Data and SAS

Dr. Lendie Follett

2022-03-23

Data

- Accessing data is the first step in any SAS programming project.
- We will be working with *structured data*, which can easily be read into SAS.
- Definition: Structured data includes defined rows and columns. You can store structured data in SAS. Oracle, Excel, Hadoop, etc…

.	.	.	.
.	.	.	.
.	.	.	.

Anatomy of a SAS data set

- Once data is read into SAS, it is called a *SAS data set.*
- There are two parts to any SAS data set:

1. a descriptor portion and
2. a data portion.

Data Portion

If you want to see the data portion, you can use a proc print step.

```
proc print data = sashelp.iris;
run;
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

Descriptor portion:

"Metadata"

- name of SAS data set
- number of rows
- date created
- column names
- column attributes

Column Attributes

Wait a second, what is a "*column attribute*"?

- This is something all columns in SAS must have.
- There are three components.
    1. Name
    2. Type
    3. Length

1. Name:

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

- Must be 1-32 characters, starts with a letter (a-Z,A-Z) or underscore, only contains: (letters, numbers, underscores), can be uppercase or lowercase or mixed
- Examples of valid names: stat40, mylib
- Examples of invalid names: 40stat, stat.40

2. Type:

Column type is either (1) numeric or (2) character.

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

Columns of type numeric contain digits 0-9, minus sign, decimal points, or scientific notation (E). Dates are also understood as numeric! More on this later...

Character variables can contain letters, numbers, special characters, and blanks.

3. Length

- Storage (in bytes) allocated to store column values.
- This is related to the type.
- Numeric columns, by default, area always stored as 8 bytes (enough for ~ 16 digits).
- Character can be any length (practically). The length will be set according to how many characters are present in the column and will assume the length of the longest string.

So, you want to see the descriptor portion?

If you want to see the descriptor portion (which, if you recall, includes the column attributes), you can use a proc contents step. This is a new one!

```
proc contents data = sashelp.shoes;
run;
```

Case Study

sashelp.heart contains data from the Framingham heart study. Let's identify the data portion and the descriptor portion, paying special attention to the column attributes.

```
proc print data = sashelp.heart;
run;

proc contents data = sashelp.heart;
run;
```

Your Turn

In the heart data, how are missing character values represented in the data? What about missing numeric values?

Accessing Data (libraries)

- Suppose we wanted to access data that doesn't live in the pre-loaded `sashelp` library; for example, nyc_bike.sas7bdat.
- We can use a hardcoded filepath to the SAS table we want to access.

```
proc contents data = "/home/u47369495/my_shared_file_links/u47369495/data/nyc_bike
run;

proc print data = "/home/u47369495/my_shared_file_links/u47369495/data/nyc_bike.sa
run;

proc means data = "/home/u47369495/my_shared_file_links/u47369495/data/nyc_bike.sa
var temp_high;
run;
```

- There are a few issues with operating as above, by specifying the full filepath every time we reference the data set.

Accessing Data (libraries)

- Definition: A *SAS library* is a SAS-recognized collection of one or more **SAS data sets** (recall previous def).

- Drawer = SAS library, File = SAS data set

- sashelp is a library that we've already been using! It contains pre-loaded (and, yes, boring) datasets.

Libraries: Creating one

```
libname libref "filepath/to/your/folder/containing/sas/datasets";
```

- libname: the keyword of this *global statement*
- You modify 'libref' to whatever you want your library named
- You modify the filepath inside " " to point to the folder containing SAS data sets.

