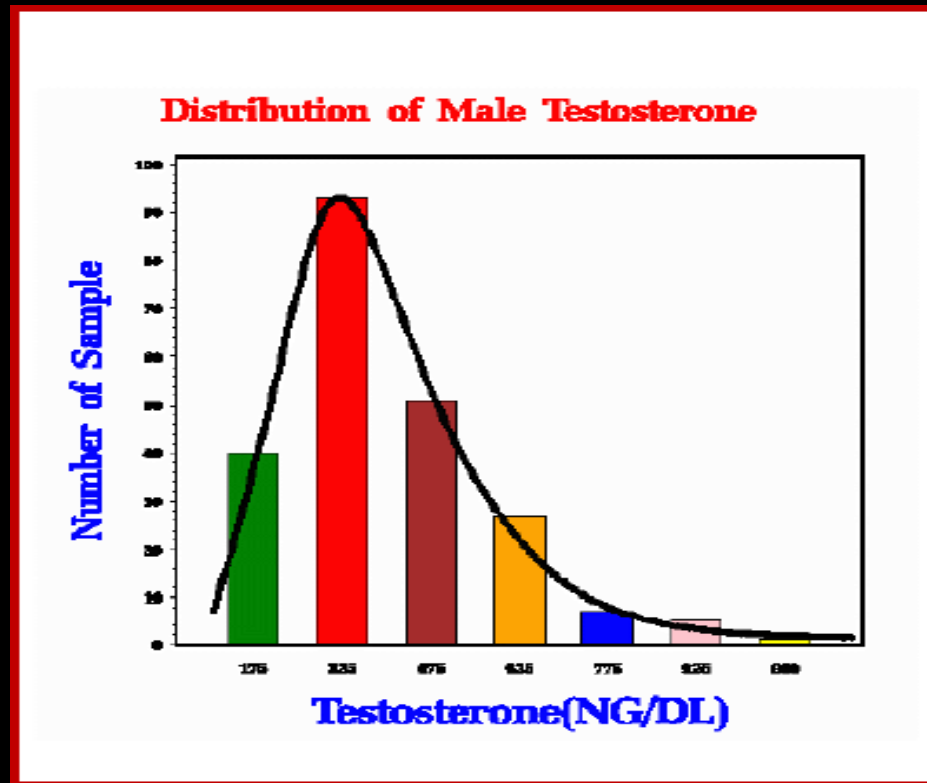


THE UNIVERSITY OF IOWA

Introduction to SAS Programming



Section I. The SAS Windows Environment

Learning Objectives

At the conclusion of this presentation, 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

In this session, we will focus on the simplest and most direct methods for reading in data, performing data transformations and conducting simple analyses.

Section I. The SAS Windows Environment

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 OS) environment.
- Access to SAS is also available on the UI campus through the Virtual Desktop -- a web-based system which allows 24/7 access to a number of software applications from virtually any computer with an Internet connection, on or off campus.

