

Please login

- Take a seat
- Login with your HawkID
- Locate SAS 9.4
 - Start / All Programs / SAS / SAS 9.4 (64 bit)
- Raise your hand if you need assistance

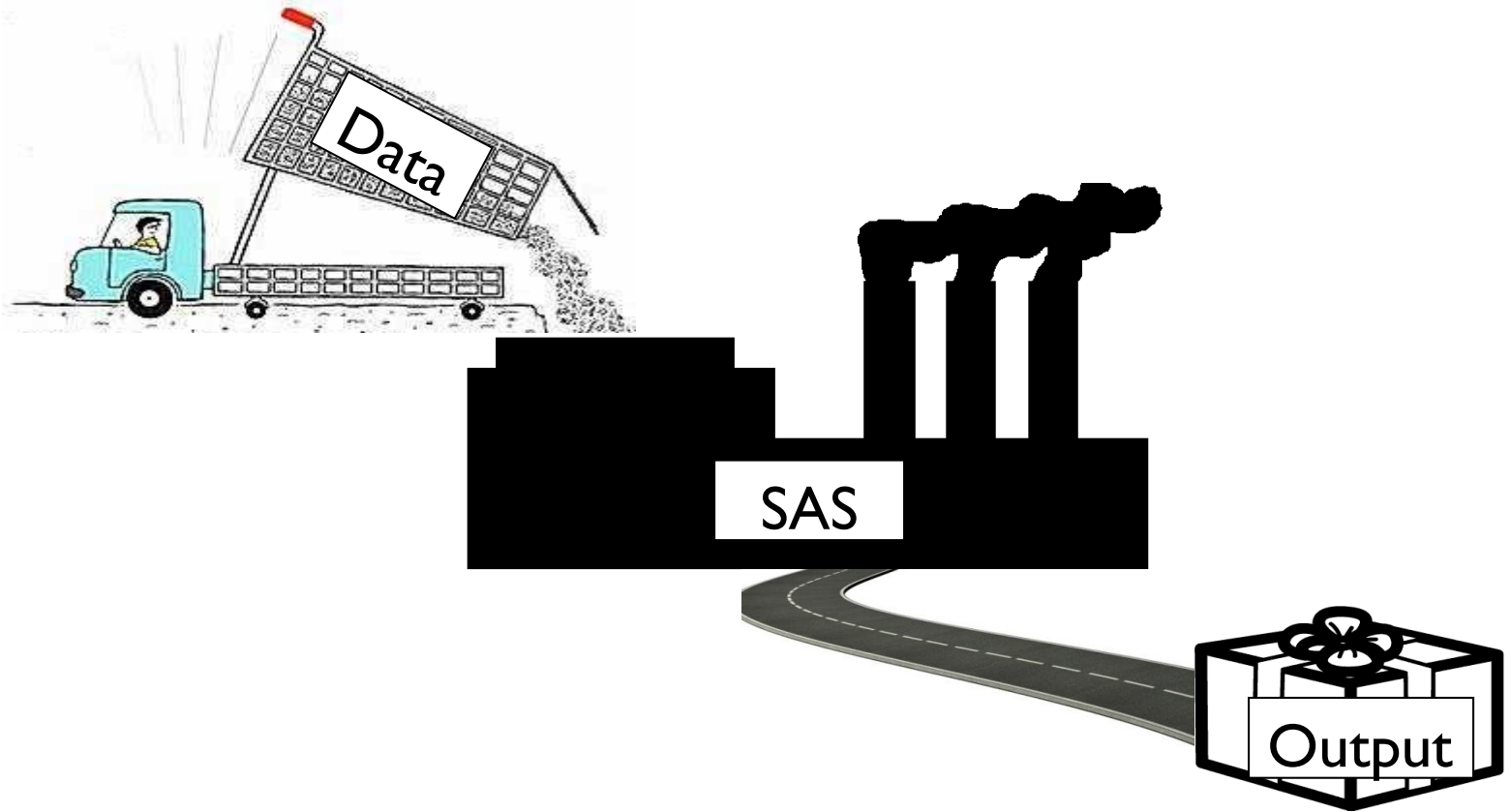
Introduction to Data Management in SAS

