
11


SHARK
Patrol

Introduction to SAS

Part 1

Goals

Be able to

- Write a short SAS program
- Run the program and look at the results
- Check for errors in the Log window
- Save the program
- Save the data set the program created
- Access the saved data set
- Import data from Excel into SAS
- Add comments to a SAS program

First steps

- Create a folder for this class on your computer (e.g. ERHS642)
- Open SAS
- We will now write a short SAS program, run it and look at the results

Type test program in Editor window

```
data test;
  input lung_cancer smoking;
cards;
1 1
0 0
1 0
run;

proc print data=test;
run;
```

In a data step you create a SAS data set

- This is the data set
- It contains lung cancer and smoking data for 3 subjects

- In a procedure (proc) step, you tell SAS what to do with the data set
- E.g., proc print tells SAS to list the data in data set “test”

Click on running guy

- In the Results Viewer, you will see…

Obs	lung_cancer	smoking
1	1	1
2	0	0
3	1	0

Select Log tab (bottom of screen)

```
28   data test;
29     input lung_cancer smoking;
30   cards;
NOTE: The data set WORK.TEST has 3 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds
34   run;
36   proc print data=test;
37     var lung_cancer smoking;
38   run;
NOTE: There were 3 observations read from the data set WORK.TEST.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.14 seconds
      cpu time           0.01 seconds
```

No error

Clear Log window…

- From Edit pull down menu, select Clear All

Errors in SAS programs

Now we'll introduce an error into the program and see what happens

Click on the Editor tab to return to the program…

Delete first semicolon

data test ← missing semicolon = error
  input lung_cancer smoking;
cards;
1 1
0 0
1 0
run;

proc print data=test;
run;

Click on running guy

- No new results in Results Viewer
- Select Log tab to check for errors

Find error in Log window…

```
39   data test
40     input lung_cancer smoking;
41   cards;
NOTE: The data set WORK.TEST has 1 observations and 0 variables.
NOTE: The data set WORK.INPUT has 1 observations and 0 variables.
NOTE: The data set WORK.LUNG_CANCER has 1 observations and 0 variables.
NOTE: The data set WORK.SMOKING has 1 observations and 0 variables.
NOTE: DATA statement used (Total process time):
45   run;
47   proc print data=test;
48     var lung_cancer smoking;
ERROR: Variable LUNG_CANCER not found.
ERROR: Variable SMOKING not found.
49   run;
NOTE: The SAS System stopped processing this step because of errors.
```

Saving the SAS program (i.e. the SAS code)

