SAS and Data Management

Kim Magee
Department of Biostatistics
College of Public Health

The University of Iowa

Review of Previous Material

Review

- **INFILE** statement

```
data bp;
infile 'c:\sas\bp.csv' dlm=',';
input clinic $ dbp1 sbp1 dbp2 sbp2;
run;
```

Name the dataset you are going to create

Import the datafile

Give the names of the variables(columns)

Indicates the variable in front of it is a character variable

Review (cont.)

- **LIBNAME** statement

- General Format
  - `libname <name of library> "<folder location>";`
- Example
  - `libname ssi "H:\SSIClass\";`

```
data class.bp;
  set bp;
run;
```

Create "permanent" dataset

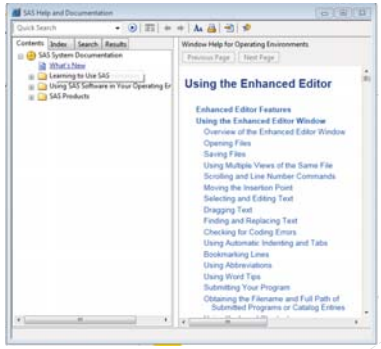
```
data bp;
  set class.bp;
run;
```

Create temporary "work" dataset from "permanent dataset in library

SAS Help

Internal Help

- Press "F1" to access Internal Help

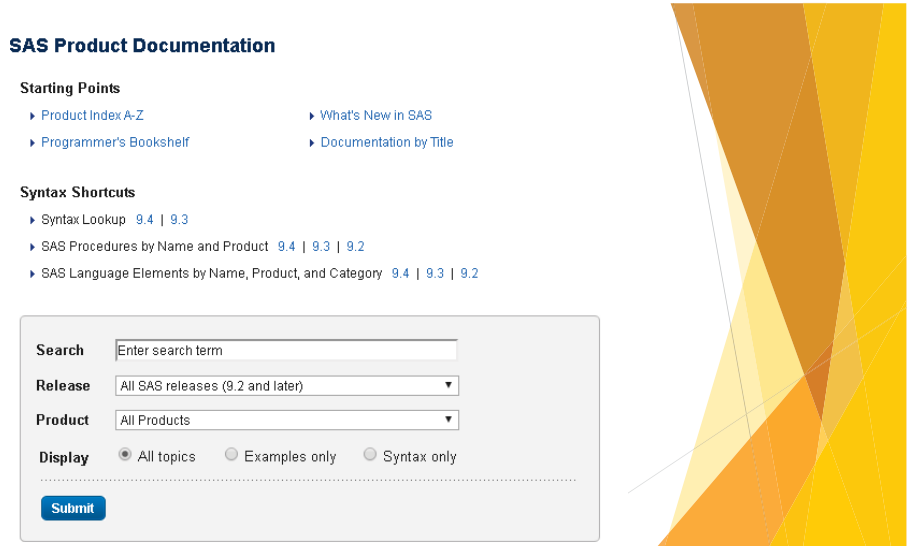


Questions?

SAS Help

Online Help

- http://support.sas.com/documentation/



Missing Data

- SAS puts "." in place of a Missing Value
- Arithmetic operations with a missing value will result in a missing return value
- In logical operations missing values are equal to other missing values
  - Missing values are less than non-missing values
  - e.g. when excluding missing data you can use the line below, assuming the value of your data is non-negative

```
data missing;
    set data;
    *if OSTime < 0;
    if Missing(OSTime);
run;
```

- Note: when exporting data with missing values to Excel, the missing values will show up as "." not blank

Formats

SAS 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.

SAS Format List

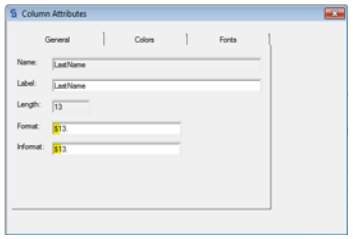
Formats (cont.)

- Letters are always character formats
- Numbers can be either numerical format or character format
  - Be careful when performing equations or merging data that they are in the correct format

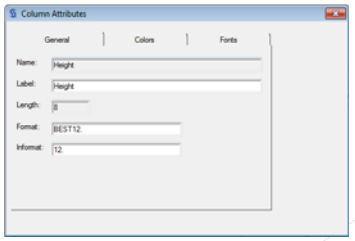
Formats (cont.)

Double click the Column Header in your Data set to see Column Attributes

Character Format



Numerical Format



Numeric Expressions

Operators in Arithmetic Expressions

Operation	Symbol	Example
addition	+	x = y + z;
subtraction	-	x = y - z;
multiplication	*	x = y * z
division	/	x = y / z
exponentiation	**	x = y ** z

SAS Numeric Expressions

Numeric Expressions (cont.)

