

Introduction to SAS on Windows

for SAS Versions 8 or 9

October 2009

I.	Introduction	2
	Availability and Cost –	2
	Hardware and Software Requirements –	2
	Documentation	2
II.	The SAS Environment	3
	SAS Commands	4
	The Viewtable Window	4
	Saving Data in the Viewtable	5
III.	SAS Language	6
IV.	Running Programs in the SAS Environment	7
	Enter a Program in the SAS Editor	7
	Submit your Program	7
	The SAS Log and Output	7
	Print or Save the Output or Log	8
	The Results window	8
V.	Reading External Data Files	9
	Text Files	9
	Save your program	11
	Exit the SAS system	11
	Use a Saved SAS program	11
	Excel Files	11
VI.	Procedures	13
	Using the Results window	14
VII.	More Data Step Features	15
	Create or Change Variables	15
	Make a Permanent SAS Dataset from a Temporary Dataset	16
	LIBNAME statement	17
	Put it All Together	17
	Creating subsets	18
VIII.	SAS Dates	19
IX.	SAS/ASSIST	20
X.	Online Documentation and Learning	25
	Documentation	25
	Online Tutor and e-Learning	25

I. Introduction

Availability and Cost –

Any student, staff or faculty at UMASS may purchase a license for SAS Software from OIT, Lederle Bldg. (A119). The license requires an initial fee at the time of purchase, and an annual fee, due each June. Call 545-9730 for pricing and purchasing information. The current release is SAS 9.2. SAS 8.2 is available for XP Home users.

SAS Software is also available in all OIT/PCCO classrooms and labs on campus. You need an OIT account to use these labs. See <http://www.oit.umass.edu/classrooms/index.html> for computer lab locations and hours.

Hardware and Software Requirements –

SAS Release 9.2

Intel or Intel-compatible Pentium 4 class processor (minimum required)
512 MB minimum memory, more for improved performance
2500 MB free hard disk space to install all licensed products except GIS Census Tract maps.
Windows XP Professional, Service Pack 2 or higher, OR Windows XP Professional for x64 systems OR Windows Vista (32-bit or x64 Business, Enterprise or Ultimate.

Internet Explorer 6, SP2 or higher.
File system support for long filenames is required.

SAS Release 8.2

Intel or Intel-compatible Pentium class processor (minimum required)
64 MB minimum memory, more for improved performance
750 MB free hard disk space to install all licensed products and Online Help.
Windows 95 with year 2000 updates provided by Microsoft OR Windows 98 Service Pack 1, or Second Edition, or Windows ME OR Windows NT Version 4.0, Service Pack 4 or higher, with year 2000 updates OR Windows 2000 or XP

Documentation

Complete reference documentation is available online:
for SAS 9.2 <http://support.sas.com/documentation/>
for SAS 8.2 <http://v8doc.sas.com/sashtml/>

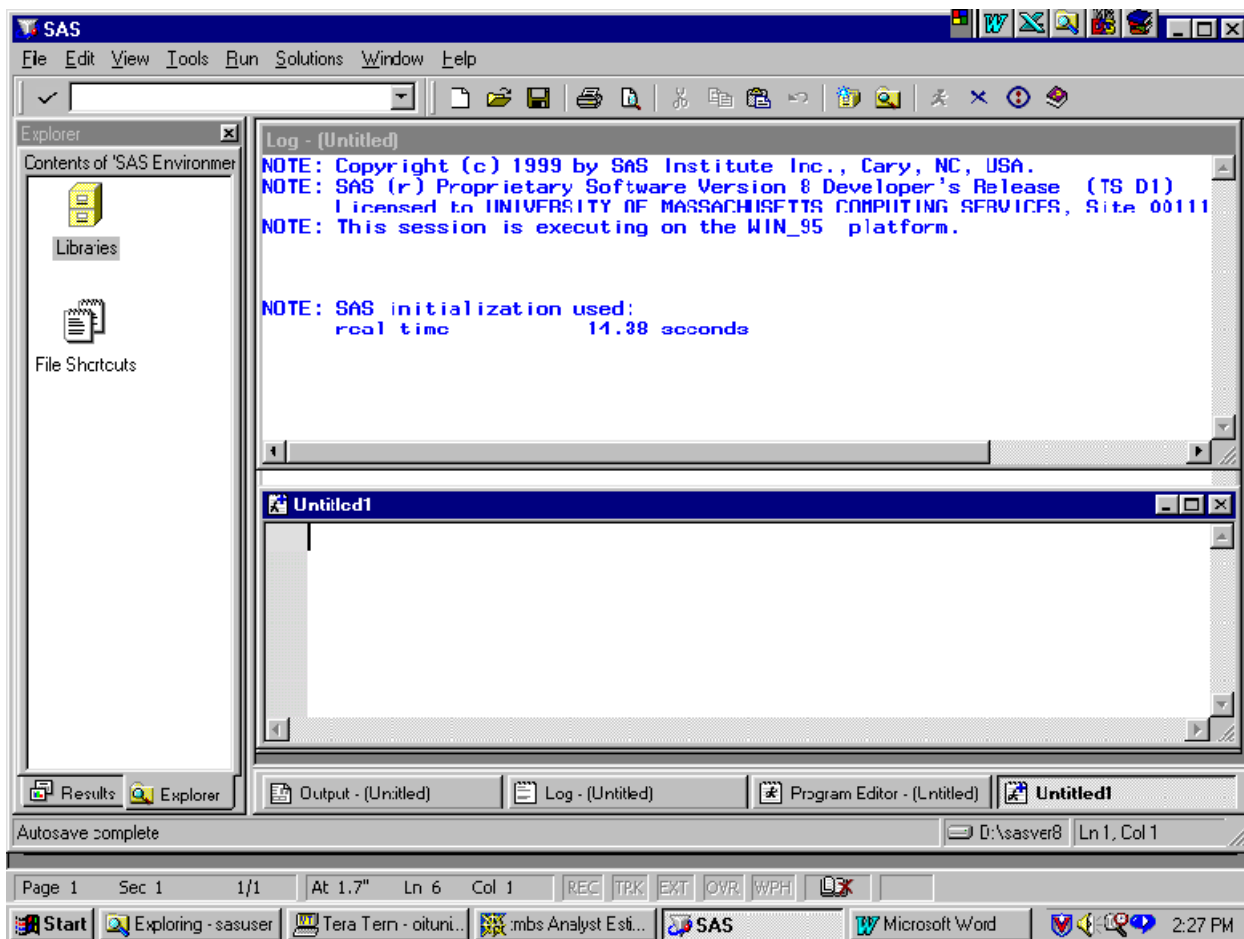
The SAS 8.2 URL is for the searchable version of the documentation, which requires Java. Some browser/Java versions do not work well with this documentation. Hardcopy documentation can be purchased directly from SAS. See:

<http://www.sas.com/pubs>

If you already have manuals from a previous release, they are a good resource in most cases, though they will not contain the most recently added features.

II. The SAS Environment

SAS provides windows for accomplishing all the basic tasks. When you start SAS, you get the five main SAS windows: Explorer, Results, Program Editor, Log, and Output. Your screen will look similar to this:



In the Explorer window, you can view and manage your SAS files and create shortcuts to non-SAS files. Use this window to create new libraries and SAS files; open any SAS file; perform most file management tasks such as moving, copying, and deleting files.

In the Program Editor window, you enter, edit, and submit SAS programs. To open your SAS programs, click in the Program Editor window. From the **FILE** menu select **Open** and select your SAS program.

The Log window displays messages about your SAS session and any SAS programs you submit.

