Reading in External Data, Formatting and Manipulating Data Sets

STAT 3505

Week 3 (February 1, 2024)

Gunes Fleming Ph.D.

Remember from Previous Class

What SAS step is typically used to generate simple descriptive summaries and graphs?

Remember from Previous Class

What SAS step is typically used to generate simple descriptive summaries and graphs?

PROC

PROC steps are typically used to process SAS data sets (that is, generate reports, graphs, and statistics).

Remember from Previous Class

Does this comment contain syntax errors?

```
  /*
Report created for budget
presentation; revised October 15.
  */
proc print data=work.newloan;

run;
```

Remember from Previous Class

Does this comment contain syntax errors?

```
  /*
Report created for budget
presentation; revised October 15.
  */
proc print data=work.newloan;

run;
```

- No. The comment is correctly specified

A block comment can contain semicolons and unbalanced quotation marks, can appear anywhere, and doesn't need a semicolon at the end.

Remember from Previous Class

What happens if you submit the following program?

```
porc print data=work.newsalesemps;

run;
```

Remember from Previous Class

What happens if you submit the following program?

```
porc print data=work.newsalesemps;

run;
```

- SAS assumes that the keyword PROC is misspelled and executes the PROC PRINT step

The log will indicate that SAS assumed that the keyword PROC was misspelled, corrected it temporarily, and executed the PROC step.

Remember from Previous Class

What result would you expect from submitting this step/ Do you see any problems with the proc step below?

```
proc print data=work.newsalesemps
run;
```

Remember from Previous Class

What result would you expect from submitting this step/ Do you see any problems with the proc step below?

```
proc print data=work.newsalesemps
run;
```

- an error message in the log

There is a missing semicolon following the data set name. When this step runs, SAS will interpret the word RUN as an option in the PROC PRINT statement (because of the missing semicolon). As a result, the PROC PRINT step will not execute, and an error message will be displayed in the log.

Remember from Previous Class

Temporary Vs. Permanent Data Sets:

- A temporary SAS data set is named with a single level name such as MEASLES, or MAR2000. (technically the names are WORK.MEASLES and WORK.MAR2000)

- A permanent SAS data set is a file saved on your hard disk (Base SAS) or a folder in server.

Remember from Previous Class

Create a Library for this Class:

- libname STAT3505 '~/my_shared_file_links/u63742093';

- To create a library, in general:

  LIBNAME *yourlibraryname* "LOCATION";

Quick Quiz
(In Class Assignment 1)

- Timed: 10 minutes
- Single attempt
- A total of 6 questions
- Work on your own
- You can use your notes

INFILE STATEMENT

- Reads in external files from the computer/server.

- Data Step Syntax with INFILE statement:

```
DATA datasetname;
    INFILE filespecification options;
    INPUT variablelist;
RUN;
```

- INFILE statement frequently used with external dataset path name.:

INFILE "*EXTERNAL INPUT FILE PATH*";

INFILE STATEMENT

- When INFILE statement is used to read external data in, there is no need for the DATALINES statement.

- INFILE statement must appear before INPUT statement that reads data lines.

- There are a number of INFILE statement options available to help read data into SAS in the exact form that you intend to read.

INFILE STATEMENT

- Some INFILE statement options are:
  - DLM: allows you to define a delimiter to be something other than a blank
  - DSD: tells SAS to recognize two consecutive delimiters as a missing value
  - FIRSTOBS: tells SAS what line should it start reading the raw data from
  - OBS: indicates which line in your raw data file should be treated as the last record to be read by SAS

INFILE STATEMENT

- Example 1:

```
DATA import_cars_txt1;
  INFILE '~/my_shared_file_links/u63742093/cars_blank_txt.txt' FIRSTOBS=2;
  INPUT make $ type $ origin $ MPG_Highway MPG_City EngineSize;
RUN;
```

- Example 2:

```
DATA import_cars_txt2;
  INFILE '~/my_shared_file_links/u63742093/cars_comma_txt.txt' DLM=',' FIRSTOBS=2 OBS=26;
  INPUT make $ type $ origin $ MPG_Highway MPG_City EngineSize;
RUN;
```

PROC IMPORT

- Imports external files such as excel, CSV, text etc. into SAS data sets.

- Syntax:

```
PROC IMPORT
        DATAFILE = " EXTERNAL INPUT FILE PATH "
        OUT = LIBNAME.DATASETNAME
        DBMS = FILETYPE
        REPLACE;
        < OPTIONS >;
RUN;
```

- REPLACE is a commonly used PROC IMPORT option. Tells SAS to replace an existing dataset with the same NAME.
- DBMS
- (As an alternative importing tool, Import Wizard can be used in SAS Base).

