```
/* Read internal data into SAS dataset junk */
DATA junk;
    INPUT Item $ Calories Protein Fat
          Fat_PC Sodium Cost Food;
    DATALINES;
Big Mac              563 26.0 8.25 53 1010 1.43 1
BK Whopper           670 27.0 9.50 51 975  1.60 1
Wendy's Double       560 41.0 8.25 54 575  2.05 1
;
RUN;
```

The DATA Step ends with the command RUN followed by a semicolon. This signals to SAS that it should run those commands before proceeding on to the analysis section of the program. This statement is not critical, but it will make your programs run better. SAS will continue to use up temporary memory until it encounters a RUN statement. If you try to execute too many commands before giving SAS a RUN statement, it may run out of temporary memory and be unable to continue.

Importing Data from External Sources

In SAS it is possible to import data from other sources such as Excel, Access, Word, etc. Data values can be space, comma, or tab delimited. SAS provides an *Import Wizard* that takes you step-by-step through the

---

process. From any SAS window, you can open the file menu in the main menu bar and select IMPORT DATA. The imported data can be saved in a SAS Work library or a SAS permanent library.

> Tips and Tricks:
> What if my Excel data file is not reading properly into SAS or not at all? If the Excel data file is not reading into SAS at all, most likely it's because your Excel data file is open. The Excel file must be closed before you import it into SAS. There are other reasons the Excel data file is not reading in properly. It could be that the data type of your Excel cells is not correctly defined. Inappropriate reading also happens when you do not have a header in the first row, since the import procedure takes the first row as header by default. However, this can be changed during the import procedure under options.

Handling Missing Data   Slide 26

Whenever SAS encounters an invalid or blank value in a data field, the value is defined as missing.   In SAS missing numeric data are represented by a single period (.) and missing character data appear as blanks in the data field.   When writing recoding statements in the DATA step, use a period to refer to missing numeric values. For example, if you wanted to recode missing values in a variable FAT to the value 99 you would write the following IF-THEN statement:

```
IF FAT=. THEN FAT=99;
```

When you want to assign certain characters to represent special missing values for all numeric variables you can use the MISSING statement. The missing values can be any letter in the alphabet or an under-score. For example, the values 'a' and 'b' will be interpreted as special missing values for every numeric variable in the data set if you include the following statement:

```
MISSING a b;
```

If your data set has missing values, put they are not represented by a period, you can add a period to the corresponding missing value locations using a data step. For example, if you have two variables, X and Y, in your data set, and 10 observations. The ninth value of Y is missing. The following code with an if statement will do:

```
data a2; set a1; If _n_ eq 9 then Y=.
```

The SAS Proc Step   Slide 27

Following the DATA step is one or more procedures (or PROCs) that perform the desired statistical analyses. As previously noted, each procedure is a unit, although some are needed to run others. Some often-used procedures for statistical analysis will be explained in detail in later sessions.

Although statements and options vary from one PROC to another, the basic PROC structure is something like the following:

```
PROC _____ DATA=_____ ;
TITLE _____ ;
FOOTNOTE _____ ;
BY _____ ;
LABEL _____ ;
FORMAT _____ ;
RUN; < and/or > QUIT;
```

SAS procedures begin with the keyword PROC which signals to the SAS system that a "canned" program is being launched. Once the name of a PROC is specified, you may then specify one or more options available within the PROC, and in any order. The DATA= option informs the SAS system what data set is to be used as input to the PROC. If omitted, SAS automatically defaults to the most recently created data set, which may not be the most recent one used.

PROC Syntax Commonalities

 Slide 28

There are syntax rules in effect when learning to write SAS procedure codes:

-- All PROCs being with a procedure name followed by procedure options; statements that define the PROC are followed by statement options.
-- Variables are listed in the order in which they are executed.
-- Dependent variables are listed first followed by independent variables which are sometimes preceded by a keyword like BY or WITH.
-- Interaction effects between variables can be requested using a vertical (|) symbol or an asterisk (*).
-- The CLASS or CLASSES statement always appears before the MODEL statement in the ANOVA or MANOVA procedures.
-- If you omit the data = command in the PROC statement, SAS will use the last data set created in the Data step.

PROC Statements and Options

 Slide 29

PROC statements are commands nested within a procedure that tells SAS what operations to perform and in some cases allows you to make your analysis more specific. Some PROC statements are necessary while others are non-compulsory. Options are commands that further describe a statement and in some cases may also further describe a procedure. The SAS language has three basic types of options: system options, statement options, and data set options.