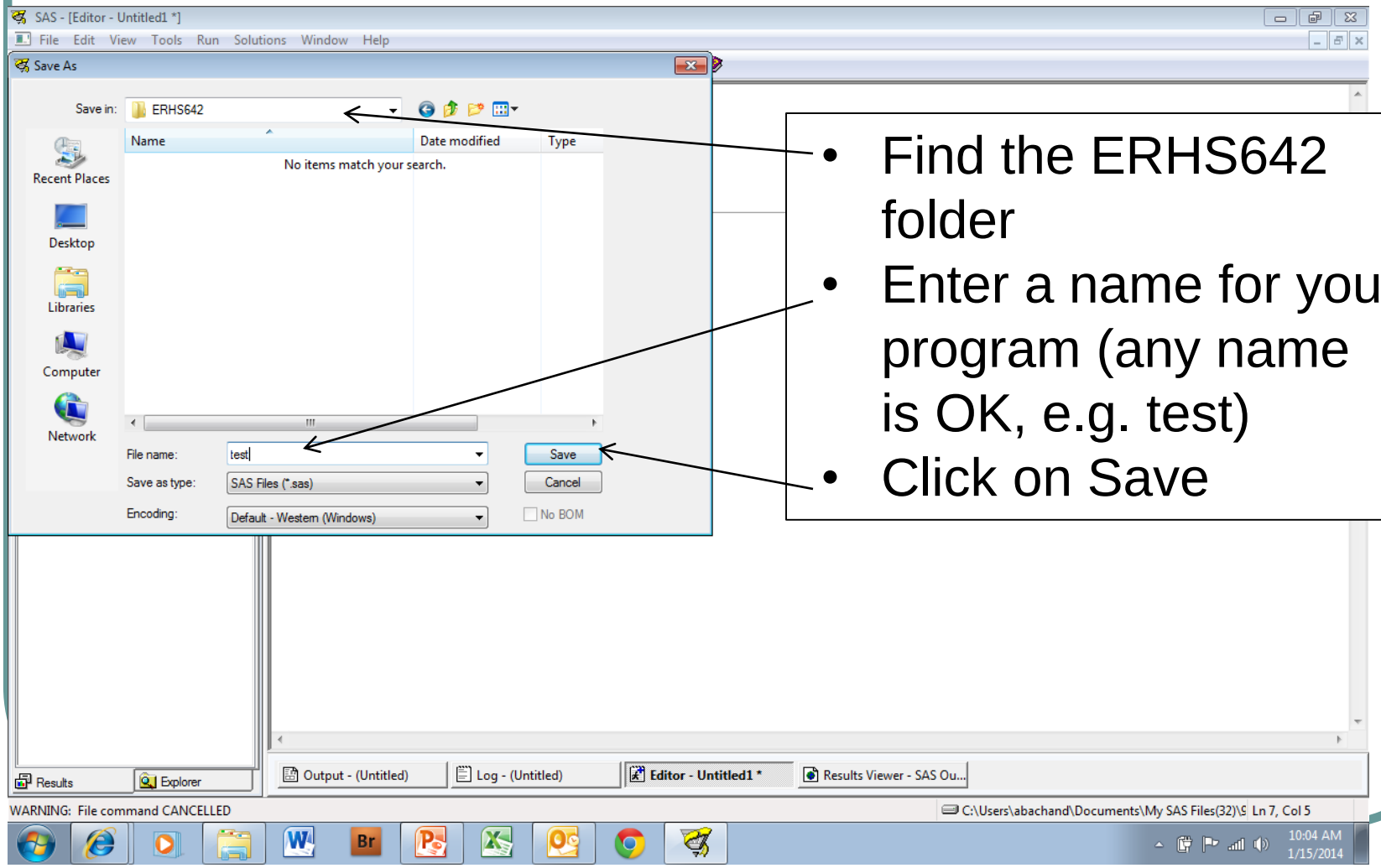
To continue,

- Clear the Log window (Edit → Clear All)
- Fix the error (add the semicolon)
- Save the program →
  - Click on Save symbol (disk)

```
data test;
  input lung_cancer smoking;
cards;
1 1
0 0
1 0
run;

proc print data=test;
run;
```

Saving the SAS program



- Find the ERHS642 folder
- Enter a name for your program (any name is OK, e.g. test)
- Click on Save

Check if the program was saved

- On your computer, check the ERHS642 folder
- It should contain a file called “test.sas”
- This file contains the entire SAS code from the Editor window

Opening the saved program

Next we'll open the saved program in SAS

- Close SAS
- Open SAS again
- Open the program
  - Click Open symbol (open folder)
  - Find ERHS642 folder
  - Double click program name

  The program appears in the Editor window

Permanently saving the data set

Lung_cancer	smoking
1	1
0	0
1	0

Now we'll save the test data set

- Why?
  - After running program test.sas, data set "test" (in SAS format) is stored in a temporary SAS "library"
  - If you close SAS, the data set is lost
  - Note: A SAS library is a place where SAS stores data
- Note:
  - We can recreate the data set by running the SAS program again
  - However, saving a permanent copy of the original data is advisable in case you make any changes

Permanently saving the data set

- Designate the ERHS642 folder on your computer as a permanent SAS library
  - Recall that a library is a place where SAS stores data
- Save the data set permanently in this library

Saving the data set: lib name statement

- Add a libname statement and expand the data set name
- Run the new program (see next slide)
- Data set “test” is now permanently stored in the C:\ERHS642 folder

```
libname sdat 'C:\ERHS642';

data sdat.test;
input lung_cancer smoking;
cards;
1 1
0 0
1 0
run;
```

- This libname statement tells SAS to use the ERHS642 folder as a permanent library called “sdat”
- Any name is OK; sdat is just an example
- You can only assign a SAS library name to an existing folder

Saving the data set: Running the program

```
libname sdat 'C:\ERHS642';

data sdat.test;
input lung_cancer smoking;
cards;
1 1
0 0
1 0
run;
```

- Highlight the libname statement and the data step
- Click on the running guy
- Only the highlighted portion of the program is executed

From Log

NOTE: Libref SDAT was successfully assigned as follows:
Engine: V9
Physical Name: C:\ERHS642
NOTE: The data set SDAT.TEST has 3 observations and 2 variables.