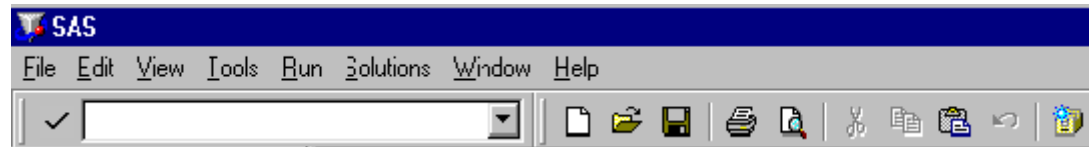
In the Output window, you can browse output from SAS programs you submit. By default, the Output window is positioned behind the Program Editor and Log windows. When you create output, the Output window automatically moves to the front of your display.

The Results window helps you navigate and manage output from SAS programs you submit. You can view, save, and print individual items of output.

Initially, the Results window is positioned behind the Explorer window and is empty until you submit a SAS program that creates output. Then it moves in front Explorer.

SAS Commands

There are SAS commands for performing a variety of tasks. You have up to three ways to issue commands: menus, icons, or the SAS command bar (or command line). This picture shows the location of these three methods of issuing SAS commands:



Menus

Pull-down menus are located at the top of your screen. The choices in the menu are: File, Edit, View, Tools, etc. The menu will change according to which window is active.

SAS command bar

The command bar, directly below the menu, is a where you can type SAS commands. Most of the commands that you can type in the command bar are also accessible through the pull-down menus or icons.

Icons

The tool bar to the right of the command bar has icons to give you quick access to the most commonly used commands.

The Viewtable Window

Besides the five main windows mentioned above, there are many other SAS windows available for various tasks. The Viewtable window is an easy way to create new data sets or browse and edit existing data sets. The Viewtable window displays tables (another name for data sets) in a tabular format. To open the Viewtable window, go to the **TOOLS** menu and select **Table Editor**. An empty Viewtable window will appear

A screenshot of the Viewtable window. The title bar reads "VIEWTABLE(New): (Untitled)". The window contains a table with 9 rows and 10 columns. The columns are labeled A through I, and the rows are labeled 1 through 9. The table is currently empty.

	A	B	C	D	E	F	G	H	I
1									
2									
3									
4									
5									
6									
7									
8									
9									

The table has rows (or observations) labeled with numbers and columns (or variables) labeled with letters. Enter the small dataset shown below. To give the columns more meaningful names click on each column headings (A, B, ...) and replace it with a better variable name (name, height, ...).

SAS will assign your columns as numeric or character based on the first row of data.

	name	height	weight	jog	tennis	F	G	H	
1	Robert	68	150	yes	yes				
2	George	67	180	no	no				
3	Agatha	63	110	no	no				
4	Sandy	60	125	yes	yes				
5	Bill	65	160	yes	no				
6									
7									
8									
9									

Saving Data in the Viewtable

When your data is complete and you are satisfied with the accuracy you need to save it. From the **FILE** menu, select **Save As**. From the Save As dialog box, select the **Work** library and then enter a member name for your data. **Libraries** are directories (folders). **Members** are files.

You always have a choice of at least three libraries: **Sashelp**, **Sasuser** and **Work**. (There may or may not also be **Maps** and/or **Gismaps** libraries.) The only one of these libraries you can safely use is **Work**. The others are used by SAS for its own purposes. **Work** is a **temporary** library or workspace that exists for the duration of your SAS session and is discarded at the end of each session. This means your data set will last for the duration of your session but will be discarded when you quit SAS.

If you wish to create a **permanent** SAS data set, you must define your own library. In the Save As dialog, click the **New Library** icon (it looks like a file drawer with a blue star on top). When the New Library dialog box appears, enter a name for the new library and select its location (path). For the workshop we will use library name: `ref` and path:

`C:\Data\Shared Files\SasWork`. Click **OK**. In the Save As dialog box, under **libraries** select your newly created library, `ref` and enter

Member Name: `exercise`. Click **Save**.

*Note that the **library** is the name of the **directory (folder)** where you store your permanent SAS datasets. It is **NOT** a specific file. The **member** name specifies a particular file.*

You have now saved data into a sas dataset in directory `c:\Data\Shared Files\SasWork`. Its Windows filename is `exercise.sas7bdat`, but within SAS you refer to the dataset using library name and the first part of the filename: `ref.exercise`.

All SAS 8 and 9 datasets on Windows use the extension .sas7bdat, so it is understood, and should not be explicitly stated.

III. SAS Language

With SAS, you use statements to write a series of instructions called a SAS program. The program tells SAS what to do using the SAS language. There are some menu-driven front ends to SAS; SAS/ASSIST and SAS Analyst, for example, make SAS appear like a point-and-click program. However, these front ends still use the SAS language to write programs for you. You will have much more flexibility using SAS if you learn to write your own programs using the SAS language.

There are a few rules for the SAS language. The most important rule is:

Every SAS statement ends with a semi-colon. The arrangement of SAS commands on lines is strictly for human legibility. As far as SAS is concerned, the only thing that separates one command from the next is the semi-colon.

SAS programs consist of two basic building blocks: **DATA** steps and **PROC** steps. A typical SAS program starts with a **DATA** step to create or manipulate a SAS data set and then passes the data to **PROC** steps for processing. However, the **DATA** step is not required. If you have a permanent SAS dataset and do not need to change it in any way prior to analysis, you can go directly to the **PROC** step. The example below uses the SAS dataset (exercise) created previously using the Viewtable window, creates a new variable (htcm), and prints the new dataset with a **PROC** step:

```
DATA work.newexer;  
SET ref.exercise;  
      /* or set 'c:\Data\Shared Files\SasWork\exercise'; */  
htcm=height*2.54;  
PROC print data=work.newexer;  
RUN;
```

The **DATA** statement begins the data step and tells SAS to name the new sas dataset newexer. The new dataset will be in the **WORK** library, which means it is temporary.

The **SET** statement tells SAS use the existing SAS dataset ref.exercise as the starting point for the new dataset. We can refer to the permanent SAS dataset that we are using to create the new one in two ways: using the library name (ref) that we created earlier to refer to its location, or specifying the complete location path where the dataset is stored. Without one of these specification, SAS will not be able to find the permanent SAS dataset.

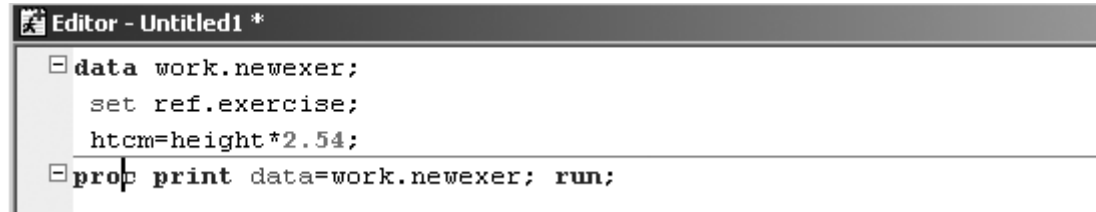
The next statement creates a new variable called htcm for each person which is the original height multiplied by 2.54. The **PROC print** ends the **DATA** step and says to process the new dataset (work.newexer) by listing it in the output window.

Each **DATA** step or **PROC** is a section of SAS program, which ends at the start of the next **DATA** step or **PROC**. Another way to end a **DATA** step or **PROC** is with a **RUN** statement. Every **DATA** step or **PROC** must be ended either by another **DATA** step or **PROC**, or by a **RUN** statement. Therefore, the final statement in a SAS program is always a **RUN** statement.

IV. Running Programs in the SAS Environment

Enter a Program in the SAS Editor

SAS programs are developed in the Editor. Click on the Editor window and then type the program shown below into the window.