System options (identified by OPTIONS statements) appear in the DATA step and usually stay in effect for the duration of the session. They are usually placed in the first line of the program so you can quickly see what options are in effect. Statement options appear in individual statements and influence how SAS runs a particular DATA or PROC step. Data set options affect only how SAS reads or writes an individual data set.

PROC PRINT

Application:

    View the observations in a SAS data set

Syntax:

```
PROC PRINT {options} DATA=filename{(OBS=100)};
  {TITLE 'Title of Output';}
  {BY variables;}
  {ID variables;}
   VAR variables;
  {SUM variables;}
  {SUMBY byvars;}
  {PAGEBY byvars;}
  {FORMAT variable numberfmt./$charfmt.;}
RUN;
```

Discussion:

    PROC PRINT allows you to print data generated by another SAS procedure with more control over the output variables and layout.

PROC UNIVARIATE

Application:

    Calculates descriptive statistics, particularly details on data distribution

Syntax:

```
PROC UNIVARIATE {options} DATA=filename;
  {BY variables};
  {WEIGHT variable;}
   VAR variables;
  {OUTPUT OUT=newfile statistics;}
RUN;
```

Discussion:

    PROC PRINT produces output similar to PROC MEANS except it provides a larger number of descriptive statistics. As with PROC MEAN, analysis can be performed using a numeric or character categorical variable in a CLASS statement. This produces the descriptive statistics by subgroups.

PROC FREQ

Slide 32

Application:

Calculates frequency table for the values of a numeric or character variable and cross tabulation of two or more variables in a data set

Syntax:

```
PROC FREQ DATA=filename;
  {TITLE 'Title of Output';}
  {BY variables;}
  {WEIGHT variable;}
   TABLES variable{*variable} {/ list NOPRINT OUT=newfile};
  {FORMAT variable numberfmt./$charfmt.;}
RUN;
```

Discussion:

In its simplest form PROC FREQ produces a one-way frequency table. To produce a crosstabulation table for two or more variables you will need to specify the variable names separated by an asterisk (*).

Creating a Permanent SAS Dataset

Slide 33

A SAS data set can be created as either temporary or permanent. A temporary SAS data set is one that exists only during the current session and is automatically erased by SAS when the session ends. A permanent SAS data set remains after the session ends and can be used again in subsequent session. If you plan to use a SAS data set more than once it is more efficient to save it as a permanent SAS data set. Permanent SAS data sets are saved in a library using a library reference (*libref*) location defined by a LIBNAME statement. The libref can change but the program must point to the same library and member name.

```
LIBNAME dietary 'c:\MySASLIB';
  DATA dietary.fastfood;
      INFILE 'c:\MyRawData\junk.dat';
      INPUT Item $ 1-20 Fat 31-34 Fat_PC 36-37;
Run;

LIBNAME example 'c:\MySASLIB';
  PROC PRINT DATA = example.fastfood;
      TITLE "Fat Content in Fast Foods";
Run;
```

Important Programming Tips

-- All statements must end in a semi-colon (;).
-- Major commands appear in **dark blue** and begin at the left margin.
-- Subcommands appear in royal blue and are indented.
-- A blank line appears after the DATA step and after each PROC step.
-- TITLE statements can be added but the title must be enclosed in quotation marks; titles appear in **purple** in the Enhanced Editor.
-- All SAS programs running in Windows environment must end with RUN statement.

Class Exercise

Section IV. Submitting a Program in SAS

Section Overview

Learning Objectives

At the conclusion of this session, participants should be able to:

- demonstrate knowledge and understanding of how to submit a SAS program for processing
- recognize the procedures SAS engages in dataset selection
- identify and correct common SAS errors

Section Outline

- Saving and Running SAS Programs
- Telling SAS which Dataset to Use
- Commenting Out SAS Commands
- Examining SAS Results
- Correcting SAS Errors
- Common SAS Programming Errors
- Using the Enhanced Editor to Find Errors
- Saving SAS Files
- Exporting SAS Files
- SAS Help and Other Resources
- Class Exercise

Saving and Running SAS Programs

To save the contents of any window to a file, under the file menu in the desired window click on SAVE AS..., type in or click on where you want to save it, name the file and click OK. Remember to SAVE the program whenever changes are made. SAS puts an asterisk after the file name in the Editor window when any changes have been made. SAS does not save the program unless it is told to save it. To run the program click on the "running person" [太] button or select the SUBMIT command under the RUN heading.

