

# Introduction to SAS

Fred Ullrich  
Department of Health Management & Policy  
College of Public Health

*Find a seat and log into your computer using your HawkID and password.  
Create a personal folder (e.g. on your "H:" drive) to store class stuff.*



...”a software suite developed by SAS Institute for advanced analytics, multivariate analyses, business intelligence, data management, and predictive analytics.”





## Overview

- Day 1
  - Introduction to SAS
  - Making Changes to Data in SAS
- Day 2
  - Introduction to SAS Procedures
  - ODS Graphics Designer
  - Introduction to SAS EG

## Uses

- Access and manage data across multiple sources
- Generate reports and perform analyses

# Interfaces

- SAS Windowing Environment  (SAS)
  - Provides a full programming interface
- SAS Enterprise Guide  (SAS EG)
  - Provides a point-and-click interface with menus and wizards to create code

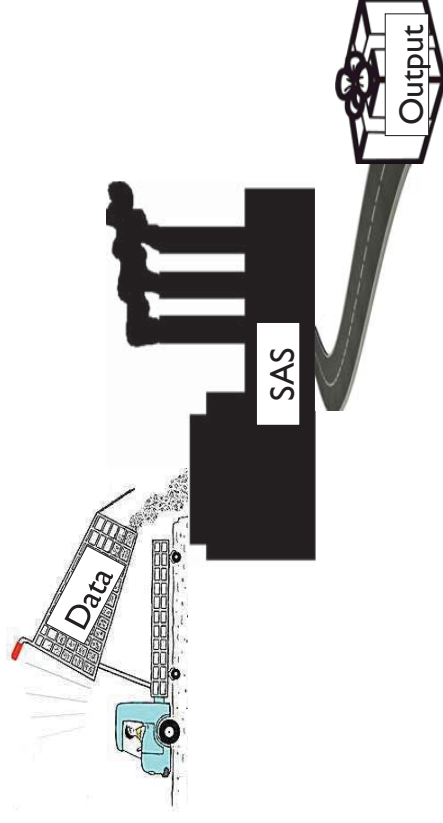
# Access at UI

- PC Installation
  - Requires purchase of SAS license
- Department licenses:
  - College of Business
  - College of Dentistry
  - College of Education
  - NADs (College of Engineering)
  - College of Liberal Arts and Sciences
  - College of Nursing
  - College of Pharmacy
  - College of Public Health
  - Iowa Consortium of Substance Abuse (VP for Rsrch)
  - Public Policy (VP for Research)
- Virtual Desktop
  - Provides access to a variety of programs through web-based system
  - Used on or off campus

# Starting the SAS System

- Off campus
  - <http://virtualdesktop.uiowa.edu/>
  - Requires installation of Citrix Receiver software
- PC installed or on campus
  - Start → All Programs → SAS → SAS 9.4
  - or
  - Start button → “S” → SAS → SAS 9.4

It's not magic... it's a tool



## Interface Windows

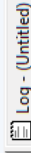
- Enhanced Editor
- Log
- Output or Results Viewer
- Explorer
- Results

## Enhanced Editor



- Where you write your SAS programs
- A SAS program is a series of commands to:
  - Import and manipulate data
  - Generate reports and perform analyses
  - Output results

## Log



- Information pertaining to the program you've submitted is automatically displayed in the log
- Contains a list of:
  - Program commands and operations
  - Notes, warnings and errors

## Output or Results Viewer



- When the SAS program executes without error, the results are displayed in the Output or Results Viewer
- The window the results will be displayed in will depend on the default setting