Section I. The SAS Windows Environment

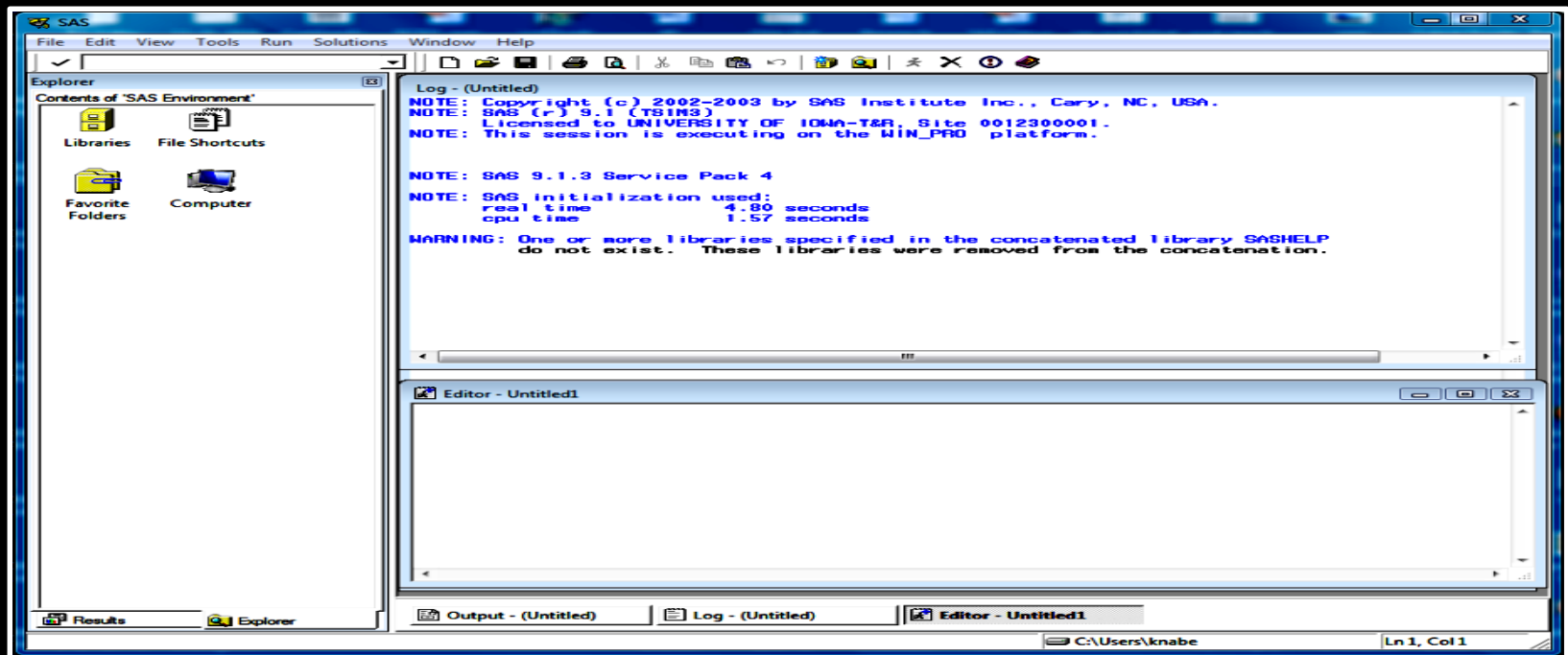
Starting the SAS System

- You can launch the SAS System by clicking first 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 *SAS 9.3 (English)*.
- 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.