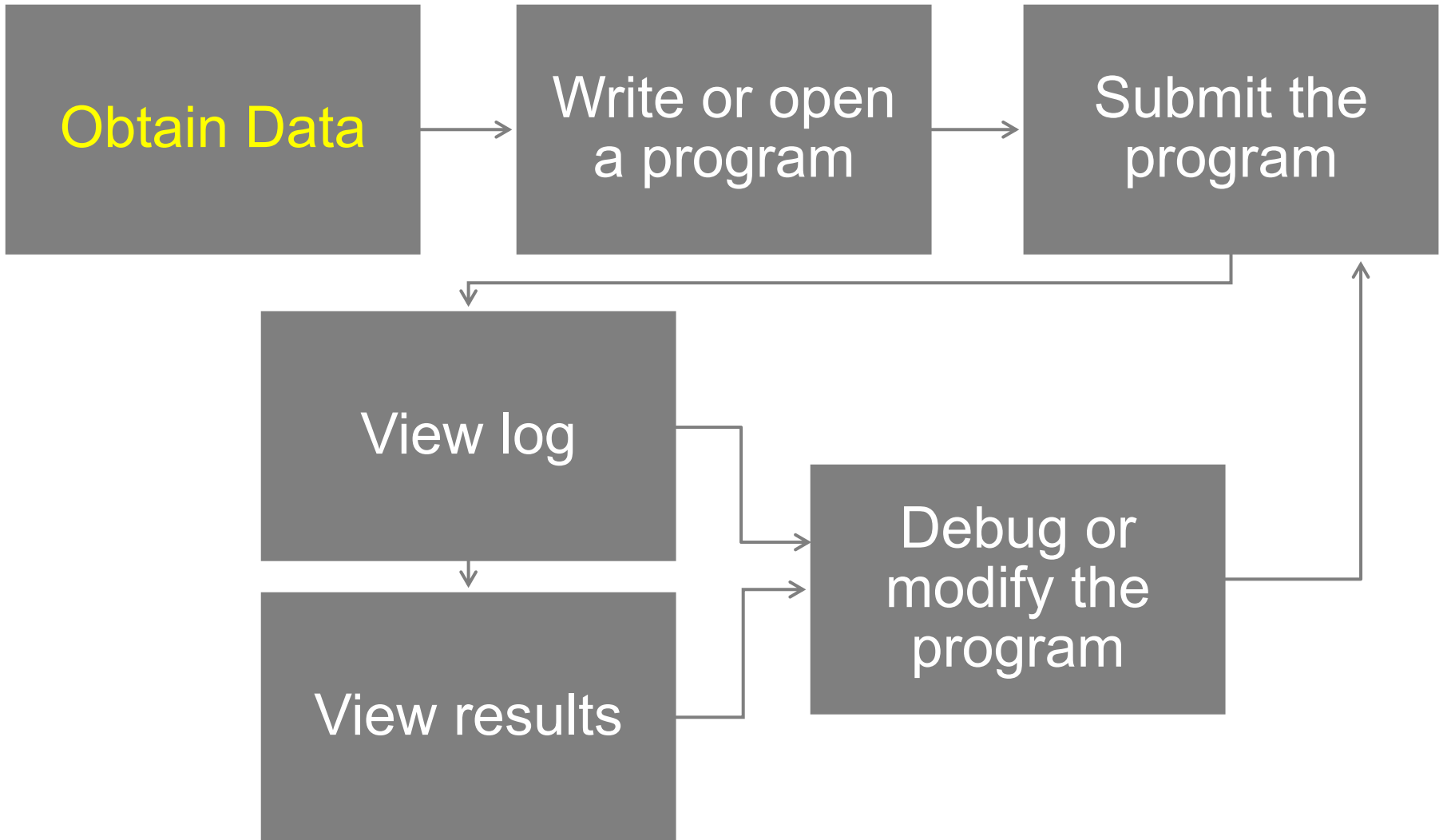
Michael Brumm

Overview

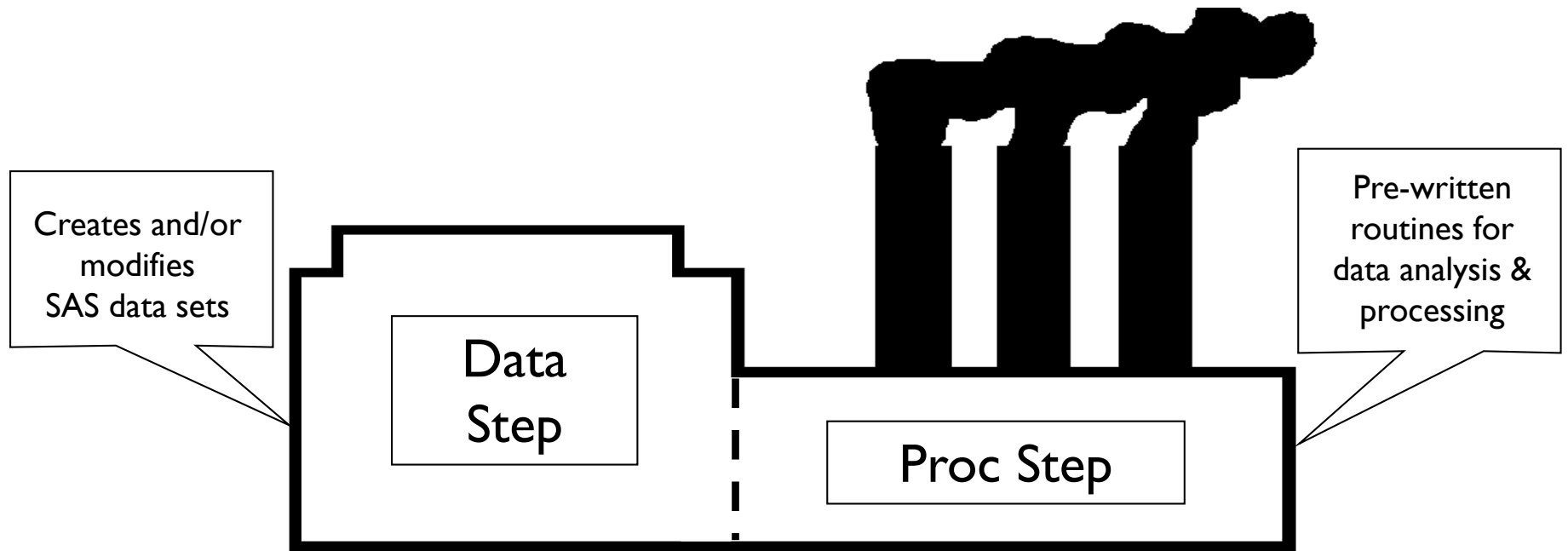
- Review
- Labels & Formats
- Missing Data
- Expressions & Functions
- If-Then/Else & Do...End
- Sub-setting
- Exporting