Telling SAS which Dataset to Use   Slide 37

If you are working with multiple datasets that you have output from multiple procedures (e.g., you have one data set that SAS made from a PROC GLM run and another from a PROC REG run, you must always name the data set you wish to use; otherwise SAS will use the dataset just previously used by default.

You do not have to run the entire program every time you make a correction to your SAS program. Each SAS procedure is relatively independent of other procedures. As long as you have the dataset you need in this procedure in SAS, you can run only part of the program by highlighting the part of the program you want to run and then clicking the run button in the tool bars.

When you are first beginning to program in SAS, you will likely want to submit your entire program each time you make additions or changes to your program. It is somewhat easier to think of your program as a unit, but in reality, it is unnecessary to run the entire program each time you make a change. Specifically, when you submit an OPTIONS step and a DATA step for the first time, the working data set is created. Subsequently, it is unnecessary to recreate this data set each time you wish to run another procedure (unless, of course, you make changes to the data set, such as creating new variables).

Commenting Out SAS Commands   Slide 38

Commenting out is essentially the converse of submitting only a portion of your SAS program. When running SAS it may be helpful to tell SAS to ignore parts of your program when you do not need to see all of the output. If, for example, you have just successfully programmed the first part of a lengthy programming task, you probably do not want to run this part every time you submit a program file when you are building the SAS code for the second part of the program. By commenting out the first part, you are telling SAS to ignore it, which sometimes will be more efficient than highlighting the part you do want to submit every time. The result and advantages are the same as submitting only a highlighted portion—you simply decide whether it is more efficient to only submit part of the program or to comment out the part of the program you want SAS to ignore.

There are two ways to comment out: (1) commenting out a single line and (2) commenting out multiple lines. To comment out a single line, simply place an asterisk (*) in front of it. SAS will ignore everything after the asterisk until it comes to a semi-colon. To comment out several lines of text, place the symbols /* before the text and the symbols */ after the text. SAS will ignore all commands between these two symbols. That part of the program that you have commented out will turn **green**, alerting you that it will be ignored by SAS when the program executes.

These commands are illustrated in the example below. The asterisk (*) preceding the line which reads "PLOT PROTEIN*FAT;" will make sure that this one line is not read when the SAS program is submitted. A section of code can be taken out by placing a forward slash followed by an asterisk (/*) at the beginning of the section and reversing these symbols at the end of the section (*/). In the example below, both the PROC MEANS and PROC UNIVARIATE commands will be ignored when the program is executed. The program begins again with PROC FREQ.

```
DATA JUNK;
INFILE 'C:\WINDOWS\DESKTOP\JUNK.DAT';
INPUT ITEM $ 120 CALORS22-24 PROI26-29
-IE TEN FAT 313436-37 SODIUM 39-42
- FAT_PC COST 44-47 FOOD 49 NAME 51;
RUN;
PROC PLOT DATA=JUNK;
PLOT FAT*CALORIES;
*PLOT PROTEIN*FAT;
RUN;
/*
PROC MEANS DATA=JUNK;
VAR CALORIES PROTEIN FAT;
RUN;
PROC UNIVARIATE DATA=JUNK;
VAR CALORIES PROTEIN FAT;
RUN;
*/
PROC FREQ DATA=JUNK;
TABLE CALORIES PROTEIN FAT;
RUN;
```

Examining the Results

As mentioned above, once a program has executed, information is usually written to the Log and Output windows. Always examine the contents of the LOG window to ensure that your program ran as you expected without error.. Once you are satisfied that no errors have occurred, continue on by looking at the Output window to see the results of your analysis. You may print or save the contents of either of these windows at any time. By convention, the log file is given a .log extension name, and the output file is given a .lis extension name.

> Tips and Tricks:
> How do you know if SAS is reading your dataset correctly? Use the proc contents procedure and see if the dataset in SAS is what you expected.

Correcting SAS Errors

Building SAS programs is an iterative process. You create a program, run it, find an error, correct the error (quickly, hopefully), build in more to your program, find a new error, etc. We begin with three things that will help speed this process up a bit: searching for "error" statements in your LOG file; telling SAS to only run a portion of your program, and the converse--telling SAS to ignore (this is called "commenting out") parts of your SAS file.