# Output or Results Viewer



Output

The FREQ Procedure

Status	Frequency	Percent
Alive	3218	61.78
Dead	1991	38.22

Results Viewer

The FREQ Procedure

Status	Frequency	Percent
Alive	3218	61.78
Dead	1991	38.22



# Results

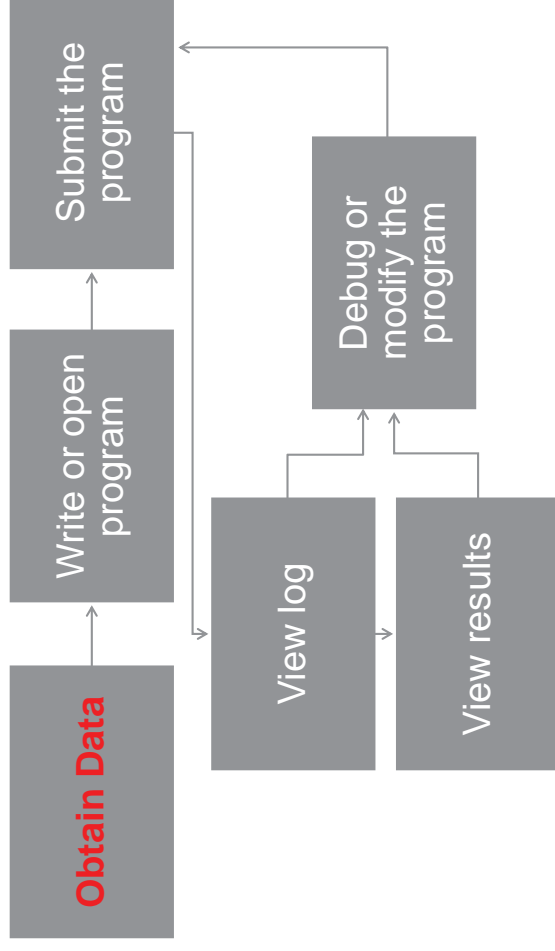


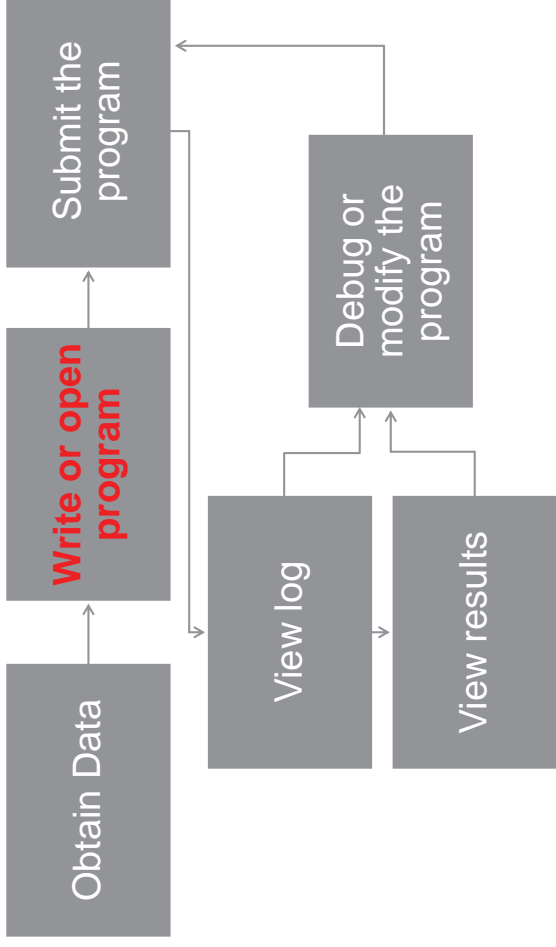
- Provides table of contents for output
- Lists each procedure in outline form
- Can be expanded to show each part

# Explorer



- Provides easy access to SAS files and data sets
- Computer  provides access to all shared devices or drives
- Libraries  contains all libraries currently defined



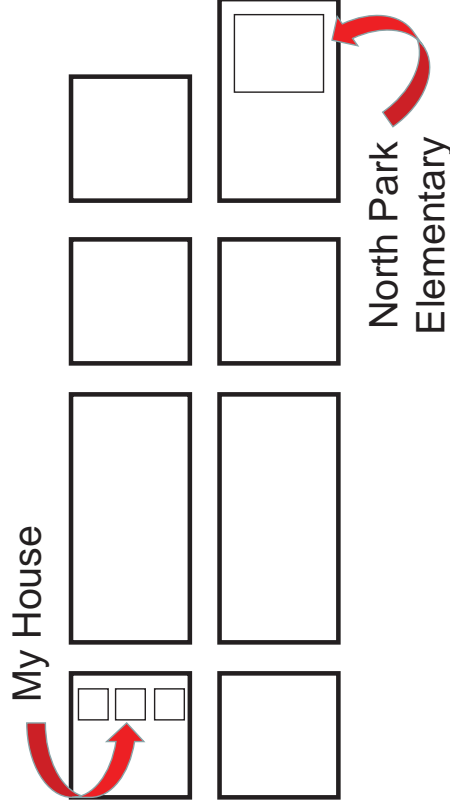


# Becoming a SAS Programmer

- SAS is best as a “write code then run” program
- To be proficient, you must learn how to write a program
  - Simple if you understand what is required