PROC IMPORT

- Commonly used options:
  - **SHEET**: Specifies the Excel worksheet to import (for Excel files).
  - **GETNAMES**: Specifies that the first row of the input file contains variable names. (YES is default).
  - **DATAROW**: Specifies the starting row in the input file where data begins.
  - **GUESSINGROWS**: Specifies the number of rows to use when determining variable types and lengths. (Default is 20).
  - **DELIMITER**: Specifies the delimiter for delimited files (for DBMS=DLM).

PROC IMPORT - Example

- Import a CSV file:

```
proc import
     datafile = "~/my_shared_file_links/u63742093/hdp.csv"
     out = work.data1_hdpimport
     dbms = CSV
     replace;
run;
```

PROC IMPORT - Example

- Import a txt with a I delimiter file:

```
proc import
    datafile = "~/my_shared_file_links/u63742093/senators.txt"
    out = data2_senatorimport
    dbms = DLM  /* default delimiter is blank */
    replace;
    delimiter = "|"; * We must indicate the delimiter as it is different than the default;
run;
```

PROC IMPORT - Example

- Import an xlsx file:

```
proc import
     datafile = "~/my_shared_file_links/u63742093/Anamoly_Detection.xlsx"
     out = data3_AnDetimport
     dbms = XLSX
     replace;
     sheet = "Raw Data"; * Sheet name that we want to import;
run;
```

PROC EXPORT

- Exports SAS files as external files such as excel, CSV, text etc.

- Syntax:

```
PROC EXPORT
        DATA = LIBNAME.DATASETNAME
        OUTFILE = "EXTERNAL OUTPUT FILE PATH\OUTPUTFILENAME.EXTENSION"
        DBMS = FILETYPE
        REPLACE;
        < OPTIONS >;
RUN;
```

- REPLACE is a commonly used PROC EXPORT option. Tells SAS to replace an existing dataset with the same name.

PROC EXPORT - Example

- Export as an xlsx file:

```
proc export
     data = stat3505.salary
     outfile = "~/my_shared_file_links/u63742093/SalaryExported.xlsx"
     dbms = xlsx
     replace;
run;
```

PROC EXPORT - Example

- Export as a txt file with a comma delimiter:

```
proc export
     data = stat3505.salary
     outfile = "~/my_shared_file_links/u63742093/SalaryExported2.txt"
     dbms = TAB
     replace;
     delimiter = ',';
run;
```

- Note that DBMS=TAB indicates that we want to export the data as a text file, and the tab character will be used as a delimiter.

  Although it says "TAB," it's actually using a comma as the delimiter here.

FORMAT/ INFORMAT

- Informats – tell SAS how the data should be read in.
- Formats – tell SAS how to print/display data.

- All informats and formats end with a dot (.) or contain a dot (.)
- The number in SAS format/informat tells SAS the number of characters to read in

FORMAT/ INFORMAT

- A FORMAT statement within the DATA step is used to tell SAS how data will be presented in output.

- The FORMAT statement, which appears in the DATA step, uses the following syntax:

  FORMAT VAR1 FORMAT1. VAR2 VAR3 VAR4 FORMAT2.etc... .;

- Example:

  FORMAT BMI 5.2 name $10.;

FORMAT/ INFORMAT

- Example:

```
DATA data4;
  INPUT @1 NAME $11. @12 BDATE DATE9.;
  FORMAT BDATE WORDDATE12.; * Assigns a format to display BDATE;
DATALINES;
Bill        08JAN1952
Jane        02FEB1953
Clyde       23MAR1949
;
proc print data=data4;
run;
```

- Date/time formats

PROC FORMAT

- So far, we have only used pre-defined SAS formats.
- PROC FORMAT allows creating custom formats.
- These custom formats allow you to specify the information to be displayed for selected values of a variable.
- Once a FORMAT has been created, it must then be applied within a SAS procedure to take effect.
- The steps for using formatted values are:
    - Create a FORMAT definition in PROC FORMAT.
    - Apply the FORMAT to one or more variables in a DATA or PROC statement.

PROC FORMAT

- A user-generated format name should:
  - A numeric format name can be up to 32 characters in length.
  - A character format name can be up to 31 characters in length.
  - The name can include letters, numbers, and underscores.
  - Begin with a letter or an underscore and should not end with a number.
  - Not be same as SAS reserved words or keywords such as FORMAT, PROC, etc.
  - Not have special characters.

Modify/Manipulate Data in Data Step (See Examples in SAS)

- Some statements that we can and will use in the examples are:
  - KEEP
  - DROP
  - LABEL
  - RENAME
  - Using WHERE or IF for subsetting data.