Section I. The SAS Windows Environment

Getting Familiar with the Terrain

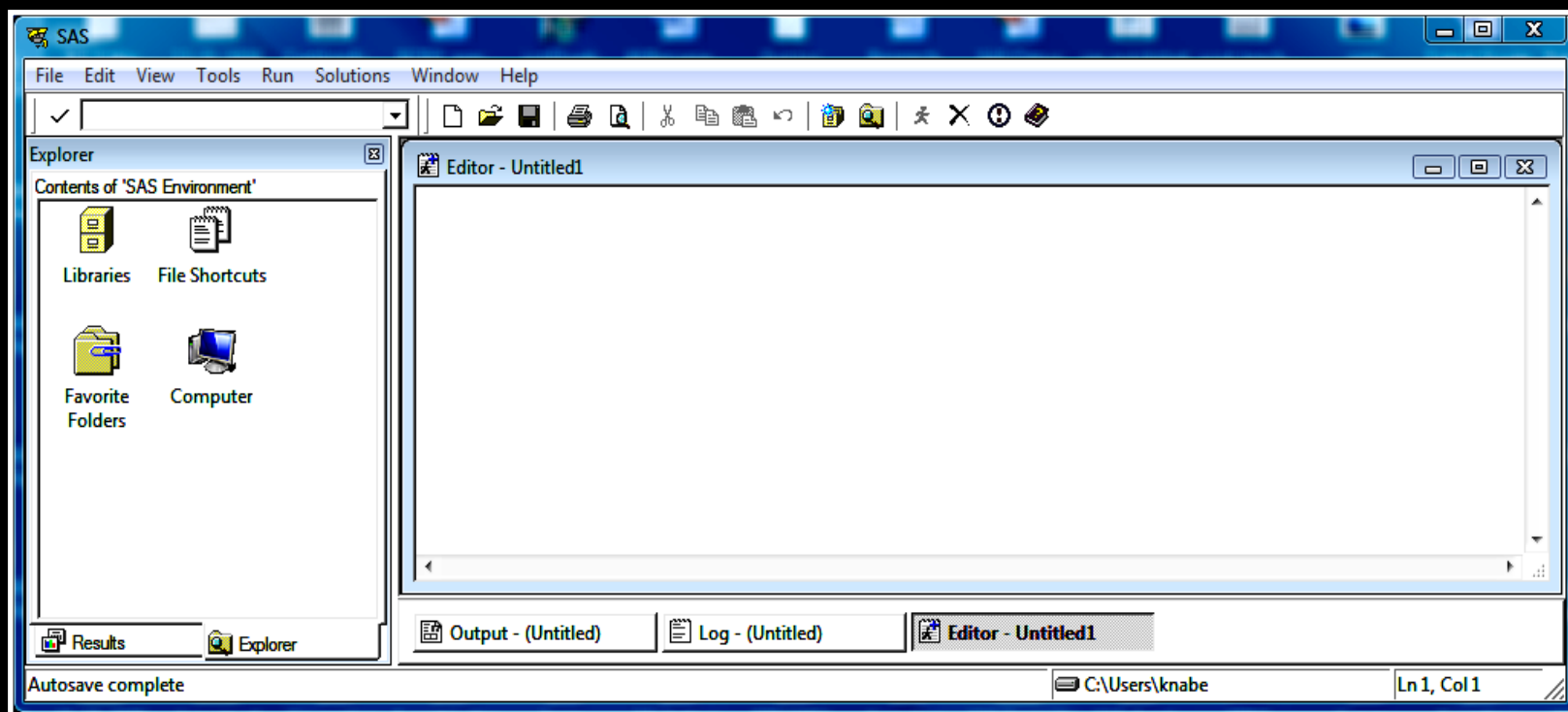
- SAS will open and automatically make available three windows: the enhanced Editor, the Log, and the Output.
- On the left hand side of the screen there is also a window labeled Explorer; this window can be helpful with file management.



Section I. The SAS Windows Environment

The Enhanced Editor Window

- The Enhanced Editor Window is where you write the SAS programs you wish to run.
- A SAS program is a series of commands that will be read into SAS, manipulate data, perform the required analysis, and output the results.