# My First Program

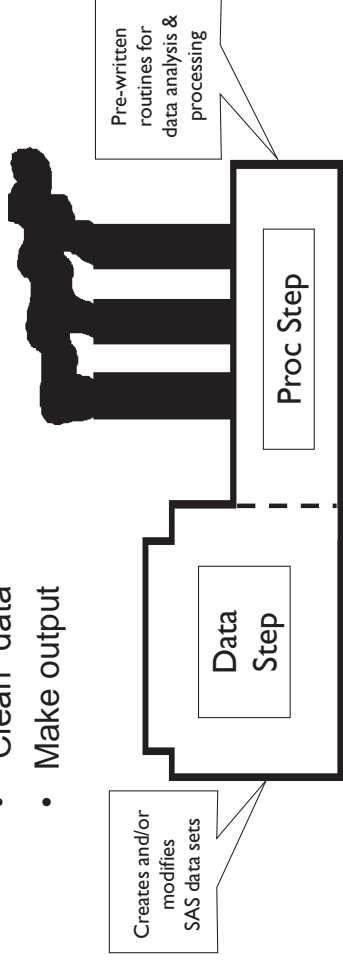


# A Basic SAS Program

- Find data
- Read data
- “Clean” data
- Make output

# A Basic SAS Program

- Find data
- Read data
- “Clean” data
- Make output



# Data (1)

## Layout

- Columns = Variables
- Rows = Observations

Bill	101	55	165.10
Tom	156	35	132.56
Sue	204	125	115.89
Ann	245	78	155.25
Jill	397	32	112.90
Bob	456	44	118.21
Tim	678	67	156.20
Matt	875	95	134.00
Kay	941	88	122.45

# Data (2)

## Variables need names

- 1-32 characters
- Must start with a character or underscore
  - Subsequent characters can be letters, numbers, or underscores
- No blanks or special characters
- Mixed-case OK
- Are not case sensitive

<u>f</u> name	<u>i</u> d	<u>d</u> ays	<u>w</u> ages
Bill	101	55	165.10
Tom	156	35	132.56
Sue	204	125	115.89
Ann	245	78	155.25
Jill	397	32	112.90
Bob	456	44	118.21
Tim	678	67	156.20
Matt	875	95	134.00
Kay	941	88	122.45

# Data (3)

## Data types

- Character
  - Can contain any character (letters, numbers, special characters, and blanks)
  - Range from 1-32,767 characters
- Numeric
  - Numbers (decimal point and minus sign)

<u>f</u> name	<u>i</u> d	<u>d</u> ays	<u>w</u> ages
Bill	101	55	165.10
Tom	156	35	132.56
Sue	204	125	115.89
Ann	245	78	155.25
Jill	397	32	112.90
Bob	456	44	118.21
Tim	678	67	156.20
Matt	875	95	134.00
Kay	941	88	122.45

# Data (4)

## Data sources

- Internal
  - Data embedded with a program
- External
  - “Local”
    - Excel, Access, delimited, text
  - “Remote”
    - Databases, servers, etc.

<u>fname</u>	<u>id</u>	<u>days</u>	<u>wages</u>
Bill	101	55	165.10
Tom	156	35	132.56
Sue	204	125	115.89
Ann	245	78	155.25
Jill	397	32	112.90
Bob	456	44	118.21
Tim	678	67	156.20
Matt	875	95	134.00
Kay	941	88	122.45

Bill	101	55	165.10
Tom	156	35	132.56
Sue	204	125	115.89
Ann	245	78	155.25
Jill	397	32	112.90
Bob	456	44	118.21
Tim	678	67	156.20
Matt	875	95	134.00
Kay	941	88	122.45

# Let's Write a Program!

# Let's Write a Program!

Use the “data” statement to tell SAS that you want to create a dataset and you want to name it “demo”.

data demo;

Bill	101	55	165.10
Tom	156	35	132.56
Sue	204	125	115.89
Ann	245	78	155.25
Jill	397	32	112.90
Bob	456	44	118.21
Tim	678	67	156.20
Matt	875	95	134.00
Kay	941	88	122.45

Use the “input” statement to tell SAS how to read in each line of the data file. This is where you provide variable names and where you tell SAS the type of each variable.

input fname \$ id days wages;

Bill	101	55	165.10
Tom	156	35	132.56
Sue	204	125	115.89
Ann	245	78	155.25
Jill	397	32	112.90
Bob	456	44	118.21
Tim	678	67	156.20
Matt	875	95	134.00
Kay	941	88	122.45

# Let's Write a Program!

# Let's Write a Program!

The "datalines" statement tells SAS that the next lines of the program actually contain data. SAS will treat each line as a new observation until it encounters a semi-colon (;)

```
data demo;
input fname $ id days wages;
datalines;
Bill 101 55 165.10
Tom 156 35 132.56
Sue 204 125 115.89
Ann 245 78 155.25
Jill 397 32 112.90
Bob 456 44 118.21
Tim 678 67 156.20
Matt 875 95 134.00
Kay 941 88 122.45
;
```