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

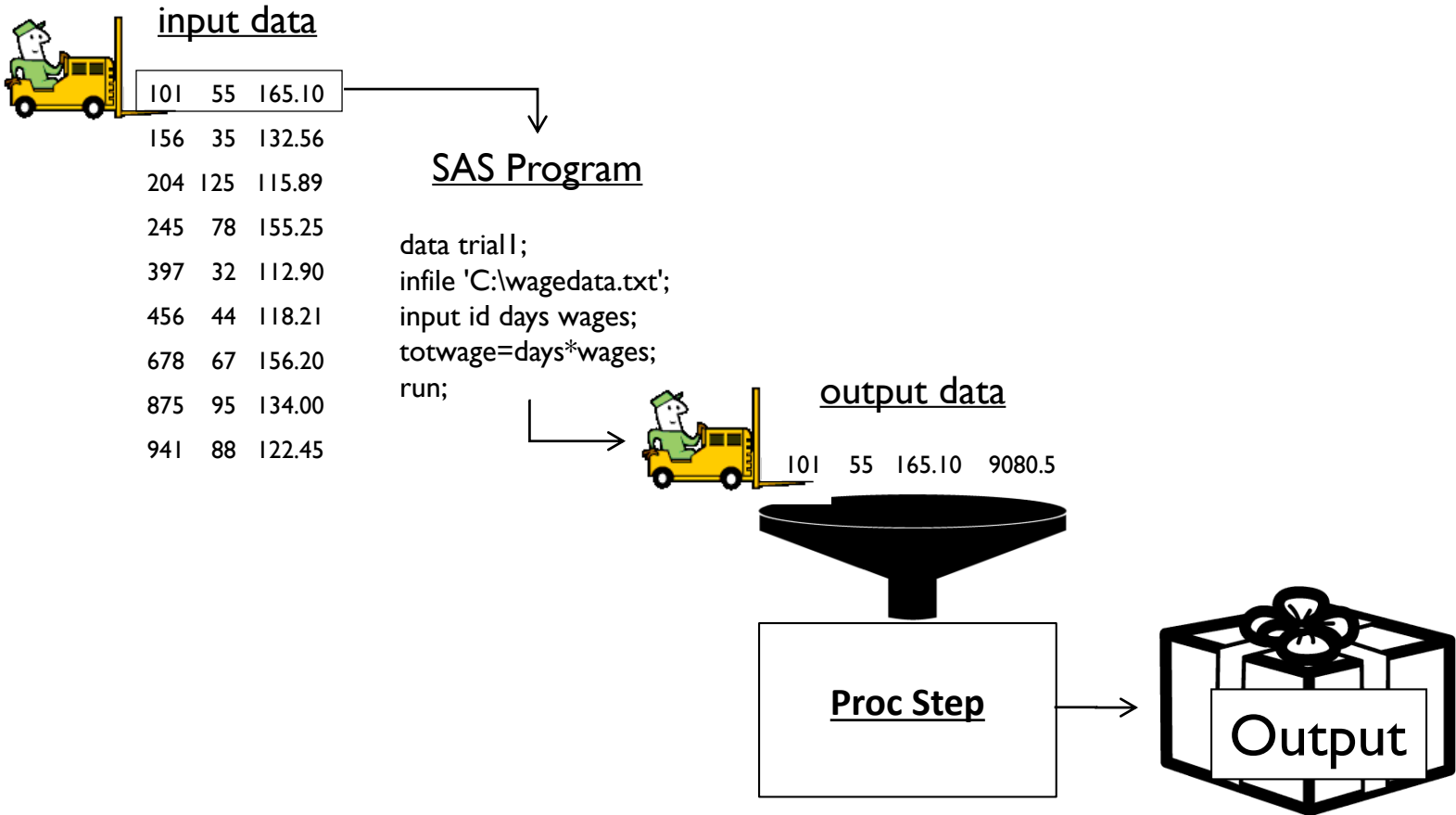




A Basic SAS Program



Structure Overview



LIBNAME statement

- General Format

- `libname` <name of library> "<folder location>";

- Example

- `libname` class "H:\SASClass\";

- `data` class.session2; ←————— Creates “permanent” data set
 `set` class.newheart;
 `run`;

- `data` session2; ←————— Creates temporary “work” data set
 `set` class.newheart;
 `run`;

SAS Help

- Press 'F1' to access internal SAS Help
- Web Help:
<http://support.sas.com/documentation/>

Questions?