When you are writing SAS programs you will make programming mistakes. Mistakes are inevitable, not a sign of failure. The easiest way to find your error(s) will be to examine your LOG file and find where SAS has placed an error statement. Part V of this guide discusses how to correct common types of errors, but what you also need to know is how to quickly find where in your program you are making the error.

All programmers -- experienced and inexperienced -- make mistakes. Do not be dismayed. Debugging (finding the errors and correcting them) takes time and patience. Often another set of eyes helps. Ask someone in your class or a Teaching Fellow to review your code. Above all else, keep calm. In this section, we review the most common mistakes and provide a checklist of things to watch out for when writing a SAS program.

Common SAS Programming Errors

1. No semicolon at the end of a statement
2. Missing or mismatched quotation marks on title(s)
3. Misspelling a variable name, proc, statement or option
4. Neglecting to sort data prior to using a BY statement
5. Creating variables in a PROC statement instead of a DATA step
6. Ambiguous IF/THEN statement(s) that fail to produce the desired results
7. Path to the data file location (INFILE statement) is incorrectly specified
8. Using the letter 'O' when you mean the number 0 (zero) or vice versa
9. Forgetting to specify a variable as character (SAS assumes every variable is numeric unless it sees a $ sign)
10. Incorrect column specifications on INPUT statement producing embedded spaces in numeric data
11. Missing data not marked with a period on list-style input causing SAS to read the next data value
12. Merging data sets that are not sorted in the right order
13. INPUT statement reads past the end of a line

Using the Enhanced Editor to Find Errors

As mentioned earlier in this manual, one big advantage of the PC SAS System is the automatic color coding provided by the Enhanced Editor. Those who came before you spent many countless hours debugging their SAS programs. The color coding alerts you immediately when some of the more common SAS programming errors occur. Hence you may avoid many of the more frustrating errors that plagued your predecessors.

The program below illustrates the various colors and how they identify three very common SAS errors. (You might want to reference Table 1 in the first section of this manual as you study this example.)   First, notice that the Major SAS commands are in bold blue and the sub commands and SAS words are in royal blue. If these commands were not in these colors, we would know that we had made a mistake. For example, in the last procedure, the sub command TITLE2 appears in black. Upon closer inspection, we realize that the semicolon has been left off the preceding command.

Also, all of the titles and file names are in purple. A common mistake occurs when the programmer forgets to close the quotation marks around a title. This is what has happened in that same TITLE command in the last procedure. The RUN; command is now in purple, because leaving off the second quotation mark has caused SAS to consider it part of the title. If a command appears in purple unexpectedly, look for a missing quotation mark.

Other errors, such as spelling errors, often appear in red. If you misspell a known SAS word or command (such as the misspelling of the sub command TABLE in the last procedure), SAS will alert you. It will NOT, however, correct you if you misspell a variable name, or other user defined word.

```
TITLE1 'ANALYZING OUR FAVORITE JUNK FOOD';
DATA JUNK;
INFILE 'C:\WINDOWS\DESKTOP\JUNK.DAT';
```

```
ITEM $ 20 CALORS-24 PROI26-29
INPUT1-IE 22TEN FAT 313436-37 SODIUM 39-42
- FAT_PC COST 44-47 FOOD 49 NAME 51;
RUN;
PROC PLOT DATA=JUNK;
PLOT FAT*CALORIES;
LOT PROTEIN*FAT;
*PRUN;
PROC MEANS DATA=JUNK;
R CALORIES PROTEIN FAT;
VARUN;
;*
PROC UNIVARIATE DATA=JUNK;
VAR CALORIES PROTEIN FAT;
RUN;
*/
PROC REG DATA=JUNK;
MODEL COST=CALORIES/P R ;
TPUT OUT=RESDAT1 R=RAWRES1 STUDENT=STDRES1;
OURUN;
PROC FREQ DATA=JUNK;
TALBE CALORIES PROTEIN FAT
TITLE2 'FREQUENCY TABLE;
RUN;
```

Using the Log File to Find Errors

Contained in the LOG file is a printout of your SAS program file. When SAS detects a programming error during program execution you will receive a somewhat cryptic error message in your LOG file. The error message includes:

- the SAS error number

- a brief note explaining the error

- the location of the error

Each type of programming error has a number. The number of the error will be listed in the error message. For the most part, knowing the error number will not be very helpful to you. If you would like, however, you can look up the error in the SAS manual. The manual provides more information about each type of error.

When SAS detects an error, the error message appears at the end of the SAS step where the error occurred.. When SAS detects an error, it underlines the error in the program code. Note that the numbers next to each line of your program correspond to the line number of your program.