Section I. The SAS Windows Environment

The Enhanced Editor Window

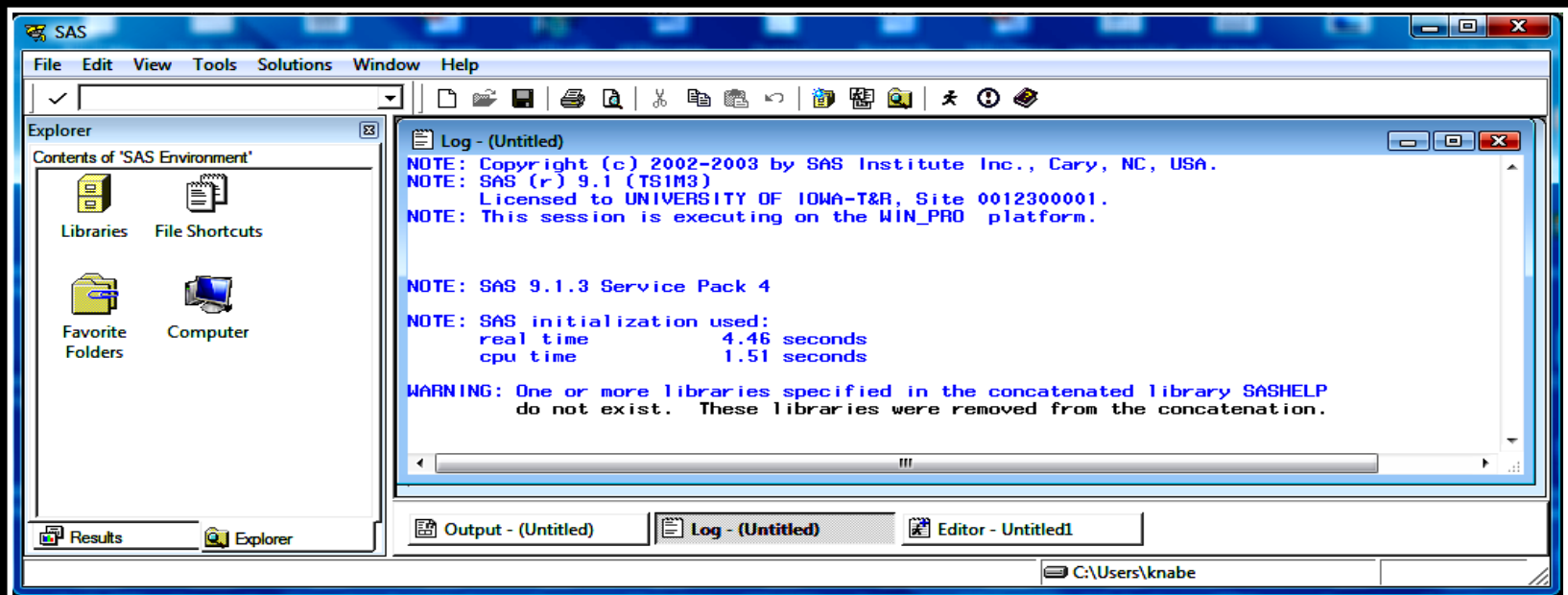
- The Enhanced Editor will give you color-coded procedures, statements, and options that may help you find errors in your program before you even run it.

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

Section I. The SAS Windows Environment

The Log Window

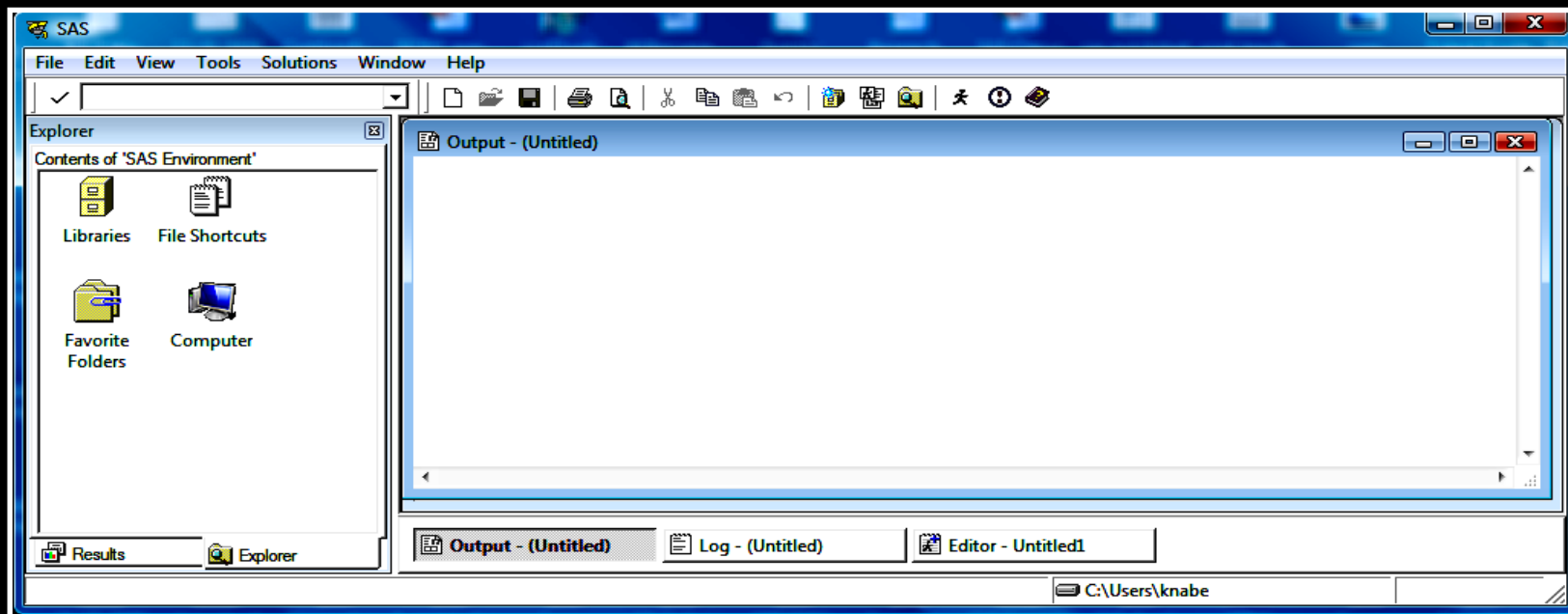
- When you run a SAS program important information about the run is automatically displayed in the Log Window.
- The SAS log contains a list of program commands and operations, as well as errors that occurred during the program execution.
- It is always a good idea to inspect the log window to confirm that the program ran without errors.