Variable Labels

- The LABEL Statement is a data step statement which assigns descriptive labels to be displayed in place of variable names
- *Syntax:* LABEL variable-1=label-1 ... variable-n=label-n;

Variable Labels

```
data class.session2;  
  set class.session2;  
  label sysbp = "Initial Systolic Blood Pressure"  
        diasbp = "Initial Diastolic Blood Pressure";  
run;
```

Before Label

sysbp	diasbp
57	38
75	30
76	50

After Label

Initial Systolic Blood Pressure	Initial Diastolic Blood Pressure
57	38
75	30
76	50

Variable Formats

- A format controls the written appearance of data values; it specifies how SAS displays the values of a particular variable
- Types of Formats

Category	Description
Character	instructs SAS to write character data values from character variables.
Date and Time	instructs SAS to write data values from variables that represent dates, times, and datetimes.
ISO 8601	instructs SAS to write date, time, and datetime values using the ISO 8601 standard.
Numeric	instructs SAS to write numeric data values from numeric variables.

Variable Formats

- Variables with values that include letters are always assigned character formats
- Variables comprised exclusively of numbers can be assigned either numeric or character formats

Applying a Character Format

- *Syntax:* \$UPCASEw.
- \$ indicates a character format is being applied, UPCASE specifies the type of format, and w specifies the width of the output field

```
data class.session2;  
    set class.session2;  
    format State $UPCASE2.;  
run;
```

Before Applying Format

State
Mo
Vt
Pa
Ca

After Applying Format

State
MO
VT
PA
CA

Applying a Numeric Format

- *Syntax:* w.d (standard numeric format)
- The w specifies the width, and the d specifies the number of digits to the right of the decimal point

```
data class.session2;  
      set class.session2;  
      format BMI 4.1;  
run;
```

Before Applying Format

Body Mass Index
25.54051
24.02398
22.1429

After Applying Format

Body Mass Index
25.5
24.0
22.1

Applying a Date Format

- *Syntax:* mmddy10.
- Writes date values in the form mm/dd/yyyy

```
data class.session2;  
    set class.session2;  
    format admitdate disdate fdate mmddy10.;  
run;
```

Before Applying Format

Hospital Admission Date	Hospital Discharge Date	Date of Last Follow Up
13JAN1997	18JAN1997	11JAN2003
19JAN1997	24JAN1997	19JAN2003
01JAN1997	06JAN1997	06JAN2003
17FEB1997	27FEB1997	11DEC1997

After Applying Format

Hospital Admission Date	Hospital Discharge Date	Date of Last Follow Up
01/13/1997	01/18/1997	01/11/2003
01/19/1997	01/24/1997	01/19/2003
01/01/1997	01/06/1997	01/06/2003
02/17/1997	02/27/1997	12/11/1997

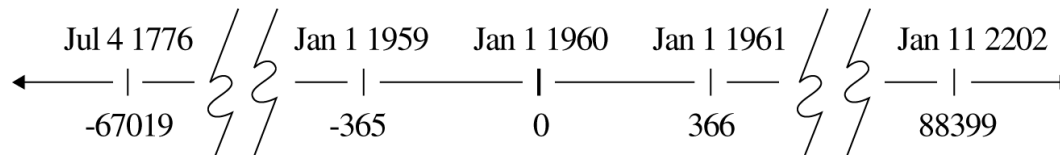
Missing Data

- SAS stores missing values in two ways:
 - Missing numeric and date values are represented by a period (.)
 - Missing character values are represented by a blank enclosed in quotes (' ')
- Numeric and date values
 - Arithmetic operations with a missing value will result in a missing return value
 - Missing values are less than non-missing values

Date Variables

- A SAS date value represents the number of days between January 1, 1960, and a specified date
- Dates before January 1, 1960, are negative; dates after are positive

Calendar Date



SAS Date Value

Numeric Expressions

- Arithmetic operators indicate that an arithmetic calculation is performed
- Reminder: if a value is missing in the data, the result is a missing value

Operation	Symbol
Addition	+
Subtraction	-
Multiplication	*
Division	/
Exponentiation	**

Numeric Expressions

- When performing arithmetic operations, understanding the order of operations is very important

Order	Operations
1	Parentheses
2	Exponents
3	Multiplication and Division
4	Addition and Subtraction

Numeric Expressions

- Mean Arterial Blood Pressure
- Create a new variable, MAP, using the below formula:

- $$\frac{\textit{Systolic BP} + 2 * \textit{Diastolic BP}}{3}$$

Common Numeric Functions

Function	Description
<u>ABS</u>	Returns the absolute value.
<u>CEIL</u>	Returns the smallest integer that is greater than or equal to the argument, fuzzed to avoid unexpected floating-point results.
<u>FLOOR</u>	Returns the largest integer that is less than or equal to the argument, fuzzed to avoid unexpected floating-point results.
<u>INT</u>	Returns the integer value, fuzzed to avoid unexpected floating-point results.
<u>INTCK</u>	Returns the number of interval boundaries of a given kind that lie between two dates, times, or datetime values.
<u>IQR</u>	Returns the interquartile range.
<u>LOG</u>	Returns the natural (base e) logarithm.
<u>MAX</u>	Returns the largest value.
<u>MEAN</u>	Returns the arithmetic mean (average).
<u>MEDIAN</u>	Returns the median value.
<u>MIN</u>	Returns the smallest value.
<u>N</u>	Returns the number of nonmissing numeric values.

Common Numeric Functions

Function	Description
<u>NMISS</u>	Returns the number of missing numeric values.
<u>PCTL</u>	Returns the percentile that corresponds to the percentage.
<u>PUT</u>	Returns a value using a specified format.
<u>QUANTILE</u>	Returns the quantile from a distribution when you specify the left probability (CDF).
<u>RAND</u>	Generates random numbers from a distribution that you specify.
<u>RANGE</u>	Returns the range of the nonmissing values.
<u>ROUND</u>	Rounds the first argument to the nearest multiple of the second argument.
<u>SQRT</u>	Returns the square root of a value.
<u>STD</u>	Returns the standard deviation of the nonmissing arguments.
<u>STDERR</u>	Returns the standard error of the mean of the nonmissing arguments.
<u>SUM</u>	Returns the sum of the nonmissing arguments.
<u>VAR</u>	Returns the variance of the nonmissing arguments.

Common Date & Time Functions

Function	Description
<u>DATDIF</u>	Returns the number of days between two dates after computing the difference between the dates according to specified day count conventions.
<u>DATEPART</u>	Extracts the date from a SAS datetime value.
<u>DATETIME</u>	Returns the current date and time of day as a SAS datetime value.
<u>DAY</u>	Returns the day of the month from a SAS date value.
<u>DHMS</u>	Returns a SAS datetime value from date, hour, minute, and second values.
<u>MONTH</u>	Returns the month from a SAS date value.
<u>TIMEPART</u>	Extracts a time value from a SAS datetime value.
<u>TODAY</u>	Returns the current date as a numeric SAS date value.
<u>WEEK</u>	Returns the week-number value.
<u>YEAR</u>	Returns the year from a SAS date value.
<u>YRDIF</u>	Returns the difference in years between two dates according to specified day count conventions; returns a person's age.

Common Character Functions

Function	Description
<u>CATS</u>	Removes leading and trailing blanks, and returns a concatenated character string.
<u>CATX</u>	Removes leading and trailing blanks, inserts delimiters, and returns a concatenated character string.
<u>CMISS</u>	Counts the number of missing arguments.
<u>COMPRESS</u>	Returns a character string with specified characters removed from the original string.
<u>COUNT</u>	Counts the number of times that a specified substring appears within a character string.
<u>COUNTC</u>	Counts the number of characters in a string that appear or do not appear in a list of characters.
<u>COUNTW</u>	Counts the number of words in a character string.
<u>FIND</u>	Searches for a specific substring of characters within a character string.
<u>INDEX</u>	Searches a character expression for a string of characters, and returns the position of the string's first character for the first occurrence of the string.
<u>INPUT</u>	Returns the value that is produced when SAS converts an expression by using the specified informat.
<u>LAG</u>	Returns values from a queue.

Common Character Functions

Function	Description
<u>LENGTH</u>	Returns the length of a non-blank character string, excluding trailing blanks, and returns 1 for a blank character string.
<u>LOWCASE</u>	Converts all uppercase single-width English alphabet letters in an argument to lowercase.
<u>PROPCASE</u>	Converts all words in an argument to proper case.
<u>SCAN</u>	Returns the nth word from a character string.
<u>STRIP</u>	Returns a character string with all leading and trailing blanks removed.
<u>SUBSTR (right of =)</u>	Extracts a substring from an argument.
<u>TRANSLATE</u>	Replaces specific characters in a character expression.
<u>TRANSTRN</u>	Replaces or removes all occurrences of a substring in a character string.
<u>TRANWRD</u>	Replaces all occurrences of a substring in a character string.
<u>TRIM</u>	Removes trailing blanks from a character string, and returns one blank if the string is missing.
<u>UPCASE</u>	Converts all lowercase single-width English alphabet letters in an argument to uppercase.

If-Then/Else Statements

- An IF-THEN/ELSE Statement executes a SAS statement for observations that meet specific conditions
- Statements can be nested to produce a series of logical evaluations, which stop once a true statement is encountered
- More computationally efficient than repeated IF statements

If-Then/Else Statements

- *Syntax*: **IF** expression **THEN** statement;
<**ELSE** statement;>
 - *expression* can be any logical SAS expression
 - *THEN statement* specifies the statement that is executed if expression is true
 - *ELSE statement* (optional) specifies the statement that is executed if expression is false.