Exercise

- Calculate the mean arterial pressure (MAP)
- $MAP = \frac{Systolic\ Blood\ Pressure + 2 * Diastolic\ Blood\ Pressure}{3}$
- Calculate the number of years the subject has been in the study
  - Use Admission Date and Date of Last Follow Up

Numeric Expressions (cont.)

```
/*Numeric Expressions*/
data numeric;
    set data;

    MAP = (sysbp + 2 * diasbp) / 3;
run;
```

Numeric Functions

SAS Function	Description
ABS(x)	|x|
EXP(x)	e^x
FACT(x)	x!
LOG(x) / **LOG10**(x)	ln(x) / log_10(x)
ROUND(x,unit)	Round x to nearest multiple of unit
Int(x)	Round x to nearest integer
SQRT(x)	$\sqrt{x}$
MAX(arg-1,arg-2,...)	Maximum value of arguments
MIN(arg-1,arg-2,...)	Minimum value of arguments
STD(arg-1,arg-2,...)	Standard Deviation of arguments
MEAN(arg-1,arg-2,...)	Mean of arguments
SUM(arg-1,arg-2,...)	Sum of arguments
N(arg-1,arg-2,...)	Counts the number of arguments

Numeric Functions (cont.)

Exercise

- Calculate the number of symptoms as **symptoms_count**
- Calculate the sum of symptoms as **symptoms_sum**
- Calculate the mean of symptoms as **symptoms_mean**
- Calculate ln(LOS) and round to the nearest tenth as **ln_los**

- **Symptoms:**
  - **Atrial Fibriliation: afb**
  - **Cardiogenic Shock: sho**
  - **Congestive Heart Complications: chf**
  - **Complete heart Block: av3**

If-Then/Else Statements

- Logical SAS expression
- **Then** : Statement that is executed if the expression is true
- **Else** : Statement that is executed if the expression is false
- **If - Then/Else** : Statements are nested to produce a series of evaluations
  - Stops once a true statement is encountered
  - More computationally efficient than repeated if statements

If-Then/Else Statements

- Exercise:
  - Create variable **BMI_over** with two categories using "If/Else":
    - BMI greater than 25
    - BMI less than 25
  - Create variable **BMI_class** with four categories using "If/Else if":
    - BMI less than 18.5
    - BMI between 18.5 and 25
    - BMI between 25 and 30
    - BMI over 30

Do…End Statements

- The **DO** Statement specifies a group of statements to be executed as a unit.

```
/*Do...End statement*/
data DoEnd;
    set numeric;

    if OSCen = 0 then do;
        current_age = years + age;
    end;
    else do;
        deceased_age = years + age;
    end;
run;
```

Working with Character Variables

- 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
- Helpful when merging datasets
  - Variables need to be the same length
- Syntax: **LENGTH** variable-1 <**$**>length … variable-n <**$**>length;
  - **$**: Specifies the variable is a character type
  - length**:** number of character s

Working with Character Variables (cont.)

Using the **LENGTH** Statement

level
Moderate
Moderate
Moderate
Low
Low
Moderate
Low
High
High

Without the **LENGTH** Statement

level
Mod
Mod
Mod
Low
Low
Mod
Low
Hig
Hig

Working with Character Variables (cont.)

- **Exercise:**
- Create 3 different levels for **Symptoms**
  - None
  - One
  - Two or more

Character Functions

SAS Function	Description
Upcase(str) /**Lowercase**(str)	Convert to uppercase/ lowercase
Length(str)	Length excluding trailing blanks
Lengthc(str)	Length including trailing blanks
Strip(str)	Removes leading and trailing blanks
Tranwrd(str, target, replacement)	Replaces all occurrences
Catx(delimiter, srt-1, str-2, …)	Concatenates variables separated by delimiter
Index(source,excerpt)	Searches sources for excerpt and returns position number
Scan(str,count)	Returns the nth character from string
Substr(str,position,n)	Returns n number of characters from str starting at position

SAS Character Functions

Character Functions (cont.)

- Exercise
- Change all titles from “Mrs.” to “Ms.”
- Create variable **Name** with format “Last, First”
  - Hint: delimiters need “” around them
- Create variable **AreaCode** from **TelephoneNumber**
- Create variable **Address** with format “StreetAddress, City, State”
- Create variable **State** extracted from **Address**
- Create an all capitalized variable **City**

Date Variables

SAS Function	Description
DATE()/**TODAY**()	Returns todays date
DAY(x)	Returns day of month of date variable
MONTH(x)	Returns month of date variable
YEAR(x)	Returns year of date variable
WEEK(x)	Returns week of date variable
DATEPART(x)	Pulls date from Date-Time variable
TIMEPART(x)	Pulls time from Date-Time variable
‘DDMMMYYYY’**d**	Creates date variable

SAS Date Functions

Date Variables (cont.)