Section I. The SAS Windows Environment

The Output Window

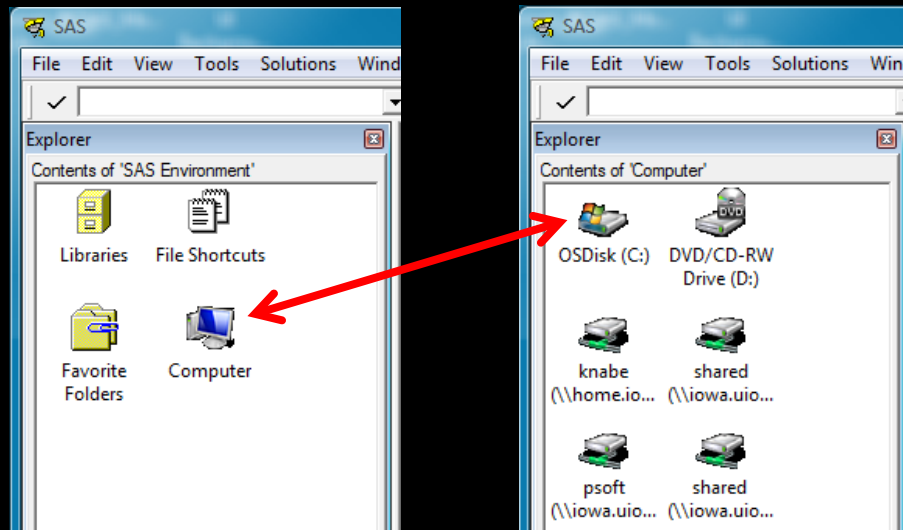
- When the program executes without error, the results of the analyses are displayed in the Output Window.
- When the contents of the Output Window contains incorrect output, the contents of the Output Window or any other active window can be quickly cleared by placing the cursor in the window and selecting Ctrl-E.