Length Statement

- The LENGTH Statement is a data step statement specifying the internal storage lengths of variables
- Length can only be set *prior* to the assignment of values
- *Syntax*: **LENGTH** variable-1 <\$>length ... variable-n <\$>length;
 - \$ (optional) specifies that the preceding variables are character variables
 - *length* specifies a numeric constant that is the number of bytes used for storing variable values

Length Statement

```
data class.session2;  
  set class.session2;  
  length Symptoms $11.;  
  if symptoms_sum = 0 then Symptoms = "None";  
  else if symptoms_sum = 1 then Symptoms = "One";  
  else if symptoms_sum > 1 then Symptoms = "Two or  
  more";  
  
run;
```

With LENGTH Statement

Symptoms
None
None
None
None
One
Two or more

Without LENGTH Statement

Symptoms
None
None
None
None
One
Two

Do...End Statements

- The DO Statement specifies a group of statements to be executed as a unit.
- A simple DO statement is often used within IF-THEN/ELSE statements to designate a group of statements to be executed depending on whether the IF condition is true or false.

Do...End Statements

- *Syntax:* **DO;**
 statement-1;
 ...
 statement-n;
END;

```
data class.session2;  
    set class.session2;  
    if dstat = 0 then do;  
        dstat_text = "Alive";  
        TotalTime = LOS + OSTime;  
    end;  
    else if dstat = 1 then do;  
        dstat_text = "Dead";  
        TotalTime = LOS;  
    end;  
run;
```

Sub-Setting By Observation

- The DELETE Statement can be used to exclude observations that meet specified criteria

```
data subset_delete;  
    set class.session2;  
    if OSTime = . then delete;  
  
run
```

Sub-Setting By Observation

- The OUTPUT Statement can be used to only include observations that meet specified criteria

```
data subset_output;  
    set class.session2;  
    if gender = 1 then output;  
run;
```

Sub-Setting By Variable

- The DROP Statement can be used to delete unwanted variables

```
data subset_drop;  
    set class.session2;  
    drop Title Address ZipCode Occupation;  
run;
```

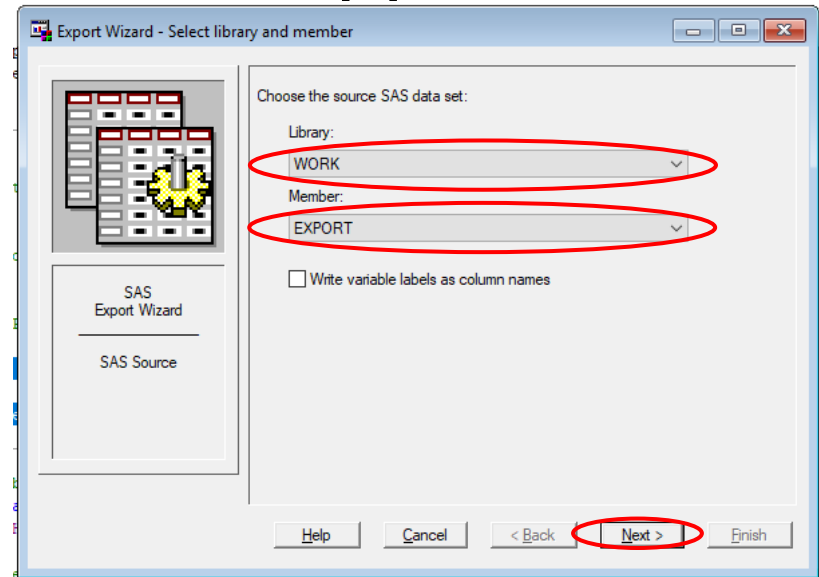
Sub-Setting By Variable

- The KEEP Statement can be used to keep only the variables you want

```
data subset_keep;  
    set class.session2;  
    keep ID age gender fdate;  
run;
```