A screenshot of the SAS Editor window titled "Editor - Untitled1 *". The window contains the following SAS code:

```
data work.newexer;  
  set ref.exercise;  
  htcm=height*2.54;  
proc print data=work.newexer; run;
```

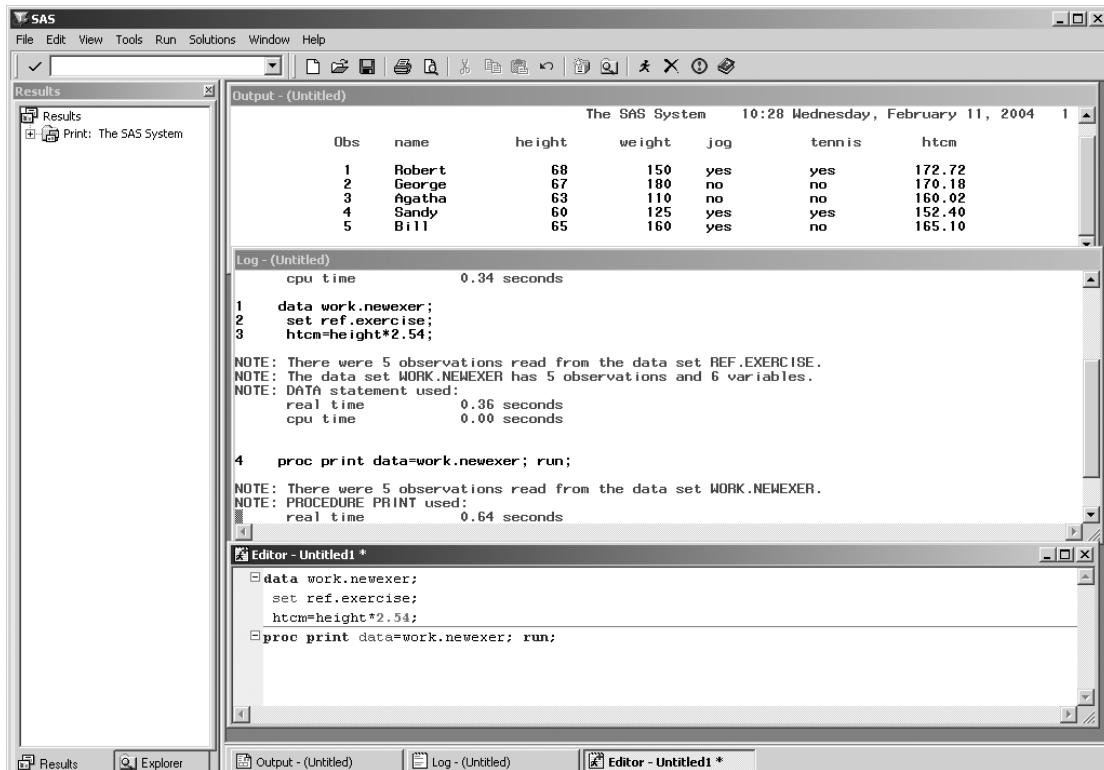
Submit your Program

When your program is ready, you execute it using the **SUBMIT** command. You can use any of three ways to **SUBMIT** a program:

- Click the **submit** icon on the tool bar (the runner figure)
- with the Editor window active, type **SUBMIT** in the command line
- with the Editor window active, and select **RUN** → **Submit** from the menu.

The SAS Log and Output

After you submit your program, the results of your program go into the Log and Output windows. If your program produced any output, the Output window comes to the foreground and you will get new entries in your Results window. The Results window is like a table of contents for your SAS output. The following figure shows what your screen might look like after you submit the above program, with the window sizes adjusted so the output, log and editor are all visible.



Print or Save the Output or Log

If you want to print or save the entire contents of the Output or Log windows, first click in the desired window to activate it, then from the **FILE** menu, select **Print** or **Save As**.

The Results window

When you have a lot of output, the Results window can be very helpful. The Results window is like a table of contents for your output. It lists each procedure that produces output. When you open, or expand, the procedure in the Results tree, you can see each part of the procedure output. Using the Results window you can go directly to any section of a lengthy output, and print or save sections of output, rather than the entire output. We will work more with the Results window when we have more output.

V. Reading External Data Files

SAS procedures can only work with SAS datasets. If you get data in a non-SAS file, your first task is to make it into a SAS dataset.

Text Files

If your data are in raw data files (also referred to as text, or ASCII files), you need to write a **DATA** step program to read the data and create a SAS data set. We will illustrate this by reading the ASCII data file `minidat.dat`. Here is the description of the `minidat.dat` data file:

Students in an introductory statistics course participated in a simple experiment. The students took their own pulse rate and then they were asked to flip a coin. If the coin came up heads, then they were asked to run in place for one minute. Then everyone took their own pulse again. The following eight variables were recorded on each student:

	<u>Variable</u>	<u>Description</u>	<u>Missing</u>
1	pulse1	first pulse rate	0
2	pulse2	second pulse rate	0
3	ran	1=ran in place, 2=did not run	
4	smoke	1=smoke, 2=does not smoke	9
5	sex	1=male, 2=female	
6	height	height in inches	
7	weight	weight in pounds	0
8	activity	level of activity, 1=slight ,2=moderate,3=a lot	9

The file can download it from <http://www.umass.edu/statdata/statdata>. Click on Index to Local Datasets and find the `minidat` data file. We are using the **text** version.

Here is a SAS program to read the data and make a temporary SAS dataset (substitute the location of your `minidat.dat` file in the `infile` command):

```
Editor - Untitled1 *
data work.minidat;
infile
  'C:\Documents and Settings\ACGuest.OIT\Desktop\SasWork\minidat.dat' missover;
input pulse1 pulse2 ran smoke sex height weight activity;
run;
proc print data=work.minidat (obs=10); run;
```

The **DATA** statement begins the data step and tells SAS to name the SAS dataset `minidat`. The **WORK** prefix makes the dataset temporary.

The **INFILE** statement tell SAS where to find the ASCII data file called `minidat.dat`. Unlike with SAS datasets, you must specify the entire filename, including the file extension. The `missover` option prevents SAS from reading data from the next line when a line is incomplete.

The **INPUT** statement assigns a variable name to each piece of information in the `minidat.dat` file type. A `$` following a variable name is needed to indicate any variables that

contains character data. The minidat data file has only numeric data, so we do not need to use the \$. The simple **INPUT** statement shown here will work if the data file has:

- each value on a data line separated from the next by at least one blank space
- character data values no more than eight characters long
- missing data values represented by a period or other character rather than left blank

The **INPUT** statement has many options that can be used to read data files that do not meet all of the above criteria..

A **RUN** statement marks the **DATA** step.

PROC PRINT tells SAS to list the first 10 observations in the Output window.

Submit the above program. Your log and output window should look like this:

The screenshot shows two windows from the SAS interface. The top window is titled "Output - (Untitled)" and displays a table of data for 10 observations. The bottom window is titled "Log - (Untitled)" and shows the SAS log messages, including the number of records read and the execution of the PROC PRINT statement.

Obs	pulse1	pulse2	ran	smoke	sex	height	weight
1	64	88	1	2	2	66.0	140
2	58	70	1	2	1	72.0	145
3	62	0	1	1	1	73.5	160
4	66	78	1	1	1	73.0	190
5	0	80	1	2	1	69.0	155
6	74	84	1	2	1	73.0	165
7	84	84	1	9	1	72.0	0
8	68	72	1	2	1	74.0	190
9	62	75	1	2	1	72.0	195
10	76	118	1	2	1	71.0	138

Log - (Untitled)
RECFM=V,LRECL=256
NOTE: 91 records were read from the infile 'c:\windows\desktop\saswork\minidat'.
The minimum record length was 32.
The maximum record length was 32.
NOTE: The data set WORK.MINIDAT has 91 observations and 8 variables.
NOTE: DATA statement used:
real time 0.05 seconds

10 proc print data=minidat(obs=10);run;
NOTE: There were 10 observations read from the dataset WORK.MINIDAT.
NOTE: PROCEDURE PRINT

The Log window indicates that 91 records were read from the minidat.dat file and a SAS dataset minidat (stored temporarily in the WORK library) was created with 91 observations and 8 variables. The print procedure listed the data for 10 observations.