The "run" statement isn't always necessary, but it's a good practice to tell SAS that this is the end of the DATA step or PROC step.

```
data demo;
input fname $ id days wages;
datalines;
Bill 101 55 165.10
Tom 156 35 132.56
Sue 204 125 115.89
Ann 245 78 155.25
Jill 397 32 112.90
Bob 456 44 118.21
Tim 678 67 156.20
Matt 875 95 134.00
Kay 941 88 122.45
;
```

# Let's Write a Program!

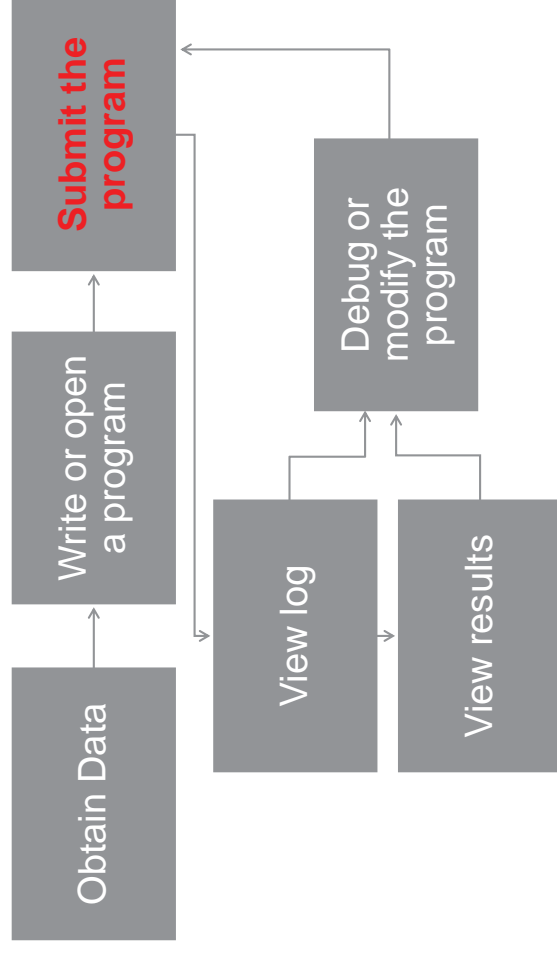
Now that our data is in a SAS dataset, we can run a simple PROC to see what the data looks like. Again, the "run" statement isn't always necessary, but it's a good practice to tell SAS that this is the end of the DATA step or PROC step.

```
data demo;
input fname $ id days wages;
cards;
Bill 101 55 165.10
Tom 156 35 132.56
Sue 204 125 115.89
Ann 245 78 155.25
Jill 397 32 112.90
Bob 456 44 118.21
Tim 678 67 156.20
Matt 875 95 134.00
Kay 941 88 122.45
;
run;
proc print;
run;
```

# Let's Write a Program!

The "run" statement isn't always necessary, but it's a good practice to tell SAS that this is the end of the DATA step or PROC step.

```
data demo;
input fname $ id days wages;
datalines;
Bill 101 55 165.10
Tom 156 35 132.56
Sue 204 125 115.89
Ann 245 78 155.25
Jill 397 32 112.90
Bob 456 44 118.21
Tim 678 67 156.20
Matt 875 95 134.00
Kay 941 88 122.45
;
```



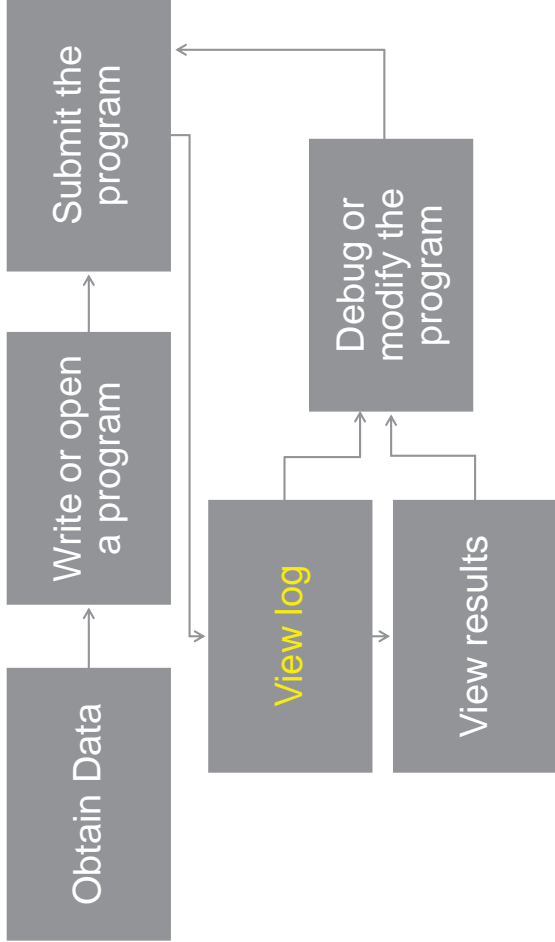


# Submitting the Program

- Can submit all or part of a program
- Click the “running man” 