Check if the data set was saved

- On your computer, check the ERHS642 folder
- It should contain a file called “test.sas7bdat”
- This file contains the data set “test” in SAS format

Accessing the saved data set

- Data set “test” can be accessed from SAS using the SAS library name

To try it,

- Close SAS
- Open SAS

Accessing the saved data set

- Enter the SAS code below
- Note: **sdat.** was added to proc print; this tells SAS in which library to look for the data set
- Run the program

```
libname sdat 'C:\ERHS642';
proc print data=sdat.test;
run;
```

Obs	lung_cancer	smoking
1	1	1
2	0	0
3	1	0

proc print tells SAS to list the data from data set “test” which is stored in the library called “sdat” (i.e. in the ERHS642 folder)

So far, you have learned to…

- Create ERHS642 folder; open SAS
- Type program (i.e. SAS code) in Editor window
  - Program contains data (in the data step) and instructions what to do with the data (in the procedure step; here, proc print)
- Save program (i.e. SAS code) in ERHS642 folder by clicking on the disk symbol
- Save data set on your computer by assigning the ERHS642 folder as a SAS library

Importing data from MS Excel

- Larger data sets can be imported from MS Excel (or other sources)

To try it,

- Save the data set “chdage.xlsx” in the ERHS642 folder
- Double click on the file to see the data in Excel
- Open SAS

Import the data

- From the File pull down menu, select Import data
- Screen 1: Excel is the default; click on Next (bottom of screen)
- Screen 2: Browse for the “chdage” data set

  Note: First, change the file extension at the bottom of the screen to xlsx
- Screen 3: Click on Next (ignore options and sheet for now)
- Screen 4: In the Member field, enter the name you want to give the data set; any name is fine; let’s choose chdage
- Screen 5: Click on Finish

Check for errors

1. Check the Log

- Look for the following message:

NOTE: WORK.CHDAGE data set was successfully created.

NOTE: The data set WORK.CHDAGE has 100 observations and 3 variables.

2. Write a short program to see the contents of the data set and the actual data

```
proc contents position data=chdage; run;

proc print data=chdage; run;
```

Step 2 results

Your Results Viewer should show

- The contents of the data set
  - Variables and variable type (Num=numeric) in alphabetical order and, in a second table, in the order in which they were entered in the Excel file
- A list of the data
  - 100 observations (rows)
  - 3 variables, id, age and chd (columns)

Saving the new program code

- Click on Save symbol (disk)
- Find the ERHS642 folder
- Type program name (any name you like, e.g. chdage)
- Click on Save
- In the ERHS642 folder, find chdage.sas

Permanently saving the new data set

- Add a libname statement and data step to the SAS program; save new code (click on disk), run and check Log window

  libname sdat 'C:\ERHS642';

  data sdat.chdage; set chdage; run;

- *data sdat.chdage* instructs SAS to permanently save a data set in the sdat library (i.e. the ERHS642 folder) and name it chdage
- *set chdage* instructs SAS that this permanent data set is the chdage data set (currently stored in the temporary SAS library)

Accessing the saved data set

- Save the new program code (click on the disk)
- Close SAS; open SAS
- The chdage data set is no longer stored in the temporary SAS library
- Open your program (chdage.sas) (click on the open folder)
- Move libname statement to the top
- Block out remaining SAS code using /* and */
  - SAS code between /* and */ is ignored when the program is run

Accessing the saved data set

libname sdat 'C:\ERHS642';

/

proc contents position data=chdage; run;

proc print data=chdage; run;

data sdat.chdage; set chdage; run;

****

- Add a new data step and run the program

data chdage; set sdat.chdage; run;

Accessing the saved data set

- *data chdage* instructs SAS to save a data set in the temporary SAS library and name it chdage
- *set sdat.chdage* instructs SAS that this temporary data set is the chdage data set (currently stored in the permanent SAS library sdat (i.e. the ERHS642 folder)

- Why are we creating a temporary data set when we already have a permanent version?