Save your program

To save a program for future use, from the Editor window's **FILE** menu, choose **Save As** and enter a name for your program. File type should be `.sas`. Choose an appropriate directory. In the workshop, use `C:\Data\Shared Files\SasWork` with the name `class.sas`.

Exit the SAS system

Now that your program is saved, you can exit SAS. Although the SAS dataset is temporary and will be discarded when you end SAS, you can use the saved program to re-create it any time. To exit from SAS, from the **FILE** menu click on **Exit**.

Use a Saved SAS program

Re-start SAS and open the saved program in the Editor window. From the **FILE** menu, select **Open** (in SAS 8) or **Open Program** (in SAS 9). When the Open dialog box appears, browse to the appropriate directory and select the saved sas program file. Click **Open**. **Submit** the program to get back to where you were before you exited SAS.

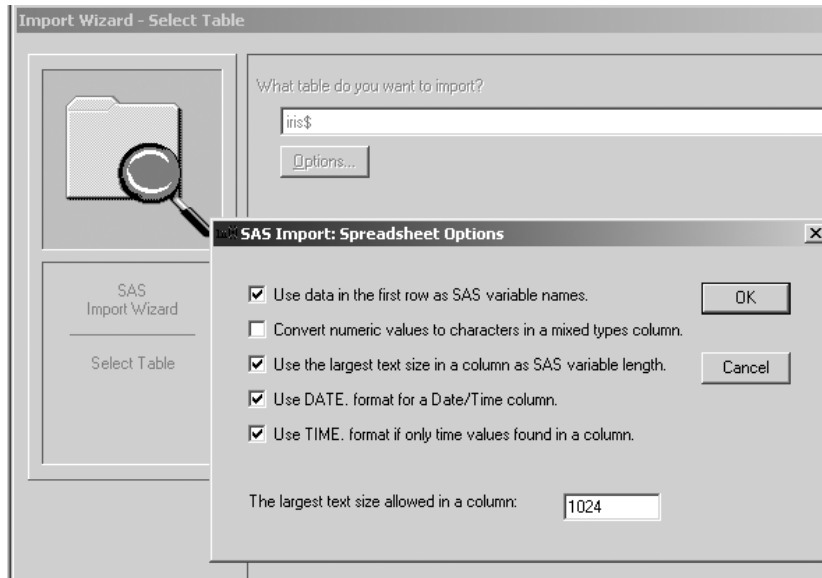
Excel Files

Data in Excel worksheets can be imported easily provided the worksheet is suitably organized. The Excel worksheet should have rows corresponding to observations (subjects) and columns corresponding to variables. The first row may contain variable names. Get rid of anything extraneous (for example, totals) in the sheet you want to import into SAS.

To import a worksheet, first make sure it is not open in Excel.

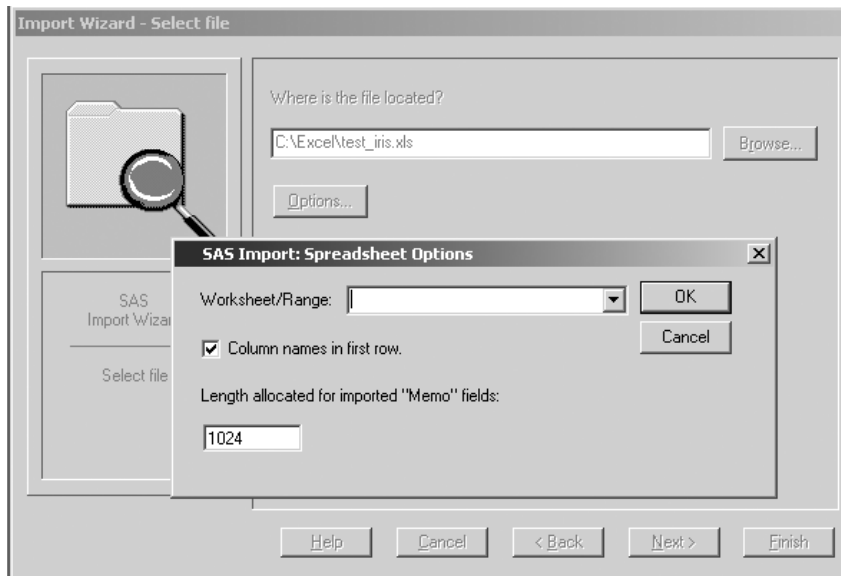
SAS 9.1.3

- From SAS's **File** menu, select **Import Data**.
- Follow the Import Wizard dialog to select the version of Excel, and the file to import.
- On the **Select Table** screen, use the "What table do you want to import" dropdown box to select the correct worksheet.
- Click **Options** (see illustration below)
- If you have variable names in the first row, make sure "Use data in first row as variable names" is checked in the Options selection. Otherwise remove the check mark.
- On the next screen, select the **Library** where the data is to be stored (use **WORK** if you have not defined your own SAS library) and enter a name for the new SAS dataset.
- Click **Finish**.



SAS 8.2

- From SAS's **File** menu, select **Import Data**.
- Follow the Import Wizard dialog to select the version of Excel, and the file to import.
- If the Excel file has multiple worksheets, use the **Worksheet/Range** dropdown box to select the correct worksheet. If there is only one worksheet you can leave this box blank.
- On the **Select File** screen, click **Options** (see illustration below)
- If you have variable names in the first row, make sure "Column names in first row" is checked in the Options dialog. Otherwise remove the check mark.
- On the next screen, select the **Library** where the data is to be stored (use **WORK** if you have not defined your own SAS library) and enter a name for the new SAS dataset.
- Click **Finish**.



VI. Procedures

Using the temporary SAS dataset `work.minidat`, we would like to get frequencies for the categorical variables: `ran`, `sex`, `smoke` and `activity` and descriptive statistics for the continuous variables: `pulse1`, `pulse2`, `height`, and `weight`. We will use PROC FREQ and PROC MEANS to do this:

```

Editor - Untitled1 *
proc freq data=work.minidat;
  tables ran smoke sex activity;
proc means data=work.minidat;
  var pulse1 pulse2 height weight;
run;
  
```

PROC FREQ is used to obtain counts of each value for the variables specified on the **TABLE** subcommand. PROC MEANS is used to get descriptive statistics for the variables listed on the **VAR** subcommand. In general, each procedure has its own subcommands, though there are some subcommands that can be used on any procedure. The **DATA=** option is used on all procedures to indicate which SAS dataset to use for the analysis.

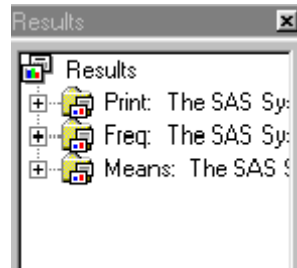
Here is the output from PROC FREQ and PROC MEANS:

Output - (Untitled)					
ran	Frequency	Percent	Frequency	Percent	
1	36	39.56	36	39.56	
2	55	60.44	91	100.00	
smoke	Frequency	Percent	Cumulative Frequency	Cumulative Percent	
1	28	30.77	28	30.77	
2	62	68.13	90	98.90	
9	1	1.10	91	100.00	
sex	Frequency	Percent	Cumulative Frequency	Cumulative Percent	
1	55	60.44	55	60.44	
2	36	39.56	91	100.00	
activity	Frequency	Percent	Cumulative Frequency	Cumulative Percent	
1	9	9.89	9	9.89	
2	60	65.93	69	75.82	
3	21	23.00	90	98.90	
9	1	1.10	91	100.00	

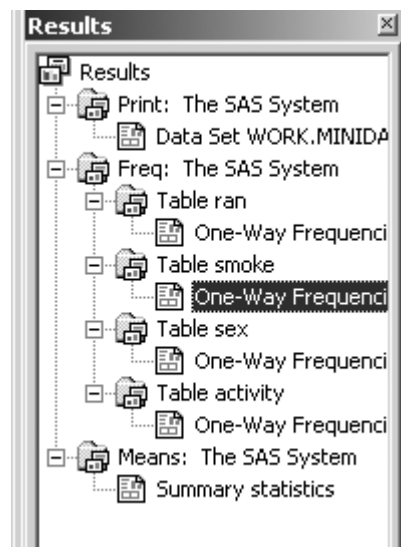
The MEANS Procedure					
Variable	N	Mean	Std Dev	Minimum	Maximum
pulse1	91	72.6489516	13.1945895	0	100.0000000
pulse2	91	79.8021978	19.0176642	0	140.0000000
height	91	68.7032967	3.6800725	61.0000000	75.0000000
weight	91	142.2417582	27.3391622	0	195.0000000

Using the Results window

Now that we have run more procedures, let's take another look at the Results windows to explore its uses. The following output shows what your Results window might look like after running Proc Print, Proc Freq, and Proc Means.