# Results!

Obs	fname	id	days	wages
1	Bill	101	55	165.10
2	Tom	156	35	132.56
3	Sue	204	125	115.89
4	Ann	245	78	155.25
5	Jill	397	32	112.90
6	Bob	456	44	118.21
7	Tim	678	67	156.20
8	Matt	875	95	134.00
9	Kay	941	88	122.45



# Log

NOTE: Copyright (c) 2002-2012 by SAS Institute Inc., Cary, NC, USA.  
 NOTE: SAS (r) Proprietary Software 9.4 (TS1M3)  
 Licensed to UNIVERSITY OF IOWA - SFA T&R, Site 70086217.  
 NOTE: This session is executing on the X64\_7PRO platform.

NOTE: Additional host information:

X64\_7PRO WIN 6.1.7601 Service Pack 1 Workstation

NOTE: SAS initialization used:  
 real time 0.65 seconds  
 cpu time 0.49 seconds

1 data demo;  
 2 input frame \$ id days wages;  
 3 datalines;

NOTE: The data set WORK.DEMO has 9 observations and 4 variables.  
 NOTE: DATA statement used (Total process time):

real time 0.01 seconds  
 cpu time 0.01 seconds

13 ;  
 14 run;  
 15 proc print;  
 16 run;  
 17 run;

NOTE: There were 9 observations read from the data set WORK.DEMO.  
 NOTE: PROCEDURE PRINT used (Total process time):  
 real time 0.42 seconds  
 cpu time 0.14 seconds

# Time to do some data fixing!

# Assignment Statements

- Basic method for adding to or modifying a SAS data set
- Has the form  
Variable=expression;
  - Numeric constant  
Year=2015;
  - Character constant;  
Study="Heart";
  - Copy a variable  
Newvariable=Oldvariable;

# Arithmetic Calculations

Operation	Symbol	Example
Addition	+	CholestAdjst=Cholesterol+5;
Subtraction	-	SystAdjst=Systolic-10;
Multiplication	*	Heightm=Height*0.0254;
Division	/	BPRatio=SystAdjst/DiastAdjst
Exponentiation	**	Heightm2=Heightm**2

# Let's Write a Program!

Create a new variable named "year" and give it a constant value of 2016.

Then create a new variable named "totwages" that is the product of wages and days.

```
data demo;
input fname $ id days wages;
year=2016;
totwages=wages*days;
datalines;
Bill 101 55 165.10
Tom 156 35 132.56
~
Matt 875 95 134.00
Kay 941 88 122.45
;
run;

proc print;
run;
```

# BREAK

## Structural Components

- Every program typically has two parts:
  - DATA step
    - Reading data and variable manipulations
  - PROC step
    - Generates descriptive information and performs statistical analyses

```
data demo;
input fname $ id days wages;
datalines;
Bill 101 55 165.10
Tom 156 35 132.56
Sue 204 125 115.89
Ann 245 78 155.25
Jill 397 32 112.90
Bob 456 44 118.21
Tim 678 67 156.20
Matt 875 95 134.00
Kay 941 88 122.45
;
run;

proc print;
run;
```