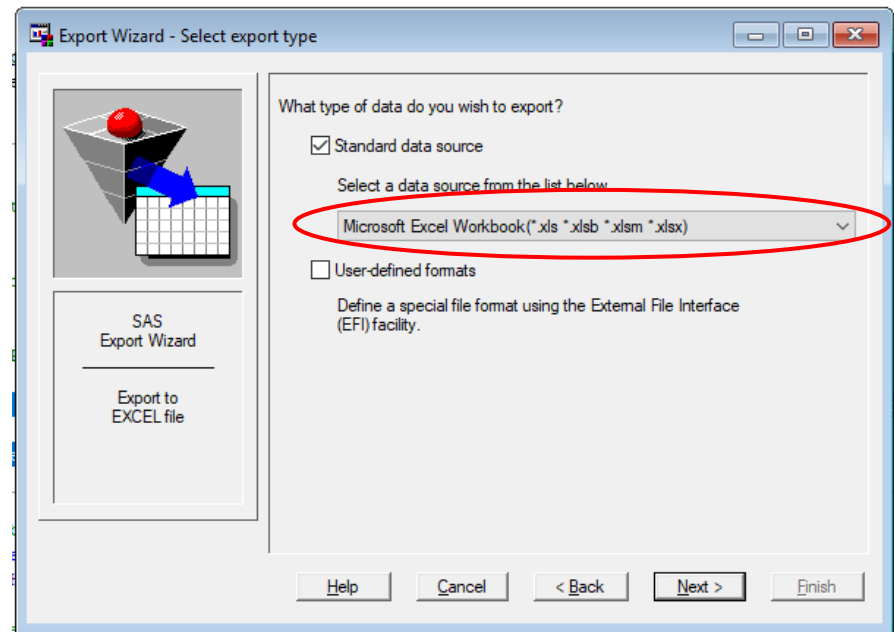
Export Wizard

- Select File » Export Data
- The window dialog box will appear
- Select the SAS library
- Select the member of the library
- Click Next



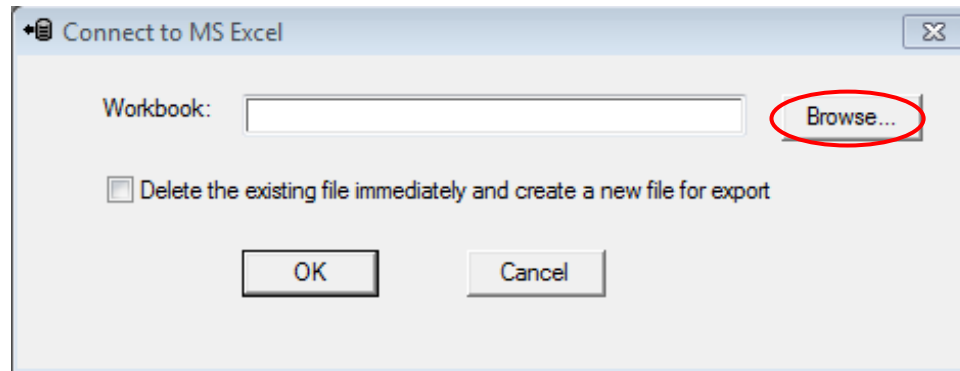
Export Wizard

- The “Select Export Type” dialog box will appear
- Select a data source you want to export to



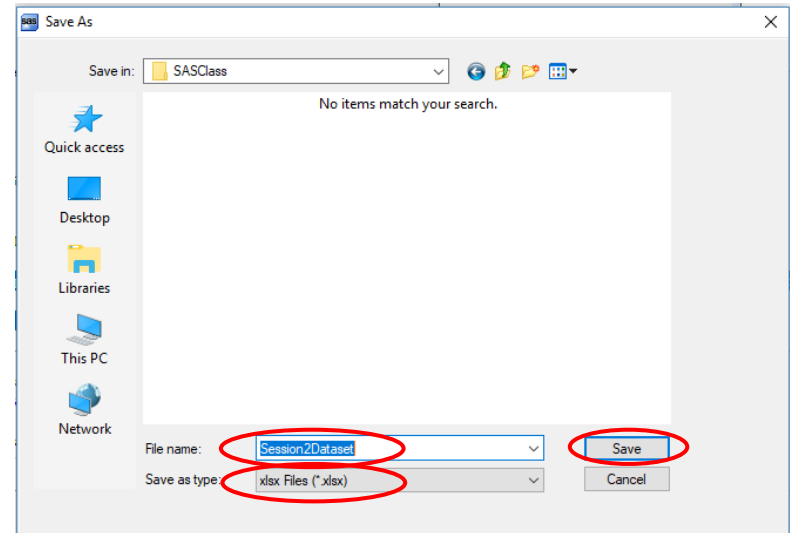
Export Wizard

- Click the Browse button to locate the folder where you want the data saved
 - H:\SASClass\