There is one entry in the window for each procedure. Note the plus sign (+) to the left of each procedure. Expand the procedures by clicking on all the plus signs. The Results window should now look like this:



Using the Results window you can go directly to the output of any procedure. For example, double-click on the "One-Way Frequency" entry under `Table smoke`. The Output window jumps to the results of the Frequency table for variable `Smoke`.

You can also print or save just the parts of the output you want. In the Results window, right-click the item of interest. Select **Print** or **Save As** from the pop-up menu.

VII. More Data Step Features

Often your data is not exactly in the form you need to do your analysis. You may need to create new variables by combining existing variables, or you may want to analyze only part of your data, or the data has been assigned missing value codes that are not what SAS expects for missing values. These kinds of manipulation of your data can be done in the **DATA** step.

Create or Change Variables

In the description of the `minidat.dat` data, we see that missing values for *smoke* and *activity* are coded 9, and for *pulse1*, *pulse2* and *weight* they are 0. As you can see in the above output, SAS uses these codes as if they were legitimate data values. We need to tell it to exclude these codes from all analyses. In addition, we'd like to look at the difference between *pulse2* and *pulse1* as a way of assessing the effect of running.

We use a **DATA** step to assign the missing values and create the new variable. We assign the missing values first so the newly created variable will be missing when either of the variables used in computing it has a missing value. We then run **PROC FREQ** and **PROC MEANS** again to check that we have assigned missing values and created the new variable correctly.

```
Program Editor - minidat2.sas
data minifix; set minidat;
if smoke=9 then smoke=.;
if activity=9 then activity=.;
if pulse1=0 then pulse1=.;
if pulse2=0 then pulse2=.;
if weight=0 then weight=.;
pulsdif=pulse2-pulse1;
proc freq data=minifix;
  tables smoke activity;
proc means data=minifix;
  var pulse1 pulse2 weight pulsdif;
run;
```

The **DATA** statement begins the data step and tells SAS to call the new SAS dataset `minifix`. The **SET** statement says to use the temporary SAS dataset `minidat` as the starting point for the new SAS dataset `minifix`.

Note: If we do not explicitly name a library, **WORK** is assumed. Thus, dataset `minidat` is the same as `work.minidat`, and `minifix` is `work.minifix`.

The five **IF** statements assign a period (missing value code) to the corresponding variable whenever its value is 9 (variables: *smoke* and *activity*) or 0 (variables: *pulse1*, *pulse2* and *weight*). The next statement creates new variable *pulsdif* as the difference between *pulse2* and *pulse1*. We then run the frequency and means procedures again to check our results.

Here is the output on the revised data:

Output - (Untitled) The SAS System 11:34 Thursday, S

The FREQ Procedure

smoke	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	28	31.11	28	31.11
2	62	68.89	90	100.00

Frequency Missing = 1

activity	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	9	10.00	9	10.00
2	60	66.67	69	76.67
3	21	23.33	90	100.00

Frequency Missing = 1

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
pulse1	90	73.4555556	10.7746697	54.0000000	100.0000000
pulse2	90	80.6888889	17.1284929	50.0000000	140.0000000
weight	90	143.8222222	22.9339931	95.0000000	195.0000000
pulsdif	89	7.1123596	13.7885313	-14.0000000	48.0000000

Make a Permanent SAS Dataset from a Temporary Dataset

Now that we have made the corrections to the data, we'd like to save the corrected version as a permanent dataset called `minidat`. In the SAS Explorer Window, double-click the `WORK` library, then double-click the `minifix` dataset. This will open `work.minifix` in the Table Editor. From the Table Editor's **FILE** menu, choose **Save As**, select the library you wish to use (*) and enter the Member Name `minidat`.

(*) If there are no Libraries listed other than the default libraries (`SASHELP`, `SASUSER` and `WORK`) then you need to define a library for your permanent SAS datasets. Click on the **New Library** icon. When the New Library dialog box appears, enter a name for the new library (**ref**) and select its location (`C:\Data\Shared Files\SasWork`). Click **OK**. In the Save As dialog box you can now select **ref** under libraries, and enter **Member Name: minidat**.

This saves the SAS permanent dataset as file `minidat.sas7bdat` in directory `c:\Data\Shared Files\SasWork`.

Note: When you create a new library you do not have to call it **ref**. You may call it anything you wish (up to 8 characters, no blanks or special characters). This name is just a shortcut established for this session to identify the location where the SAS dataset will be stored.

LIBNAME statement

You do not have to use the New Library icon to define and select libraries other than WORK. Libraries can be defined using the **LIBNAME** statement in a SAS program. This is particularly convenient for saving lengthy programs that are to be run without user intervention.

In the SAS program below, the **LIBNAME** statement defines a library called **IN** that points to the directory C:\Data\Shared Files\SasWork. The SET statement copies the temporary dataset *work.minifix* into the permanent dataset, *in.minidat* which will be stored in the location specified on the libname command:

```
libname in 'c:\Data\Shared Files\SasWork';
data in.minidat;
  set work.minifix;
run;
```

Put it All Together

In the examples we used many steps to read the text data file, make some changes to the data, and finally save it as a permanent SAS dataset. We did this in order to gradually add to your repertoire of SAS statements. Using the statements you now know, it is possible to accomplish the entire task at once.

Here is the SAS code to define a SAS library, read the *minidat.dat* text file, make the corrections to the missing values, compute the new variable, and save the resulting *minidat* data as a permanent SAS dataset:

```
libname saslib 'C:\Data\Shared Files\SasWork\';
data saslib.minidat;
infile
'C:\Documents and Settings\ACGuest.OIT\Desktop\SasWork\minidat.dat'
missover;
input pulse1 pulse2 ran smoke sex height weight activity;
if (pulse1 eq 0) then pulse1=.;
if (pulse2 eq 0) then pulse2=.;
if (smoke eq 9) then smoke=.;
if (weight eq 0) then weight=.;
if (activity eq 9) then activity=.;
pulsedif=pulse2-pulse1;
run;
```

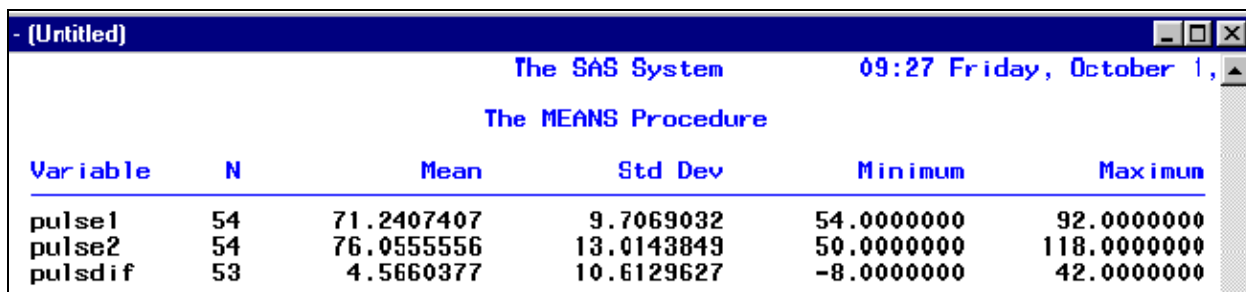
Creating subsets

Often when doing an analysis, we wish to analyze only a portion the data. For example, we wish to do an analysis using only the males in the `minidat` SAS dataset and then repeat the analysis using the females. Again we use a **DATA** step to create these two subsets. Below is the program to create these two subsets and run the means procedure on the subsets.

```
Editor - Untitled1 *
data male female;
  set in.minidat;
  if sex=1 then output male;
  if sex=2 then output female;
Proc means data=female;
  var pulse1 pulse2 pulsdif;
Proc means data=male;
  var pulse1 pulse2 pulsdif;
run;
```