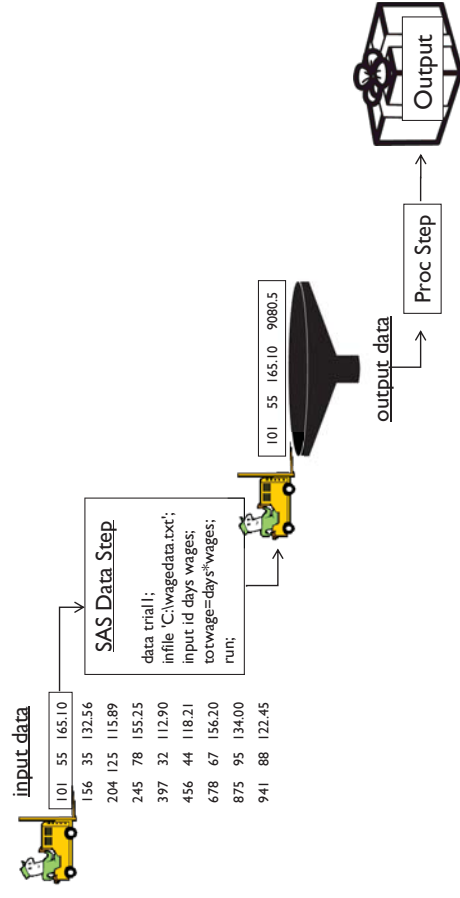
## DATA Step

- Reads and modifies data
  - Calculations
  - Recoding variables
  - Combine data sets by concatenation or merging
- Data steps execute line by line and observation by observation

```
data demo;
input fname $ id days wages;
datalines;
Bill 101 55 165.10
Tom 156 35 132.56
Sue 204 125 115.89
Ann 245 78 155.25
Jill 397 32 112.90
Bob 456 44 118.21
Tim 678 67 156.20
Matt 875 95 134.00
Kay 941 88 122.45
;
run;

proc print;
run;
```

## Structure Overview



## PROC Step

- Produces output
- Each procedure (PROC) has unique characteristics
- There are lots and lots of PROCs
- PROCs will be covered in more detail tomorrow.

```
data demo;
input fname $ id days wages;
datalines;
Bill 101 55 165.10
Tom 156 35 132.56
Sue 204 125 115.89
Ann 245 78 155.25
Jill 397 32 112.90
Bob 456 44 118.21
Tim 678 67 156.20
Matt 875 95 134.00
Kay 941 88 122.45
;
run;

proc print;
run;
```

## Let's Write another Program!

- Read in an “external” data file
  - C:\SASClass\bp.csv
  - Data on clinic and diastolic and systolic blood pressure at initial and follow-up visit.
- CSV: comma-separated values
  - Common data format
  - Easily imported/exported from Excel

C,84,138,93,143  
D,89,150,91,140  
,78,116,100,162  
A,,86,155  
C,81,145,86,140

## Let's Write another Program!

Use the “data” statement to tell SAS that you want to create a dataset and you want to name it “bp”.

Use the “infile” statement to tell SAS the name and location of the external data file. Also tell SAS that the data values are delimited with a comma.

C,84,138,93,143  
D,89,150,91,140  
,78,116,100,162  
A,,86,155  
C,81,145,86,140

## Let's Write another Program!

C,84,138,93,143  
D,89,150,91,140  
,78,116,100,162  
A,,86,155  
C,81,145,86,140

# Let's Write another Program!

Use the "input" statement to tell SAS how to read in each line of the data file. This is where you provide variable names and where you tell SAS the type of each variable.

```
data bp;
infile 'c:\sasclass\bp.csv' dsd;
input clinic $ dbp1 sbp1 dbp2 sbp2;
```

C,84,138,93,143  
D,89,150,91,140  
,78,116,100,162  
A,,86,155  
C,81,145,86,140

# Let's Write another Program!

Again, the "run" statement isn't always necessary, but it's a good practice to tell SAS that this is the end of the DATA step or PROC step.

```
data bp;
infile 'c:\sasclass\bp.csv' dsd;
input clinic $ dbp1 sbp1 dbp2 sbp2;
run;
proc print;
run;
```

Now that our data is in a SAS dataset, we can run a simple PROC to see what the data looks like.

C,84,138,93,143  
D,89,150,91,140  
,78,116,100,162  
A,,86,155  
C,81,145,86,140

# Programs and Outputs and Logs!

(oh my)

# Missing Data

Obs	clinic	dbp1	sbp1	dbp2	sbp2
1	C	84	138	93	143
2	D	89	150	91	140
3		78	116	100	162
4	A	.	.	86	155

- Character variables “ ”
- Numeric variables .

## Time to do some (more) data fixing!

## Let's Write another Program!

```
data bp;
infile 'c:\sasclass\bp.csv' dsd;
input clinic $ dbp1 sbp1 dbp2 sbp2;
if clinic=' ' then clinic='B';
if dbp2=. Then dbp2=60;
run;

proc print;
run;
```

"Fix" the record with the missing value for clinic – set it to "B"

Correct the record with the missing dbp2 variable.

```
C,84,138,93,143
D,89,150,91,140
A,,78,116,100,162
A,,86,155
C,,81,145,86,140
```

## “Libraries” and the Libname Statement

- Must submit a libname statement to create a library reference
- Is a pointer to folder on your computer where the data files are stored
- Short hand way of telling SAS where to look for SAS data sets
  - General Format
    - libname <name of library> "<folder location>";
  - Example
    - libname class "H:\SASUsersGroup\datasets\";

## Libname Rules

- 1-8 characters
- Must start with a letter
  - Subsequent characters can be letters, numbers or an underscore
- No spaces