Temporary vs. permanent data

- Creating a temporary data set from a permanent data set is optional
- Instead, you can just add sdat to the procedure step (as we did before)

  proc print data=**sdat.**chdage; run;

- However, you will likely manipulate the data (e.g. categorize a continuous variable)
- It is safer to manipulate data in the temporary data set than to change the original data set
- You can always save the manipulated temporary data set as a second permanent data set

Example of data manipulation

```
data chdage; set sdat.chdage;
        if 0 <=age<=50 then age_cat=0;
  else if 50< age<=70 then age_cat=1;
run;

proc print data=chdage; run;
```

- You created a new variable called age_cat which equals 0 for ages 50 and below and equals 1 for ages >50 (69 is the maximum age in this data set)
- proc print lists the data for the three original variables (id, age, chd) and the new variable (age_cat)

Adding comments to a SAS program

- Text between **/*** and ***/** is ignored when the model is run
- Text between ***** and **;** is ignored when the model is run
- This can be used to block out parts of the program
- It can also be used to add comments and descriptions

chdage program without comments

Our chdage program so far:

```
libname sdat 'C:\ERHS642';
/*
proc contents position data=chdage; run;
proc print data=chdage; run;
data sdat.chdage; set chdage; run;
*/
data chdage; set sdat.chdage;
          if 0 <=age<=50 then age_cat=0;
  else if 50< age<=70 then age_cat=1;
run;
proc print data=chdage; run;
```

chdage program with comments

```
libname sdat 'C:\ERHS642';
**************************************************************;
* chdage.sas                                                 *;
* Import data from Excel                                     *;
* Save original data as permanent SAS data set               *;
* Create age categories  in temporary data set               *;
* ***********************************************************;
/*
* Check for import errors, save as permanent data set;
proc contents position data=chdage; run;
proc print data=chdage; run;
data sdat.chdage; set chdage; run;
*/
```

chdage program with comments continued

```
* Create age categories in temporary data set ;
data chdage; set sdat.chdage;
        if 0 <=age<=50 then age_cat=0;
  else if 50< age<=70 then age_cat=1;
run;

proc print data=chdage; run;
```

SUMMARY

Creating/saving SAS data

- (Create ERHS642 folder; open SAS)
- Import data set from MS Excel
  - Import data; check for errors
  - Save data set as permanent SAS data set
  - Note: The data set doesn’t have to be imported from Excel again
- Or enter data in SAS program
  - Write code to enter data directly in SAS
  - Save data as permanent SAS data set

Using saved SAS data

- Create a temporary data set from the permanent data set
  - You can manipulate the data in the temporary data set
  - If you like, you can save the new temporary data set as a second permanent data set
- Or use the permanent data set directly in a procedure step (e.g., proc print)

chdage example - current program

```
libname sdat 'C:\ERHS642';
/*
proc contents position data=chdage; run;
proc print data=chdage; run;
data sdat.chdage; set chdage; run;
*/
data chdage; set sdat.chdage;
        if 0 <=age<=50 then age_cat=0;
  else if 50< age<=70 then age_cat=1;
run;
proc print data=chdage; run;
```

- Create temporary data set from permanent data set
- Manipulate data
- List data from temporary data set

chdage example – alternative program 1

```
libname sdat 'C:\ERHS642';
/*
proc contents position data=chdage; run;
proc print data=chdage; run;
data sdat.chdage; set chdage; run;
*/
proc print data=sdat.chdage; run;
```

- List data from permanent data set

chdage example - alternative program 2

```
libname sdat 'C:\ERHS642';
/* proc contents position data=chdage; run;
proc print data=chdage; run;
data sdat.chdage; set chdage; run; */

data chdage; set sdat.chdage;
      if 0 <=age<=50 then age_cat=0;
  else if 50< age<=70 then age_cat=1;
run;
proc print data=chdage; run;

data sdat.chdage2; set chdage; run;
```

- Create temporary data set from permanent data set
- Manipulate data
- List data from temporary data set
- Save temporary data set with new variable as second permanent data set

test example – current program

```
libname sdat 'C:\ERHS642';
data sdat.test;
input lung_cancer smoking;
cards;
1 1
0 0
1 0
run;
proc print data=sdat.test; run;
```

- Create permanent data set
- List data from permanent data set

test example – alternative program 1

```
libname sdat 'C:\ERHS642';
data sdat.test;
input lung_cancer smoking;
cards;
1 1
0 0
1 0
run;
data test; set sdat.test;
  if lung_cancer=1 and smoking=1 then both=1;
  else both=0;
run;
```

- Create permanent data set
- Create temporary data set and manipulate data

test example – alternative program 1 continued

```
proc print data=test; run;
data sdat.test2; set test; run;
```

- List data from temporary data set
- Save temporary data set as second permanent data set