- Formatting Date Variables
- Internal SAS dates are just a string of numbers.
  - 0 = 1Jan1960
  - Each day after 1Jan1960 adds 1
  - Each day before 1Jan1960 subtracts 1
- Formatting makes it easier to read
  - Use **PUT**() function to create new formatted variables

SAS Format	Description
Date9.	DDMMYYYY
Datetime.	DDMMYYYY:HH:MM:SS
Time.	HHH:MM:SS

SAS Date and Time Formats

Date Variables (cont.)

- **Exercise:**
- Find today’s date two different ways
- Find the current year
- Create **HOUR, MINUTE,** and **Second** variables
- Use variables created above to create a date-time variable, **DTTIME**
- Find **TIME** using **DTTIME**
- Format **DTTIME**, **TIME**, **TODAY**
- If the subject is still alive
  - calculate when they should have their next visit
    - 60 days from last visit
  - Calculate the number of days between **LastContact** and **TODAY**
  - Hint: Try using a **DO** loop!

Variable Labels

- Using a **LABEL** statement in SAS will permanently associate a label with a variable
- Syntax: **LABEL** variable-1=label-1 … variable-n=label-n;

Variable Labels (cont.)

```
data labels;
    set ssi.data;

    Label DOB = 'Date of Birth' AgeAtStart = 'Age at Start of Study';
run;
```

Before Label

age	sho
70	0
88	0
73	0

After Label

Age at Start of Study	Cardiogenic Shock
70	0
88	0
73	0

Sub-Setting Data

- **Delete** Statement
  - Use the delete statement to exclude subjects with certain observations
  - e.g. delete those with a missing value
- **Output** Statement
  - The output statement only includes observations that meet the criteria specified
    - e.g. only include females
    - You can also use the output statement to create individual datasets

```
data subset_delete;
    set data;

/*  if OSTime = . then delete;*/
    if missing(OSTime) then delete;
run;
```

```
data subset_output;
    set data;

    if gender = 1 then output;
run;
```

Sub-Setting Data (cont.)

DROP/KEEP Statement

- Use the **DROP** statement to get rid of unwanted variables
- Use the **KEEP** statement to keep only variables you want
- If you have a set of variables with same pre-fix that are numbered differently you can lump them all together using ":" rather than type all out separately

```
data subset_keep;
    set subset_output;
    vitals1 = hr;
    vitals2 = sysbp;
    vitals3 = diasbp;
    vitals4 = bmi;

    keep ID Occupation vitals:;
run;
```

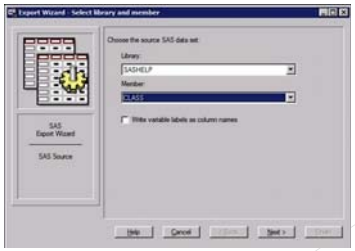
Sub-Setting Data (cont.)

- **Exercise:**
- Create 4 datasets, Subset1-Subset4
  - Subset1 with Blood Type O+ and O-
  - Subset2 with Blood Type A+
  - Subset3 with Blood Type B+
  - Subjset4 with Blood Type AB+

Exporting Data

Using the Export Wizard

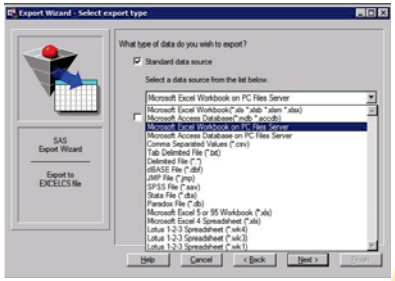
- Select **File » Export Data**
- The window dialog box will appear
- Select your SAS library libref
  - Library: **WORK**
- Select the **member** of the library
  - Member: **Export**
- Click **Next**



Exporting Data (cont.)

Export Wizard (cont.)

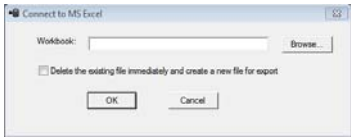
- The **Select Export Type** dialog box will appear
- Select what data source you would like to export to
- Most common are *.xlsx, *.csv, and *.txt
  - We will export ***.xlsx**



Exporting Data (cont.)

Export Wizard (cont.)