Section I. The SAS Windows Environment

The Explorer Window

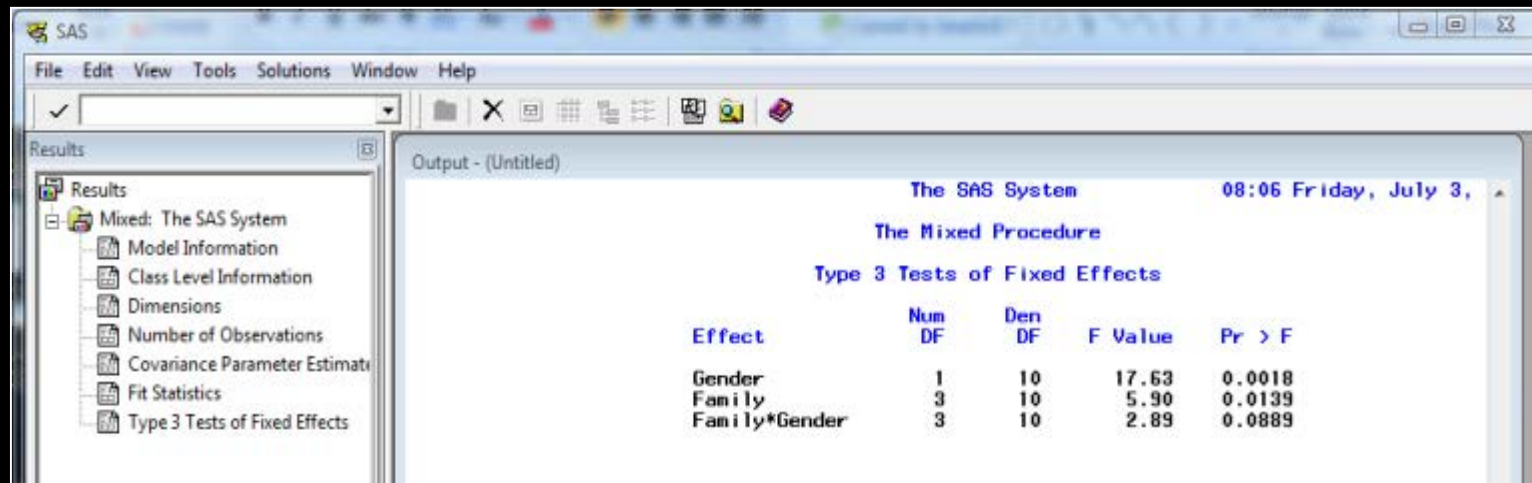
- The Explorer window provides easy access to your saved SAS files and libraries.
- Click on libraries, then the work folder, and this will show any datasets you have read or created in SAS for the current session.
- Double-click on the Libraries icon and SAS will show all the libraries that are currently defined (*Sashelp*, *Sasuser*, *Work*) and any other library for a specific SAS product (e.g., SAS /Graph *Maps*) or that has been user-defined.
- The Computer icon gives you access to all shared drives /devices affiliated with your computer.



Section I. The SAS Windows Environment

The Results Window

- Provides a table of contents for your output
- 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 of the output.



The screenshot shows the SAS Results window. On the left, a tree view under 'Results' shows 'Mixed: The SAS System' expanded to 'Type 3 Tests of Fixed Effects'. The main window displays the following output:

The SAS System 08:06 Friday, July 3,

The Mixed Procedure

Type 3 Tests of Fixed Effects

Effect	Num DF	Den DF	F Value	Pr > F
Gender	1	10	17.63	0.0018
Family	3	10	5.90	0.0139
Family*Gender	3	10	2.89	0.0889

Section II. Understanding SAS System Flow

Learning Objectives

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

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

Section II. Understanding SAS System Flow

Becoming a SAS Programmer

- SAS is best as a "write code-then-run" program; it is not a very good "point-and-click" program.
- To become proficient in the use of SAS you must learn how to write a SAS program.
- Writing a SAS program can be simple if you understand what is required.



Section II. Understanding SAS System Flow

Writing a SAS Program

- To write a SAS program 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.

Section II. Understanding SAS System Flow

A Simple SAS Program Explained

-- Below is a table that provides some information about 10 subjects who are participants in a radiation exposure study. Information includes Subject ID, Inoculation Date, Exposure Level and Reaction Code:

Subject ID	Inoculation Date	Exposure Level	Reaction Code
1234670	11-Sep-10	2000	Z
1234671	12-Sep-10	3000	
1234672	13-Sep-10	2500	Z
1234673	14-Sep-10	3200	T
1234674	15-Sep-10	8000	
1234675	16-Sep-10	2000	D
1234676	17-Sep-10	4000	
1234677	18-Sep-10	6000	S
1234678	19-Sep-10	8000	T
1234679	20-Sep-10	2000	T

Section II. Understanding SAS System Flow

Data Value, Variable, Observation, and Dataset

Data Value -- the basic unit of information.