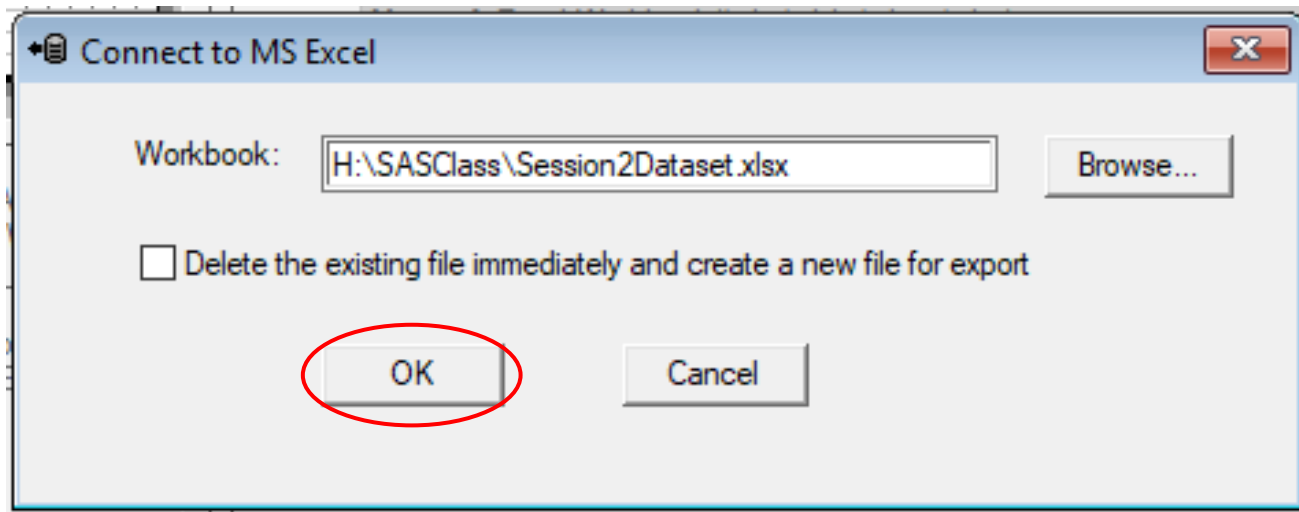
Export Wizard

- Enter File Name
- Be sure to save as type .xlsx file
 - Default is old version of Microsoft office
- Click Save



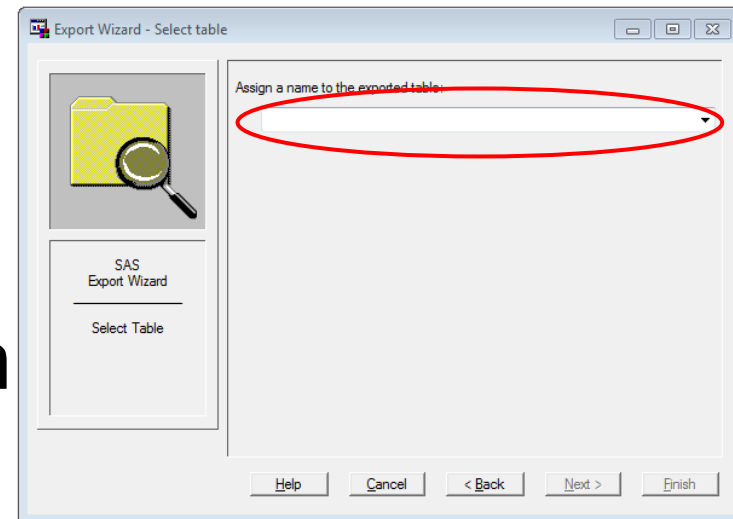
Export Wizard

- Click OK



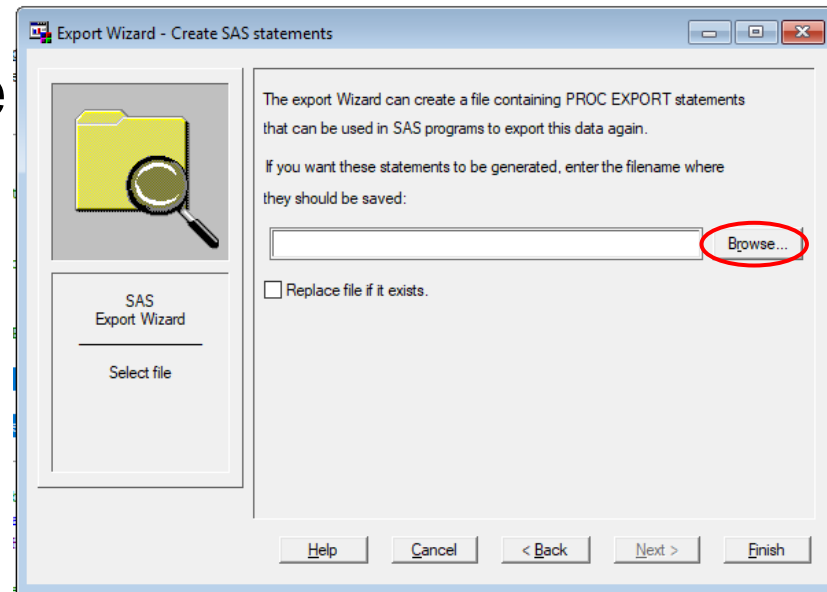
Export Wizard

- Assign a name to the exported table
 - This names the sheet in excel
- If you want to save the export code, click Next
- Otherwise, click Finish



Export Wizard

- To save the export code, click on Browse to:
 - Select the folder the program is saved to
 - Name the program
- Click finish



Export Wizard

- Export code created by export wizard

```
PROC EXPORT DATA= WORK.EXPORT  
    OUTFILE="H:\SASClass\Session2Dataset.xlsx"  
    DBMS=EXCEL REPLACE;  
    SHEET="Data";  
RUN;
```

Using ODS

- SAS ODS (Output Delivery System) provides flexibility in generating reports
- Great for printing graphs or plots

```
ods rtf file="H:\odsoutput.rtf";  
proc print data = subset_keep noobs;  
    where age > 90;  
run;  
ods rtf close;
```

Questions?

Lunch Options