Since we are creating two SAS datasets, the **DATA** statement has two names. We use the **IF** statements to specify which dataset each observation will go into. Finally, we run the means procedure on each newly created subset. Here are the result of this analysis:

Proc means for the male SAS dataset:

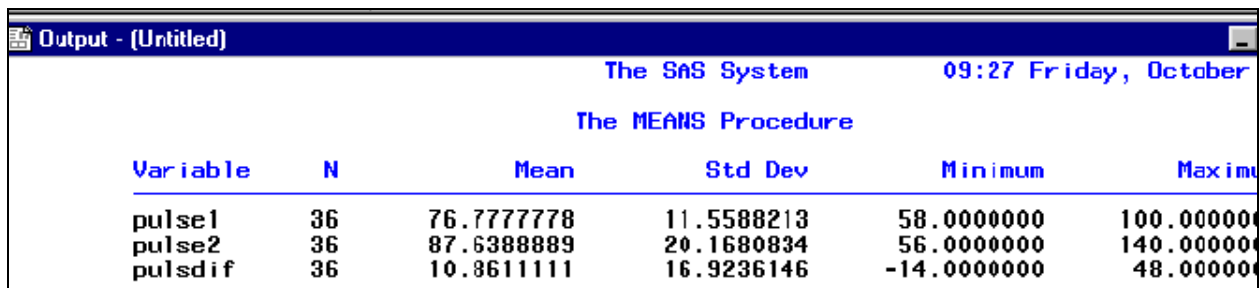


The SAS System 09:27 Friday, October 1, 2004

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
pulse1	54	71.2407407	9.7069032	54.0000000	92.0000000
pulse2	54	76.0555556	13.0143849	50.0000000	118.0000000
pulsdif	53	4.5660377	10.6129627	-8.0000000	42.0000000

Proc means for the female SAS dataset



The SAS System 09:27 Friday, October 1, 2004

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
pulse1	36	76.7777778	11.5588213	58.0000000	100.0000000
pulse2	36	87.6388889	20.1680834	56.0000000	140.0000000
pulsdif	36	10.8611111	16.9236146	-14.0000000	48.0000000

VIII. SAS Dates

Dates can be tricky to work with. Some months have 30 days, some 31, some 28 and don't forget leap year. In SAS, a date is stored as the number of days since January 1, 1960. This makes it easy to calculate elapsed time. The table below lists four dates and their values as SAS dates:

Date	SAS date value
January 1, 1959	-365
January 1, 1960	0
January 1, 1961	366
January 1, 2001	14976

SAS has special tools for working with dates: **informats** for reading dates, **functions** for manipulating dates, and **formats** for printing dates. Refer to the SAS Language manual or SAS online help for a list of date **informats**, **formats** and **functions**.

Informats

To read variables that are dates, you use formatted style input with a date **informat**. The **INPUT** statement below tells SAS to read a variable named *BDAY* using the **MMDDYY10.** informat:

```
INPUT bday mmddyy10.;
```

SAS has a variety of date **informats** for reading dates in different forms. All of these **informats** convert a date to the number of days since January 1, 1960. Refer to the documentation for a complete list of **informats**.

Dates in SAS expressions

Once a variable has been read with a SAS date **informat**, it can be used in arithmetic expressions, just like any other numeric variable. For example, if a library book is due in three weeks, you could find the due date by adding 21 days to the date it was checked out:

```
datedue = datechek + 21;
```

Functions

SAS date functions perform a number of handy operations. For example, the **TODAY** function returns a SAS date value equal to today's date. If **bday** is a SAS Date variable whose value is a person's date of birth, then the statement below can be used to compute that person's age in years:

```
Age= (TODAY () - bday) /365.25;
```

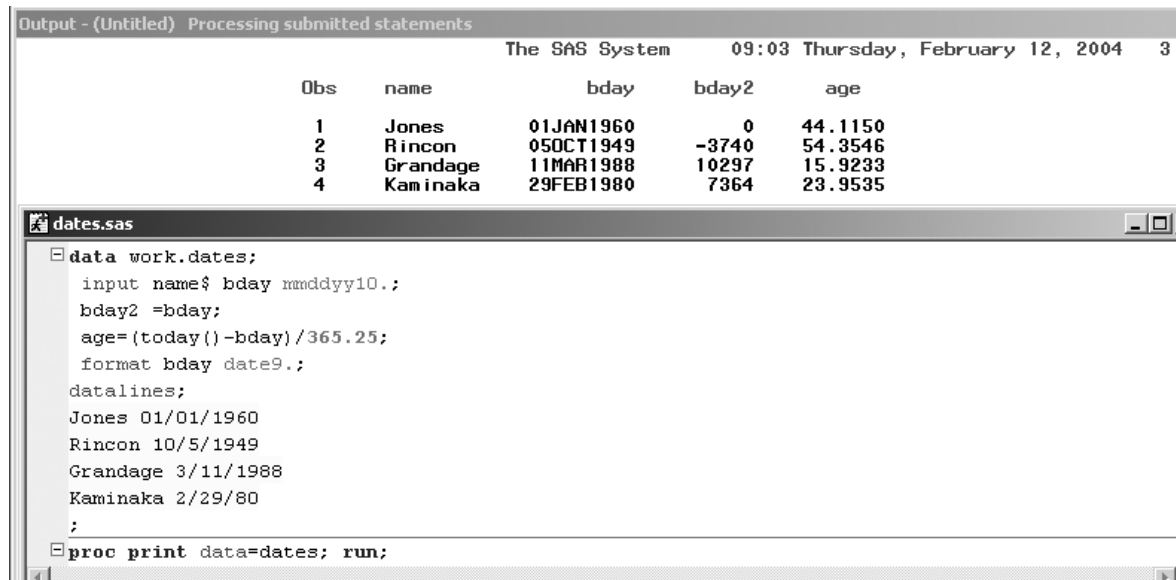
Formats

If you print a SAS date value, SAS prints the actual SAS value (the number of days since January 1, 1960). Since this is not very meaningful to most people, SAS has a variety of **formats** for printing dates in different forms. Refer to SAS documentation for a complete list of **formats**. Normally whenever you use a date **informat** to read data, you will want to assign a date **format** for printing it in a meaningful way. The **FORMAT** statement below tells SAS to print the variable *BDAY* using the **DATE9.** format:

```
FORMAT bday date9.;
```

An example using SAS Dates

Here we read a small data file containing last names and dates of birth. We compute each subject's age and display the results.