```
/*Specify the library*/
libname s40dat '/home/u47369495/my_shared_file_links/u47369495/data/';
```

- The libref must
  1. be eight characters or less,
  2. must start with either a letter or underscore, and
  3. can include only letters, numbers, and underscores.

Types of Libraries

- In general, the programmer must define the libraries needed at the beginning of each SAS session
- However, SAS automatically provides one temporary and one permanent SAS library
- These are called **work** (temporary) and **sashelp** (permanent)
- We used sashelp in our examples - e.g., sashelp.heart is the heart data set, located in sashelp library. sashelp also contains other sample data sets.
- The contents of temporary libraries are deleted at the end of each SAS session
- The contents of permanent libraries are stored until deleted by you

Libraries: Using two-level data set names

- All SAS data sets have a two-level name consisting of
  1. the libref and
  2. the data set name, separated by a period.
- For example, sashelp.heart refers to a SAS data set named "heart" that is located within the sashelp library.
- If a data set is in the work library (temporary), a one-level name may be used. That is, the library name can be omitted.
- If the data set is in a permanent library, the two-level name must be used.

Libraries: Using them

From this:

```
proc contents data = "/home/u47369495/my_shared_file_links/u47369495/data/nyc_bike
run;

proc print data = "/home/u47369495/my_shared_file_links/u47369495/data/nyc_bike.sa
run;
```

To this:

```
/*Specify the library*/
libname s40dat '/home/u47369495/my_shared_file_links/u47369495/data/';


/*Use two-level name to access data*/
proc contents data = s40dat.nyc_bike;
run;

proc print data = s40dat.nyc_bike(obs = 10);
run;
```

Libraries: Using them

- In many companies (and in this class...), you aren't allowed to modify the 'master' data. In that case, you'd do something like this:

```
/*Specify the library*/
libname s40dat '/home/u47369495/my_shared_file_links/u47369495/data/';

/*create new sas data set as a copy of class_data.nyc_bike*/
data bb;
set s40dat.nyc_bike
run;

/*Use one-level name to access data from work library*/
proc contents data = bb;
run;

/*Or, use two-level name to access data*/
proc print data = work.bb(obs = 10);
run;

proc means data = bb;
var temp_high;
run;
```

Your Turn

1. Open a new program. Write a LIBNAME statement to create a library named s40dat that reads SAS tables in the shared data folder.
2. Run the code and verify that the library was successfully assigned in the log.
3. Go back to your program and save the program as t2_yt_libname.sas in your main course files folder.

Importing Comma-delimited (CSV) Files

- In Stat 40, we will focus on importing csvs.
- "Importing csv": taking a csv file and converting it to an equivalent SAS data set that can be manipulated in SAS.

```
PROC IMPORT DATAFILE="path/file-name.csv" DBMS=CSV
                          OUT=output-table;
RUN;
```

- Replace `"path/file-name.csv"` with the complete file path AND filename (with .csv extension) to point SAS to *exactly* where the csv is currently living
- `dbms=csv` tells SAS to expect a CSV file (as opposed to a .txt or .xlsx)
- Replace `output-table` with one (if using work library) or two-level SAS output table name.

Importing Comma-delimited (CSV) Files

- In Stat 40, we will focus on importing csvs.
- "Importing csv": taking a csv file and converting it to an equivalent SAS data set that can be manipulated in SAS.

```
PROC IMPORT DATAFILE="path/file-name.csv" DBMS=CSV
                          OUT=output-table <REPLACE>;
    <GUESSINGROWS=n|MAX;>
RUN;
```

- There are some options that you may find useful when using the proc import step.
- Add `replace` in the proc import statement to indicate the SAS output table should be replaced if it already exists.
- `GUESSINGROWS=n / MAX`: SAS to use first n / all rows to make column attribute decisions.

Case Study

GOAL: Import Ames housing sales data, using `guessingrows=MAX` so that SAS looks at all rows before making decisions about column attributes. Re-import it and view log messages. Save script as t2_cs_import.sas.

Where have we been

- We know what temporary and permanent SAS libraries are
- We know how to read in data from existing SAS libraries containing SAS data sets.
- We know how to read in CSV files and create a SAS data set.

Where we are going

- Exploring data with procedures
  - proc print,
  - proc means,
  - proc univariate,
  - and more!