Introduction to SAS

Dr. Lendie Follett

2022-03-22

Getting access

We will be using SAS Studio. This is a web-based interface that connects to SAS on a server. Since it runs in a web-browser, you don't need to download any program. However, you will need to sign up, as a student, for free access.

This is your first assignment. It is due PRIOR TO CLASS ON WEDNESDAY. See instructions on the next slide:

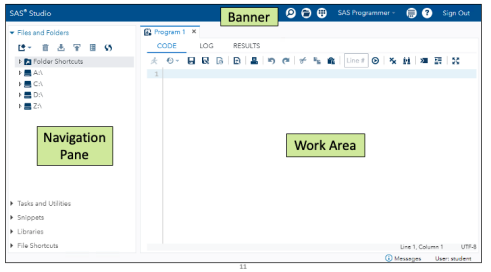
Instructions

1. Visit https://odamid.oda.sas.com and click on Register for an account.
2. After you have successfully created your account, follow these steps:
    - Sign on the the Control Center at https://odamid.oda.sas.com}
    - Look for the Enroll in a course link in the "Enrollments" section near the bottom of the page. Click this link to start the enrollment.
    - Enter the course code: e074b22e-b478-4023-8a4c-92a48701c27a
    - Submit the form.
    - Confirm that this is the correct course and then click the button to finish enrolling.

Summary of details you may need:

- Course Name: Stat 40
- Level: Undergraduate
- Institution: Drake University
- Course Code: e074b22e-b478-4023-8a4c-92a48701c27a
- Software application we will use: SAS Studio

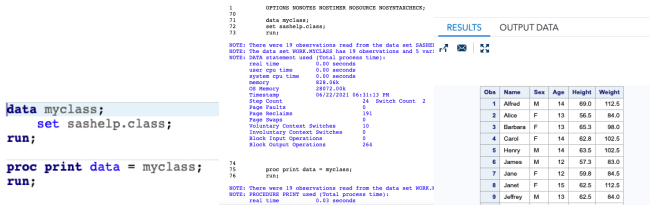
Getting Started



SAS Studio interface

SAS Interface

There are three basic components you will interact with in SAS:



Three basic components of programming in SAS. From left to right: Editor, Log, Results and Output

Case Study

Goals:

1. Navigate SAS Studio interface.
2. Locate pre-loaded data.
3. Write program.
4. Execute code 2 different ways.
5. Examine log and results viewer.

Steps

- A SAS program consists primarily of a series of *steps*
- A step can be one of two kinds: a data step or a proc step.

```
data myclass;
set sashelp.class;
run;

proc print data = myclass;
run;
```

- A step generally ends with a run statement
- A SAS program can contain any number (0,1,2,...) of data steps and proc steps
- The structure will depend on specific goal or task

Data Step

- The goal of a *data step* in a SAS program is typically to read, process, or create data.
- Definition: A *SAS data set* is a data set that SAS can understand and that can only be created by SAS
- When creating a *SAS data set*, the data step can perform various tasks such as producing new columns/variables.

Proc Step

- The goal of a *proc step* is typically to report, manage, or analyze data.
- Thus, a SAS data set is an input to a *proc step*.
- There are many unique *proc steps* which accomplish a wide variety of processing tasks. For example,
  - View a SAS data set
  - Sort a SAS data set
  - Summarise features of a SAS data set (means, sums, counts, etc...)
  - Produce graphics

More on steps

- A step is made up of *statements*, which are appropriate for that specific step.

```
title 'My ultra-thorough analysis';

data myclass;
    set sashelp.class;
    heightcm=height*2.54;
run;

proc print data=myclass;
run;

proc means data=myclass;
    var age heightcm;
run;
```


Statements

- All statements end with a semicolon!
- A SAS program can also contain *global* statements, which appear outside of steps. They don't need a run statement.

```
title 'My ultra-thorough analysis';

data myclass;
    set sashelp.class;
    heightcm=height*2.54;
run;

proc print data=myclass;
run;

proc means data=myclass;
    var age heightcm;
run;
```


Your Turn

1. View the code. How many steps are in the program?
2. How many statements are in the PROC PRINT step?

```
data mycars;
    set sashelp.cars;
    AvgMPG=mean(mpg_city, mpg_highway);
run;

title "Cars with Average MPG Over 35";
proc print data=mycars;
    var make model type avgmpg;
    where AvgMPG > 35;
run;

title "Average MPG by Car Type";
proc means data=mycars mean min max maxdec=1;
    var avgmpg;
    class type;
run;

title;
```

Your Turn

1. How many global statements are in the program?

```
data mycars;
    set sashelp.cars;
    AvgMPG=mean(mpg_city, mpg_highway);
run;

title "Cars with Average MPG Over 35";
proc print data=mycars;
    var make model type avgmpg;
    where AvgMPG > 35;
run;

title "Average MPG by Car Type";
proc means data=mycars mean min max maxdec=1;
    var avgmpg;
    class type;
run;

title;
```


Your Turn

1. Run the program and view the log.
2. How many observations were read by the PROC PRINT step?

```
data mycars;
    set sashelp.cars;
    AvgMPG=mean(mpg_city, mpg_highway);
run;

title "Cars with Average MPG Over 35";
proc print data=mycars;
    var make model type avgmpg;
    where AvgMPG > 35;
run;

title "Average MPG by Car Type";
proc means data=mycars mean min max maxdec=1;
    var avgmpg;
    class type;
run;

title;
```


Syntax - Format

```
data myclass;
    set sashelp.class;
run;

proc print data=myclass;
run;
```

```
data myclass;set sashelp.class;run;
proc print data=myclass;run;
```

Formatting makes your code easier to read and understand and, thus, easier to de-bug.

Syntax - cAsE

```
data under13;
    set sashelp.class;
    where AGE<13;
    drop heIGht Weight;
run;
```

```
DATA UNDER13;
    SET SASHELP.CLASS;
    WHERE AGE<13;
    DROP HEIGHT WEIGHT;
RUN;
```

Unquoted values can be in any case.

Syntax - Comments

- Adding comments to your code makes it more understandable to other people (and your future self!)
- If you "comment something out", SAS will ignore it when you execute the program
- Useful when testing code because you can suppress a portion of the code from execution
- There are two ways to comment in SAS
  - By putting text between "/*" and "*/"
  - By starting the statement you want to omit with a "*"

Syntax - Comments

```
/* create new SAS data set containing only
students under 13 yo */

data under13;
set sashelp.class;
where Age<13;
*drop Height Weight;
run;
```

Those three lines will not execute - SAS will ignore them.

Errors in SAS

- Misspelled keywords
- Unmatched quotation marks
- Missing semicolon
- Invalid option

All of the above are common sources of syntax errors. We all need to get comfortable with errors as they are a fact of (a programmer's) life.

I get errors every day I program, regardless of language. True story.

You can catch some errors by paying attention to SAS's color-coded syntax, or, if the code is already run, by diligently examining the log after each execution.

Your Turn

Paste this code into a new SAS editor.

```
data canadashoes; set sashelp.shoes;
    where region="Canada;
    Profit=Sales-Returns;run;

prc print data=canadashoes;run;
```

1. Format the program to make it easier to read. What syntax error is detected? Fix the error and run the program.
2. Read the log and identify any additional syntax errors or warnings.

Your Turn

1. Add a comment to describe the changes that you made to the program.
2. Run the program and examine the log and results. How many rows are in the canadashoes data?