The screenshot shows the SAS output window and the code editor. The output window displays a table with the following data:

Obs	name	bday	bday2	age
1	Jones	01JAN1960	0	44.1150
2	Rincon	05OCT1949	-3740	54.3546
3	Grandage	11MAR1988	10297	15.9233
4	Kaminaka	29FEB1980	7364	23.9535

The code editor shows the following SAS code:

```
data work.dates;
  input name$ bday mmddyy10.;
  bday2 =bday;
  age=(today()-bday)/365.25;
  format bday date9.;
datalines;
Jones 01/01/1960
Rincon 10/5/1949
Grandage 3/11/1988
Kaminaka 2/29/80
;
proc print data=dates; run;
```

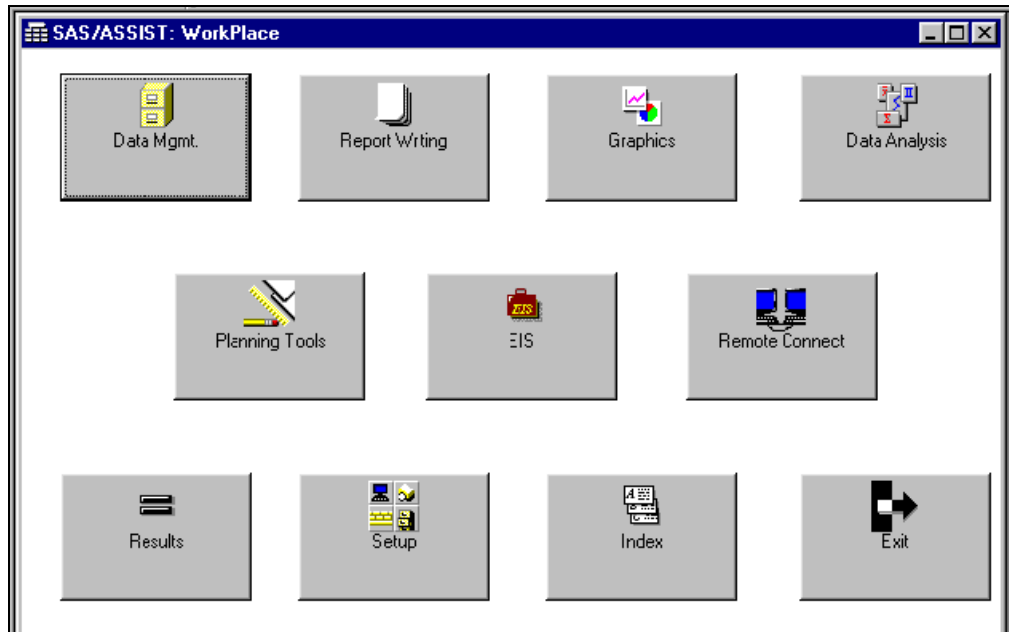
We use the informat `mmddyy10.` to read dates of birth. This informat requires data in the form **mmddyy**, where mm, dd, and yy are integers representing the month, day, and year. It permits up to 10 characters for the date representation. This allows enough room for a separator character between the month, day and year portions of the date, and to enter the year as a 4 digit value.

We then calculate age by using the `TODAY()` function to subtract each person's date of birth from the current date. Finally, we divide by 365.25 to get their age in years. We assign the **date9.** format to the date of birth variable so it will print in a meaningful way. For comparison purposes, we also make a copy of date of birth, `bday2`, without a format. We then print the data.

In the output above you can see the effect of the `date9.` format on `bday`, and no format on `bday2`.

IX. SAS/ASSIST

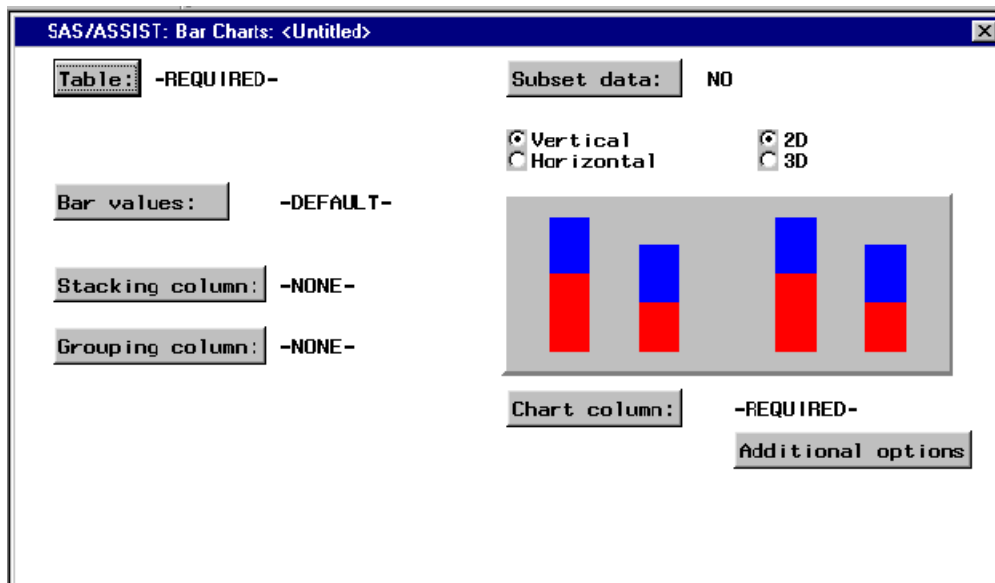
SAS/ASSIST is a menu-driven, task-oriented interface to the SAS System. To start SAS/ASSIST, go to the Solutions menu, and choose Assist. You may have to go through some initial setup screens(just click Continue) to get to the SAS/ASSIST: Start Menu:



Select a task by clicking on it with the mouse. The task menus within SAS/ASSIST are similar. Therefore, once you master one task, other tasks are easy to complete.

Let's make a bar chart using the `minidat` SAS data set. Move the pointer to **Graphics**, click and select **Bar Charts**.

This is the bar charts dialog:

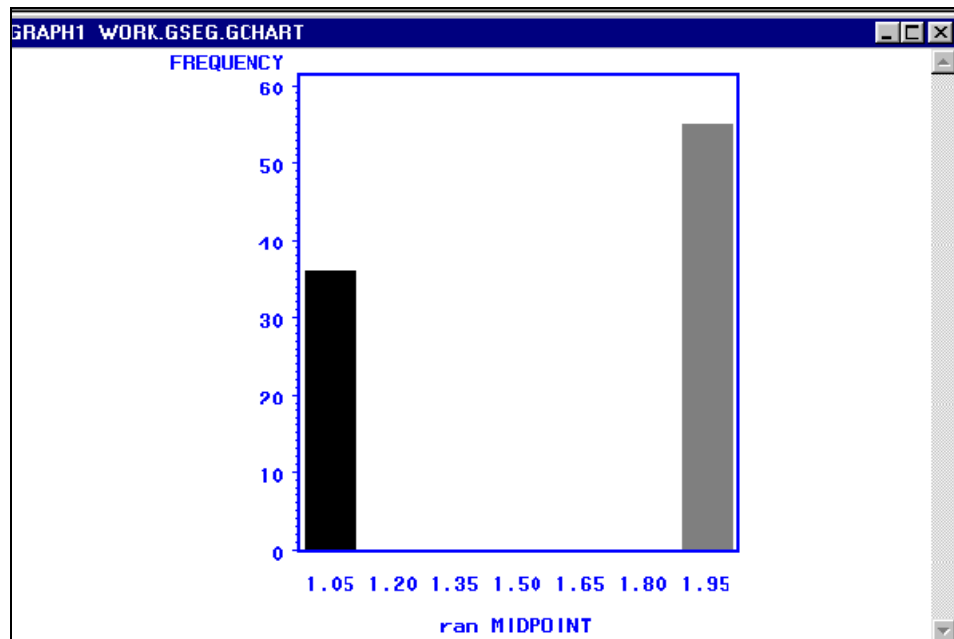


We will make a bar chart of the number of people who ran or did not run. There are two required fields: **Table** and **Chart column**. The **Table** field is where we specify which SAS dataset to use.

Click on **Table**. In the dialog box select **ref** for libraries and `minidat` for tables. Click OK.

Next click on **Chart column**. Select variable **ran**.

Click the **SUBMIT** icon, or select Run→Submit from the menu:



Although this chart shows the correct bars for frequency of runners and non runners, we would like the x axis to display only the two discrete values for ran: **1** and **2**.