Variable -- a set of data values that describes a given attribute makes up a variable. Each column of data values is a variable. 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.

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.

Dataset -- a dataset is a collection of data values usually arranged in in a rectangular table (or matrix).

Section II. Understanding SAS System Flow

Rules for SAS Names

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_.

Section II. Understanding SAS System Flow Rules for SAS Statements

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.
4. Multiple SAS statements may appear on a single 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.

Section II. Understanding SAS System Flow

Required Structural Components

-- Every SAS program will have at least two parts:

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.

-- A SAS program may have more than one DATA step and more than one PROC step.

-- SAS does not work directly on the original data file; the PROC step uses data that are stored in a SAS (.sas) data set created by the DATA step.

Section III. Getting Data into SAS

Learning Objectives

At the conclusion of this presentation, 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 III. Getting Data into SAS

The SAS DATA Step

- The DATA step starts with the keyword DATA followed by the name that you wish to assign to a SAS data set.
- The DATA step 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.

Section III. Getting Data into SAS

Reading Data into SAS

- 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.

Section III. Getting Data into SAS

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.
- SAS 9.3 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 data file.

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

- Check the SAS log window to see if the record length is sufficient to read all your data.

Section III. Getting Data into SAS

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:

<u>Format</u>	<u>Example</u>
List	<code>INPUT Item \$ Calories Protein Fat Fat_PC;</code>
Column	<code>INPUT Item \$ 1-20 Protein 26-29 Fat 31-34;</code>
Informat	<code>INPUT Item \$20. +5 Protein. @31 Fat.;</code>
Mixed	<code>INPUT Item \$ 1-20 Calories Protein @31 Fat +1 Fat_PC.;</code>

- 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. <<Preferred Format>>
- The Informat format name ends with a period (.) but it is not a requirement.

Section III. Getting Data into SAS

Using a CARDS or DATALINES Statement

- Another option is to type or paste your data set directly into the SAS editor.
- This 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 DATALINES 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 important for the last semicolon to appear on the next line after all your data has been listed. If you forget it, your last observation could be deleted because SAS reads all characters as data until it encounters a semicolon (;).

```
/* 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;
```

Section III. Getting Data into SAS

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.

Section III. Getting Data into SAS

Handling Missing Data

- 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 are represented by 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 want to recode missing values in the 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 underscore. 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;
```

Section III. Getting Data into SAS

The SAS Proc Step

- Each SAS procedure (or PROC) has unique characteristics and elements but many are shared as well. 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.

Section III. Getting Data into SAS

PROC Syntax Commonalities

- There are syntax rules in effect when learning to write SAS procedure codes.
- All PROCs begin 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 key word 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.

Section III. Getting Data into SAS

PROC Statements and Options

- 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.

Section III. Getting Data into SAS

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.

Section III. Getting Data into SAS

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 UNIVARIATE 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.

Section III. Getting Data into SAS

PROC FREQ

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 cross tabulation table for two or more variables you will need to specify the variable names separated by an asterisk (*).

Section III. Getting Data into SAS

Creating a Permanent SAS Data Set

- 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 sessions.
- 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.