# Let's Write more Program!

```
data bp;
infile 'c:\sasclass\bp.csv' dsd;
input clinic $ dbp1 sbp1 dbp2 sbp2;
if clinic=' ' then clinic='B';
if dbp2= Then dbp2=60;
run;
```

C,84,138,93,143  
D,89,150,91,140  
,78,116,100,162  
A,,,86,155  
C,81,145,86,140

```
proc print;
run;
```

```
libname sdd 'h:\sas';
```

```
data sdd.bp;
set bp;
run;
```

Use the "libname" statement to create a library name and to tell SAS where to find that library

Tell SAS what name you would like to give your "permanent" dataset. Note the two-part name (beginning with the library name).

```
data bp;
infile 'c:\sasclass\bp.csv' dsd;
input clinic $ dbp1 sbp1 dbp2 sbp2;
if clinic=' ' then clinic='B';
if dbp2= Then dbp2=60;
run;
```

C,84,138,93,143  
D,89,150,91,140  
,78,116,100,162  
A,,,86,155  
C,81,145,86,140

```
proc print;
run;
```

```
libname sdd 'h:\sas';
```

```
data sdd.bp;
set bp;
run;
```

# Let's Write more Program!

```
data bp;
infile 'c:\sasclass\bp.csv' dsd;
input clinic $ dbp1 sbp1 dbp2 sbp2;
if clinic=' ' then clinic='B';
if dbp2= Then dbp2=60;
run;
```

C,84,138,93,143  
D,89,150,91,140  
,78,116,100,162  
A,,,86,155  
C,81,145,86,140

```
proc print;
run;
```

```
libname sdd 'h:\sas';
```

```
data sdd.bp;
set bp;
run;
```

Tell SAS what dataset you would like to use for the source of your "permanent" dataset.

# Let's Write more Program!

## Rules for SAS Statements

- Begin and end in any column
- Must end with a semicolon (;)
- May consist of more than one line
- Multiple statements may appear on a single line
- One or more blanks should be placed between items
- Unquoted items can be any case

# Enhanced Editor

Editor - Untitled1

- Color coded to help you detect errors

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

# Log

Log - (Untitled)

- **Notes**
  - Additional information; an indicator of a problem
- **Warnings**
  - Program still executes but possibly not the way you expected
- **Errors**
  - Usually the result of a syntax or spelling error

# Correcting Errors Checklist

- Read the Log
- Test each part of the program
- Test program using small data sets
- Be observant of the colors in your program

# Common Programming Errors

- No semicolon at the end of a statement
- Missing or mismatched quotation marks
- Misspellings
- Using the letter 'o' instead of number 0



## Correcting DATA Errors

- Data entry errors
  - Descriptive summaries
  - Create flags to alert you of errors
- SAS coding errors
  - Spot check data

## Let's Write (yet) another Program!

- Read in a SAS Dataset
  - C:\SASclass\sample.sas7bdat
  - Data on patients and clinical characteristics.
  - It's already a SAS dataset – somebody has already done a lot of the work!

## Let's Write (yet) another Program!

```
libname ssd 'c:\sasclass\';
```

Use the "libname" statement to tell SAS to create a library name and to tell SAS where to find that library

## Let's Write (yet) another Program!

```
proc print data=ssd.sample;  
run;
```

"LOOK!" A SAS program that doesn't have a data step!  
Use the "data=" option on the print proc to tell SAS which dataset you want to print.

# Time to do some (yet more) data fixing!

# Let's Write (yet) another Program!

```
libname ssd 'c:\sasclass\';  
proc print data=ssd.sample;  
run;
```

Tell SAS that you want to create a new dataset and name it "stuff". Note, the one-part name tells SAS that this is a temporary dataset.

# Let's Write (yet) another Program!

```
libname ssd 'c:\sasclass\';  
proc print data=ssd.sample;  
run;
```

Use the "set" statement to tell SAS the name of the dataset that you want to use as a "source" for your new dataset.

```
data stuff;  
set ssd.sample;  
if cholesterol=999 then cholesterol=;
```