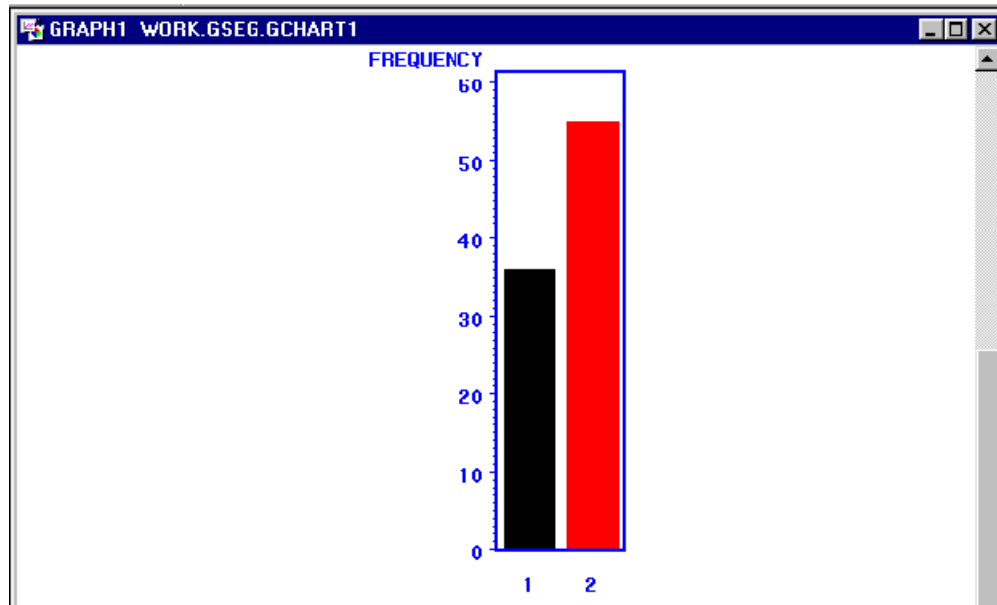
Return to the SAS/ASSIST dialog box by selecting from the **WINDOWS** pull-down menu **SAS/ASSIST:Bar Charts**.

Click **Additional options** and select **number of bars**.

From this dialog box choose: **Use each discrete chart variable value**.

Now click **Goback**, and **submit**.

Here is the revised chart (You may need to scroll to the end of the graphics output window to get to the latest chart):

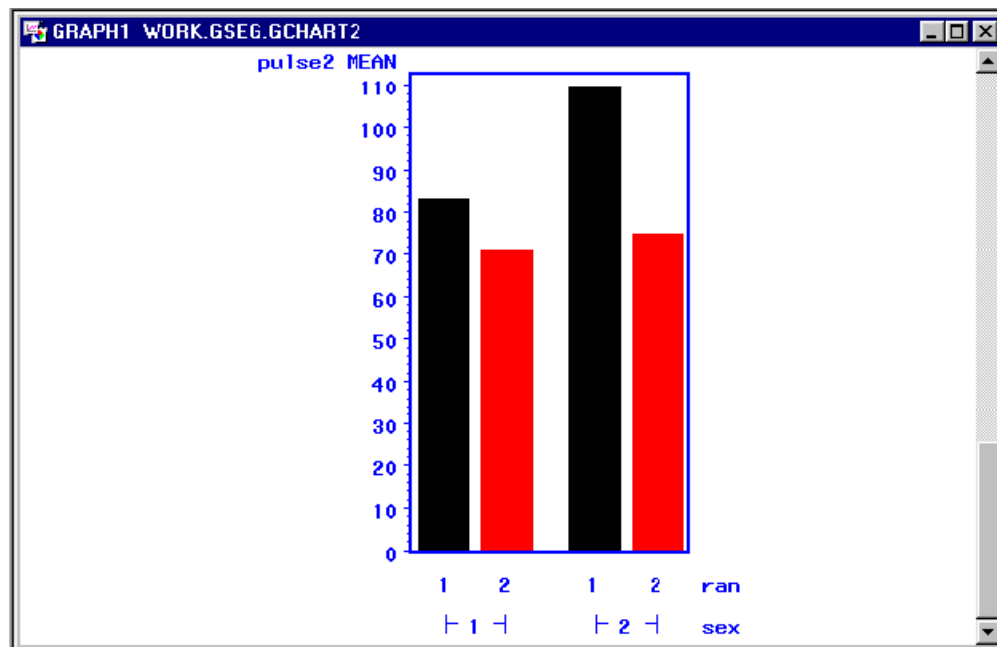


This shows the number of runners and non-runners. But what we really want to see is the average value of *pulse2* for those who ran in place compared to those who did not run, subdivided according to *sex*.

Return to the **Bar Charts** dialog box and click on **Bar value**. From Bar value dialog box select **Mean** and click on **Analysis column**. From the Analysis column dialog box, select variable **pulse2**. Click on **OK**.

In the Bar Charts dialog box click on **Grouping column**. From the Grouping column dialog box select variable **sex**.

Submit to get following graph:



SAS/ASSIST builds and documents SAS programs for each task you perform. You can view and modify these SAS programs, save them or resubmit them like any SAS program.

Go to the **Editor** window. If you can't see it, you can select it from the Windows menu. From the Editor's Run menu, select **Recall last Submit**. (Depending on what you've done, you may need to select Recall last Submit more than once. Each recall brings back the commands from the previous task in the stack.)

```

/*-----*
| Summary:
|   Creating a grouped and stacked bar chart using the table
|   REF.MINIDAT and charting the columns
|   ran along the vertical axis.
| Generated: 26OCT1999 12:50:58
|-----*/
/*-----*
| The GOPTIONS statement allows you to have more control over the
| final appearance of your output such as fonts, colors, text
| height and so on. The output device and destination is also
| specified in the options statement.
|-----*/
goptions reset=(axis, legend, pattern, symbol, title, footnote) norotate
        hpos=0 vpos=0 htext= ftext= ctext= target= gaccess= gsfmode= ;
goptions device=WIN ctext=blue
        graphrc interpol=join;

/*-----*
| PATTERN statements allow you to define colors and patterns in
| the chart, map or plot that you are creating. SAS/GRAPH uses
| any pattern statements that you specify. If more are needed,
| default PATTERN statements are used.
|-----*/
pattern1 value=SOLID;

/*-----*
| AXIS statements allow you to supply information on how your
| vertical and horizontal axes will appear on the graph.
|-----*/
axis1
  color=blue
  width=2.0 ;
axis2
  color=blue
  width=2.0;
axis3
  color=blue
  width=2.0
  ;
/*-----*
| This section produces the actual bar chart and contains the
| options that directly relate to the data and the axis area.
|-----*/
proc gchart data=REF.MINIDAT;
  vbar ran /

  maxis=axis1
  raxis=axis2
  frame
  group=sex
  type=MEAN
  summar=pulse?
  DISCRETE
  patternid=midpoint
  ;
run; quit;

```

X. Online Documentation and Learning

Documentation

You can look up SAS syntax for any command using the online documentation.

SAS 9.2

Under the Help menu, select SAS Help and Documentation. This is the full SAS System documentation.

SAS 8.2

Under the Help menu, use either SAS System Help or Books and Training→SAS Online Doc to look up the syntax for any command. SAS System Help is briefer, with little additional information. SAS Online Doc is a browser based link to the same documentation as you find at <http://v8doc.sas.com/sashtml/>

Online Tutor and e-Learning

SAS 9.2

The University of Massachusetts SAS site license includes 4 online self-paced e-Learning training products. In order to use these, you will be required to register at the SAS website and activate the product(s) you wish to use. Instructions for accessing e-Learning are available at OIT Administrative Desk, room A119 LGRC.

SAS 8.2

SAS Online Tutor is a series of 5 modules with 5 to 10 self-paced lessons per module, designed to introduce basic SAS programming to new users. Most lessons are expected to take 1 to 1.5 hours to complete – a total of 5 to 15 hours per module.

Under the Help menu, go to Books and Training→SAS Online Tutor.