```
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;
```

- The libref can change but the program must point to the same library and member name.

Section III. Getting Data into SAS

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.

Section IV. Submitting a Program in SAS


Learning Objectives

At the conclusion of this presentation, 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 IV. Submitting a Program in SAS

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"  button or select the SUBMIT command under the RUN heading.

Section IV. Submitting a Program in SAS

Telling SAS which Dataset to Use

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.

Section IV. Submitting a Program in SAS

Commenting Out SAS Commands

- 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.
- You can submit only a portion of your program by commenting out the program code you do not want to execute.
- The results and advantages are the same as submitting only a highlighted portion but sometimes more efficient if you have large chunks of code.
- There are two ways to comment out SAS Commands:
 - (1) **Single line** -- place an asterisk (*) in front of the line (you already have a semicolon in place as the terminator).
 - (2) **Multiple lines** -- place the symbols /* before the first line and the symbols */ after the last line.
- The 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.

Section IV. Submitting a Program in SAS

Examining the Results

- As noted when a program executes, information is written in the Log and Output Windows.
- Check the Log Window to ensure the program ran as expected without error.
- If no errors have occurred, look at the output in sequential order to gain a better grasp of the results.
- If a hard copy is desired, the results can be printed as long as the output file remains active.
- The log and output files can be saved at any time before exiting the SAS program.
- The log file is given a *.log* extension name and the output file is given a *.lis* (stands for "listing") extension name.

Section IV. Submitting a Program in SAS

Correcting SAS Errors

- It is important to examine your data for errors or inconsistencies in each response set, especially the data you are planning to use in your analysis.
- Even data from reliable sources can have errors.
- SAS procedures can be used to help detect data errors.
For example, PROC FREQ can be used to check categorical variables to make sure only expected values are present. PROC MEAN and PROC UNIVARIATE can be used to make sure the minimum and maximum values of a variable fall within the expected range.
- Within SAS you can also use IF-THEN/ELSE statements to create flags that alert you to inconsistencies or possible coding problems.
- If problems in the data set are identified, you need to document each instance carefully before making any coding change.

Section IV. Submitting a Program in SAS

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. Ambiguous IF/THEN statement(s) that do not produce the desired results
6. Path to the data file location (INFILE statement) is incorrectly specified
7. Using the letter 'o' when you mean the number 0 (zero) or vice versa
8. Forgetting to specify that a variable is character (SAS assumes every variable is numeric unless it sees a \$ sign)
9. Incorrect column specifications on INPUT statement producing embedded spaces in numeric data
10. Missing data not marked with a period on list-style input causing SAS to read the next data value
11. Merging data sets that are not sorted in the right order
12. INPUT statement reads past the end of a line

Section IV. Submitting a Program in SAS

Using the Enhanced Editor to Find Errors

- One big advantage of the PC SAS System is the automatic color coding provided by the Enhanced Editor.
- The color coding alerts you immediately when some of the more common SAS programming errors occur.
- A common mistake occurs when the programmer forgets to close the quotation marks around a 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, SAS will alert you; it will not, however, correct you if you misspell a variable name or other user defined word.

Section IV. Submitting a Program in SAS

Using the Log File to Find Errors

-- Check the SAS log for three types of messages about the run:

1. **Errors (in red)**

Usually the result of a syntax or spelling mistake. The location of the error is easily found because it is underlined, but it is not necessarily the source of the error (this could be earlier in the program).

2. **Warnings (in green)**

The program still executes with warnings but SAS might have done something you did not want it to do. Read any warnings carefully and make sure you know what they are about and that you agree with them.

3. **Notes (in blue)**

Sometimes just a piece of information; other times an indicator of a problem. Read all notes carefully.

Section IV. Submitting a Program in SAS

Correcting SAS Errors Checklist

- ✓ 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.

Section IV. Submitting a Program in SAS

Saving SAS Files

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 line and use indentation to show the different parts of the program.

Section IV. Submitting A Program in SAS

Exporting SAS Files

- 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 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 file using PROC EXPORT statements.

Section IV. Submitting a Program in SAS

SAS Help and Other Resources

- SAS help is 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 also 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 help 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, 4th or 5th Edition). All editions are available online for limited preview at <http://books.google.com>. You can purchase new and/or used copies of each edition 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_PhilSUG.ppt.
- For SAS coding tips/techniques go to <http://www.sconsig.com.sastip.htm>.
- For a LISTSERV based world-wide SAS discussion group that is always open for discussion check out SAS-L at <http://listserv.uga.edu/archives/sas-l.html>.