> **Tips and Tricks:**
> An advantage PC SAS has over its mainframe counterpart is that all the search functions used in many Windows based programs apply. Therefore, to find the word "error" in your LOG file simply go to the LOG window on your SAS screen, choose FIND under the EDIT menu (or use CTRL-F). Enter the word "error" in the text box, and SAS will bring you to the first occurrence of the word ERROR in your LOG. You should then look carefully at your SAS program to figure out why you are getting this error statement. You can continue searching for errors by clicking the "Find Next" button. Remember, finding the error statement is only part of the battle; you must then determine a resolution.

Don't forget after you find the error, you must go back and edit the program file in the Enhanced Editor to make the necessary changes. Editing the log file does absolutely no good! After making the necessary

changes to your SAS file, rerun the program and recheck the LOG file to be sure the program ran as you expected.

Correcting SAS Errors Checklist
Slide 44

✓ **Read the SAS Log**
The log has a wealth of information about your program that may be helpful in finding the source of your errors.

✓ **Test each part of the program**
Increase your program efficiency by making sure each part of the program works before moving on to the next part.

✓ **Test program using small data sets**
Use options **OBS=n** (tells SAS to stop reading data at observation n) or **FIRSTOBS=n** (tells SAS to start reading data at observation n) in a DATA or PROC step to select subset of the full data set (timesaving if you have a large amount of data).

✓ **Be observant of the colors in your program**
The Enhanced Editor color codes your program statements as you write, making it easy to discover missing semicolons etc because the rest of your program will appear in the wrong colors.

✓ **Make program errors easier to detect**
Put only one SAS statement on each line and use indentation to show the different parts of the program.

Saving SAS Files
Slide 45

If you want to save the work you've done in a session, you'll need to save the contents of each window separately. Usually, you only need to save the program; you can always run the program to reproduce the log and output. To save a program file, you'll need first to make sure the Enhanced Editor is the active window, then go to file and select the SAVE command. Similarly, you can save a log file when a Log window is active, or an output file when the Output window is active.

Exporting SAS Files
Slide 46

Exporting a SAS data set to Excel, Access, SPSS, or other software program is the opposite procedure of the import process. SAS provides an *Export Wizard* that takes you step-by-step through the following process:

Step 1. Choose the library and member name for the data set that you want to export

Step 2. Choose the type of file you want to create

Step 3. Choose the location (directory path) where you want to save the exported data

Step 4. Choose whether you wish to save the programming statements that are generated by the *Export Wizard*

If you choose not to use the *Wizard,* you can also create a SAS export program using PROC EXPORT statements. This topic will be discussed in more detail during a latter session.

SAS Help and Other Resources
Slide 47

The SAS Help menu is useful if you want to self-improve your knowledge of SAS procedures. There are two ways to obtain SAS help. One method is to go to the Help menu, then select SAS Help and Documentation. You can then go to the Index tab and type in the keyword (procedure, word or phrase) to find. If you are

looking for information about a particular procedure, SAS Help will give you the syntax of the procedure as well as some examples. If you have a specific question, you can use the Search tab, and type in the key word of your question. The other way to get help from within the SAS program is to select the Help menu , then click on SAS on the Web. This will lead you to resources (including books and training) available from SAS Institute. SAS help is also available from the SAS Institute at http://support.sas.com.

Local SAS help is also available. If you are associated with the Colleges of Medicine, Dentistry, Nursing, or Pharmacy, you can get assistance from the Biostatistics Consulting Center at http://www.public-health.uiowa.edu/biostat/biocon.html. If you are affiliated with the College of Education, you can get assistance from the Statistical Outreach Center at http://www.education.uiowa.edu/StatOutreach/.

A really excellent source of information on SAS software is *The Little SAS Book: A Primer* by Lora D. Delwiche and Susan J. Slaughter (3rd or 4th Edition). Both editions are available online for limited preview at http://books.google.com. You can purchase new or used copies from Amazon at http://www.amazon.com/books-used-books-textbooks/; for a list of useful SAS Web links check out the Michael Davis discussion at www.bassettconsulting.com/Useful_SAS_links_PhilaSUG.ppt; for SAS coding tips/techniques go to http://www.sconsig.com.sastip.htm; and for a LISTSERV based world-wide SAS discussion group that is always open for discussion check out SAS-L http://listserv.uga.edu/archives/sas-l.html.