Top 10 SAS Questions – Revisited

#10: What is a SAS data set/file?

- Before you can do anything with your data, SAS must be able to read it → it must be in a SAS data set.
    - Contains data and information about the data (e.g., variable and value labels)
    - Stored in a special format that is written to and read by SAS

- SAS data files and raw data are "read" into SAS differently.

- SAS can take a raw data file and convert it into a SAS data set.
    - sassypants.txt → sassypants.sas7bdat

#9: What's a libname and how do I use it?

- Remember, SAS data files tend to take on the form of *name*.sas7bdat.

- Within SAS, SAS data files have a two-level name separated by a period
    - 1st level: libref (SAS data library reference)
        - This is a nickname that corresponds and points to a particular location where the data are stored (jump drive, cd, directory, folder, etc.).

    - 2nd level: filename of SAS data set without extension
        - This is a *member name* that uniquely identifies the data set within the library

    - Before you can use a libref, you have to define it…commonly done by using the libname statement (but there are other ways, p. 67 *LSB*).
        - libname *libref* '*drive:\directory*';
        - This place must exist!

- Examples…
    -
      ```
      libname tars 'T:\TARS Data\Data';
      ```

        - Here, I have a libref called tars.
            - "tars" is what I have named the location 'T:\TARS Data\Data.'

            - If I want to locate any files that are in 'T:\TARS Data\Data', or create a file and store it in 'T:\TARS Data\Data', I can refer to it with the libref "tars."

            - Note: Using the libname to define a libref does not change the name of the folders. For example, the folder “Data” is still called “Data.”

        - Within the ‘T:\TARS Data\Data’ folder, there are the following SAS data files:

          wave1.sas7bdat wave2.sas7bdat wave3.sas7bdat
          wave12.sas7bdat wave13.sas7bdat wave123.sas7bdat

    - `libname mytars ‘T:\TARS\Meredith’;`
        - Here, I have a libref called mytars.
            - “mytars” is what I have named the location ‘T:\TARS\Meredith’ (a folder on the server within the TARS folder called “Meredith”).

            - If I want to locate any files that are in ‘ T:\TARS\Meredith’, or create a file and store it in ‘ T:\TARS\Meredith’, I can refer to it with the libref “mytars.”

    - I want to use the SAS data set called wave123.sas7bdat. I want to be able to save the data set and any changes I have made as a new data set called “example” in the folder “Meredith” on the server in the TARS folder.

      ```
      libname tars ‘T:\TARS Data\Data’;
      libname mytars ‘T:\TARS\Meredith’;
      data mytars.example; set tars.wave123; run;
      ```

        - Remember, “data *new*; set *old*;”

- Once you use a libname statement to define a libref, that folder can be seen in your explorer window.

- If you click on one of the libraries, SAS will show you what SAS data files are in the library.

#9: What's the difference between a temporary and permanent data file?

- Temporary: exists only during current job or session and is automatically erased by SAS when you close SAS

- Permanent: exists somewhere (jump drive, folder, etc.) even after SAS session has ended

- You don't explicitly tell SAS to make data set temporary or permanent…it is implied by the name you give it when you create or reference it.

  - Temporary data set: give it a two-level name with WORK for the *libref* OR give it a one-level name (member name only).
    - This data set will have an implied libref of "work."

  - Permanent data set: give it a two-level name with anything but WORK for the *libref.*
    - REMEMBER…for a permanent data set, you must have defined the libref with a libname statement!

```
libname tars 'T:\TARS Data\Data';
data a; set tars.wave123; run;
```

#7: SAS shut down and I didn't save my program – what should I do?!

- The latest unsaved version of your program is saved automatically by SAS and held in a temporary folder with the name Autosave of program.$as."

- The default setting for this feature is 10 min. but can be changed :
  Go to the Program Editor: Tools, Options, Preferences, Edit, Autosave

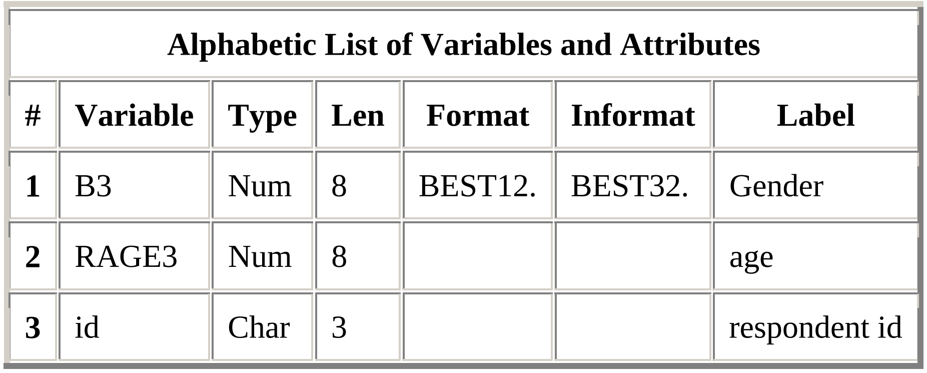
- To retrieve the last version you do the following:
  - In SAS v. 9, the autosave files are defaulted to save to C:\Documents and Settings\username\Application Data\SAS\EnhancedEditor
  - Or you can perform a search with the word "Autosave."

- For more information, see http://support.sas.com/techsup/unotes/SN/012/012392.html

#6: How do I convert a character variable to numeric and vice versa?