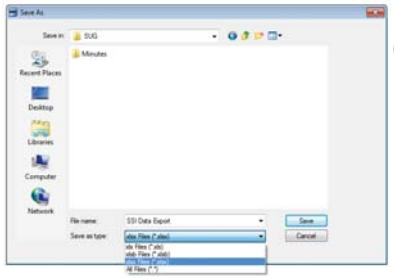
- Click the **Browse** button to point to a folder where you want the data saved
  - e.g. save to your **H:\ drive**

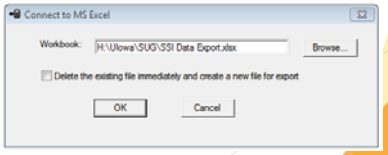


Exporting Data (cont.)

Export Wizard (cont.)

- Enter **File Name**
- Be sure to scroll down to **.xlsx file**
  - Default is old version of Microsoft office
- Click **Save**
- Then click **OK**

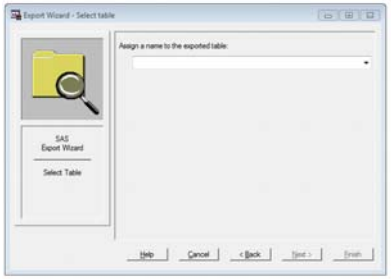




Exporting Data (cont.)

Export Wizard (cont.)

- Assign a name to the exported table
  - This names the sheet in excel
- You can be done here and click **Finish**
- If you want to save the code click **Next**
- Assign a file directory where it will be saved and click **Finish**
  - It will save a file with code for **Proc Export**



Exporting Data (cont.)

Using Proc Export

DBMS= data-source-identifier

DBMS= specifies the type of external data source the EXPORT procedure creates. To export to a DBMS table, specify DBMS= using a supported database identifier. For example, DBMS=ACCESS specifies a Microsoft Access 2000, 2002, 2003, or 2007 database.

Note: Transcoding is not supported for DBMS=XLS. The output yields unpredictable results. Use DBMS=EXCEL as an alternative.

Data Source Identifier Summary

Data Source Identifier	Output Data Source	File Extension
ACCESS	Microsoft Access 2000, 2002, 2003, 2007, or 2010 table (using the LIBNAME statement)	.mdb .accdb
ACCESSCS	Microsoft Access table connecting remotely through PC Files Server	.mdb .accdb
CSV	delimited file (comma-separated values)	.csv
DBF	dBASE 5.0, IV, III+, and III files	.dbf
DBFMEMO	dBASE 5.0, IV, III+, and III files with memos FoxPro and VisualPro with memos	.dbf .fpt
DLM	delimited file (default delimiter is a blank)	
DTA	Stata file	.dta
EXCEL	Excel 97, 2000, 2002, 2003, 2007 or 2010 workbook (using the LIBNAME statement)	.xls .xlsb .xlsx
EXCEL4	Excel 4.0 workbook (using PROC DBLOAD)	.xls
EXCEL5	Excel 5.0 or 7.0 (95) workbook (using PROC DBLOAD)	.xls
EXCELCS	Excel workbook connecting remotely through PC Files Server	.xls .xlsb
JMP	JMP files	.jmp
PARADOX	Paradox DB files	.db
PCFS	JMP files, Stata files, and SPSS files connecting remotely through PC Files Server	.jmp, .dta, .sav
SAV	SPSS files, compressed and uncompressed binary files	.sav
TAB	delimited file (tab-delimited values)	.txt
WK1	Lotus 1-2-3 Release 2 spreadsheet	.wk1
WK3	Lotus 1-2-3 Release 3 spreadsheet	.wk3
WK4	Lotus 1-2-3 releases 4 and 5 spreadsheet	.wk4
XLS	Excel 97, 2000, 2002, or 2003 spreadsheet (using file formats) **Note:** Transcoding is not supported for DBMS=XLS. The output yields unpredictable results. Use DBMS=EXCEL as an alternative.	.xls

Exporting Data (cont.)

Using Proc Export (cont.)

```
proc export data=subset_keep
    outfile = 'H:\SSI Data.xlsx'
    dbms=xlsx
    replace;
run;
```

Using proc Export to subset data

```
proc export data=data (where=(gender=1))
    outfile='H:\Femalelist.csv'
    dbms=csv
    replace;
run;
```

SAS Proc Export

Exporting Data (cont.)

Using ODS

- SAS Output Delivery System
- Gives flexibility in generating reports
- Great for printing graphs or plots

```
ods pdf file="H:\odsoutput.pdf";
proc print data = subset_keep;
    where Age < 60;
run;
ods pdf close;
```

SAS ODS output Tip Sheet