# Let's Write (yet) another Program!

```
libname ssd 'c:\sasclass\';  
proc print data=ssd.sample;  
run;
```

# Let's Write (yet) another Program!

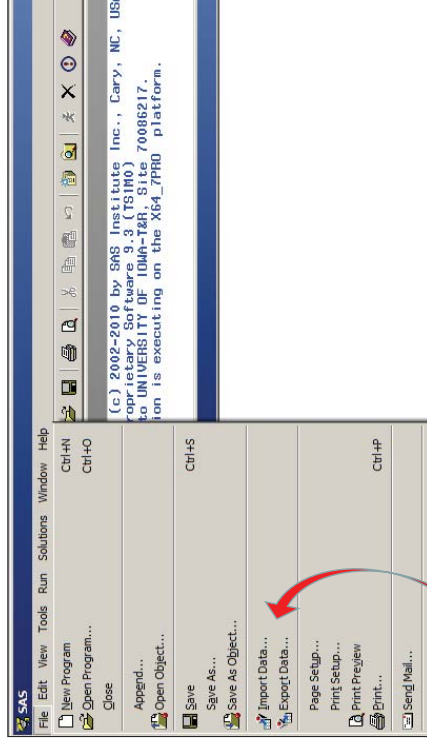
```
libname ssd 'c:\sasclass\';  
  
proc print data=ssd.sample;  
run;  
  
data stuff;  
set ssd.sample;  
if cholesterol=999 then cholesterol=;  
run;  
  
proc print data=stuff;  
run;
```

Use a "run" statement to finish the data step.  
Look at the new dataset using a Proc Print.

## Import/Export Data

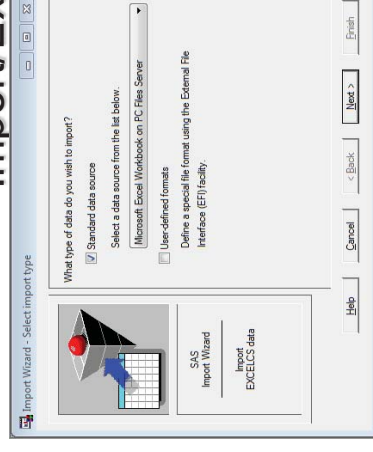
- SAS can import data from, and export data to, many different formats
  - MS-Excel
  - MS-Access
  - .csv
  - SPSS
  - Stata
  - many others
- A variety of methods for importing/exporting
- Best approach depends on variety of factors
  - Operating system (Linux, Windows, 32/64-bit)
  - SAS version (9.2, 9.3, 32/64-bit)
  - Originating/destination software (Excel, .csv, SPSS)
- Use the Wizard
- Be careful, pay attention

## Import/Export Data (2)

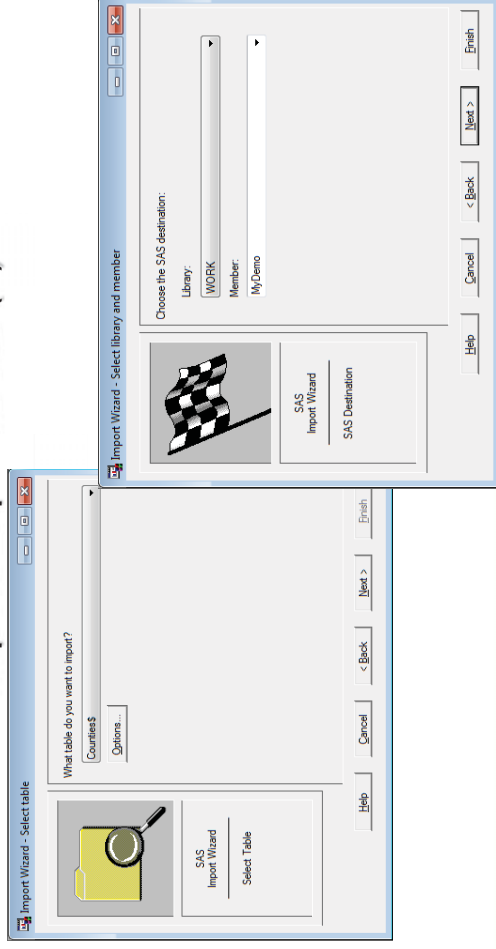


**Wizards!**

## Import/Export Data (3)



## Import/Export Data (4)



## Import/Export Data (5)

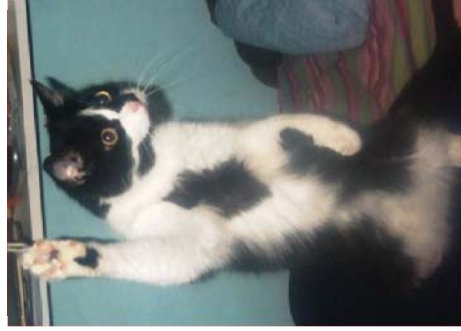


```

PROC IMPORT OUT= WORK.demo
  DATAFILE= "H:\My Documents\SAS\UI SAS bootcamp\2017\demos\patient.xlsx"
  DBMS=EXCELX REPLACE;
  RANGE="Sheet1$";
  SCANTEXT=YES;
  USEDATE=YES;
  SCANTIME=YES;
RUN;

```

Questions?



Evaluate!



# Food?