- First, how do you know if your variable is formatted as character or numeric?
    - Codebook
    - Proc contents

```
proc contents data=mer.a;
run;
```



Alphabetic List of Variables and Attributes						
#	**Variable**	**Type**	**Len**	**Format**	**Informat**	**Label**
1	B3	Num	8	BEST12.	BEST32.	Gender
2	RAGE3	Num	8			age
3	id	Char	3			respondent id

- Character to Numeric
    - *newvar* = input(*oldvar*, *informat*);
        - Informat is the format of the numeric variable you are converting to.
    - Ex: You want to convert variable *id* (e.g., 218, 712, 728) from character to numeric (old variable = *id*, new variable = *rid*).

```
rid = INPUT(id, 3.);
```

- Numeric to Character
    - *newvar* = put(*oldvar*, *format*);
        - Format is the format of the numeric you are converting from.
    - Ex: You want to convert variable *date* (e.g. 20060705, 20051210) to character (old variable = *date*, new variable = *date1*).

```
date1 = PUT(date, 8.);
```

#5: How can I efficiently recode my numeric missings to .'s?

- Sure, you could code each individual variable…

```
if var1 = -98 or var1 = -99 then var1 = .;
if var2 = -98 or var2 = -99 then var2 = .;
if var3 = -98 or var3 = -99 then var3 = .;
if var4 = -98 or var4 = -99 then var4 = .;
                    …
```

- …but to do it efficiently, use an array!

```
array v (7) var1-var5 age marstat;
        do i=1 to 7;
                if v(i)=-98 or v(i)=-99 then v(i)= . ;
                        end;
```

#4: How can I use my SAS data file in Stata?

- The easiest way is probably using Stat/Transfer, if available.
- If you don't have access to Stat/Transfer, you can use SAS coding.
- Let's imagine I have a SAS file called "a" saved in 't:\TARS\Meredith' and that I'm on a beach far, far away from B.G. and Stat/Transer.

```
libname in 't:\TARS\Meredith';
*referencing the location of the SAS file to convert;

libname out XPORT 't:\TARS\Meredith\trans.xpt';
*referencing the location and name of file to create;

data out.trans;
     set in.a;
run;
```

- Now, open Stata
- In Stata…
  - o File
  - o Import
  - o FDA data (SAS XPORT)
  - o New window will open, browse for your SAS transport file wherever you stored it.
  - o Click Okay

#3: Why don’t I have any output?

- Take a look at your log file…it will give you clues to the problem
- Three types of messages
  - Errors
    - Red print in log; location of error usually underlined
    - Program won’t run
    - Usually syntax or spelling mistake

  - Warnings
    - Green print
    - Less serious than errors because SAS will still run
    - Be careful…even though SAS is running, the warning may mean that it is not running the way you want it to

  - Notes
    - Blue print
    - Sometimes just give information, but sometimes indicate a problem

Note: Variable x is uninitialized.
Warning: Variable x is not found.
Error: Variable x is not found.

- These errors mean that your program includes a reference to a variable name that SAS has never seen.

- Variables are normally initialized when they are read in or created.

- If you use a variable for the first time in a way that doesn’t assign a value to the variable, SAS tries to fix this by assigning a value of missing for all observations.

- If this happens in the data step, SAS will print the note, initialize the variable, and continue running.

- If you get one of these messages in a PROC step, SAS will attempt to run the program, but it may not be able to continue because SAS cannot find (or doesn’t recognize) the variable you are listing.

- How does this happen?
  - misspell a variable name
  - use a variable that was dropped or not created (sometimes the variable is created later in the program)
  - mix up PROC and DATA steps

#2: How do I use these date variables?

- Data sets have different ways of handling dates…
  - separate variables for month, day, year
  - century months
  - one variable yyyymmdd
  - etc.

- Dates are often used for calculating…
  - Durations (e.g. duration of cohabitation)
  - Age at a particular time (e.g. age at first birth)

- But how do you do this when your dates are set up like this:
  - Month = month of birth
  - Year = year of birth

  …Use century months!

- To convert from month & year to century months…
  - cm = (year*12) + month
  - Allows you to more easily calculate duration
  - Start cohabiting March 2003, end cohabiting June 2007.
    - Cmstart = (2003*12) + 3 = 24039
    - Cmend = (2007*12) + 6 = 24090
    - Durcohab = cmend – cmstart = 51

- To convert from century months to months, years…
  - Cmonth/12
    - 24039/12 = 2003.25
    - 24040/12 = 2003.333
    - 24041/12 = 2003.4167
    - 51/12 = 4.25

    - Number to left of decimal is years. To get months, multiply number to right of decimal by 12.

- To deal with a dates like these: dob = yyyymmdd…

  - “Pick off” year, month, and day from variables like this.
    - In the DATA step, use the SUBSTR function.

      - *new-variable* = SUBSTR(*varname*,start,length)

    - Variable to substring must be character…new variables can be converted to numeric later if needed

- "varname" is the name of the original variable, "start" is the starting point for the substring (how many characters in), and "length" is the number of characters SAS is should read to create the new variable's value.

```
birthyr = substr(dob, 1, 4);
birthmo = substr(dob, 5, 2);
birthday = substr(dob, 7, 2);
```

#1: Where can I find fabulous examples and help sheets?

http://www.bgsu.edu/organizations/cfdr/dataresearch/programming_research.htm

Reference:

Delwiche, L. D. & Slaughter, S. J. (2003). *The Little SAS Book: A Primer, 3rd Ed.* Cary, NC: SAS Institute, Inc.