Common Coding Mistakes: SAS

1. **Missing semicolon - this is the most common SAS coding error. Make sure each line of your SAS code ends with a ;**

Example:

BAD - missing a ; in the first line:

```
proc means data=one
var age;
run;
```

GOOD:

```
proc means data=one;
var age;
run;
```

2. **Confusion between dataset to create and dataset to copy - `DATA` tells SAS to create a new dataset and the `SET` statement tells SAS that your new dataset is a copy of this already existing SAS dataset. It is very important to get this correct because mixing up these statements could lead to overwriting permanent files.**

Examples:

```
      libname folder 'C:\temp';
*To make a temporary copy called "temp_copy" of a permanent SAS
dataset called "permSASdata";
      data temp_copy;
        set folder.permSASdata;
*To make a temporary copy called "temp_copy" of a temporary SAS
dataset called "tempSASdata";
      data temp_copy;
        set tempSASdata;
*To make a permanent copy called "perm_copy" of a temporary SAS
dataset called "tempSASdata";
      data folder.perm_copy;
        set tempSASdata;
*To make a permanent copy called "perm_copy" of a permanent SAS
dataset called "permSASdata";
      data folder.perm_copy;
        set folder.permSASdata;
```

3. **Using `input` statement to create a new SAS dataset that is a copy of an existing SAS dataset - If the dataset you want to copy is a SAS dataset, you should always use a `SET` statement. `INPUT` is only used to specify variables that are in a non-SAS dataset (like when you have external files such as .txt or want to manually enter the data using `datalines` or `cards`). Hint: use the data entry flowchart from class seven.**

Example 1: I want to create a new SAS dataset called *new* that is a copy of the existing SAS dataset *old.*

```
      data new;
       set old;
      run;
```

Common Coding Mistakes: SAS

Example 2: I want to create a new SAS dataset called *new* from a the file *old.txt.*

```
data new;
infile 'C:/.../mySASdata/old.txt'
input id age sex sysbp;
run;
```

Example 3: I want to create a new SAS dataset called *new2* from a data matrix.

```
data new2;
input studyid name $ sex $ age weight height;
cards;
1 carol f 22 120 64
2 joe m . 130 68
3 ed m 40 120 69
4 ann f 29   . 65
5 leslie . 17 175 73
6 julie f 55 125 .
;
run;
```

4. **Data step overrides previous dataset - (example shown below) In these situations, you may have variables created in the functions that change the original data. This is ESPECIALLY important when you are overriding a permanent dataset.**

```
data dataset1;
set dataset1;
*[Functions to manipulate data];
run;
```

We **HIGHLY** recommend that you **NEVER** overwrite any datasets in SAS. Always make sure to give your new dataset a different name from the currently existing one, for example, we would make the following change to the SAS code shown above:

```
data dataset2;
set dataset1;
*[Functions to manipulate data];
run;
```

5. **Filename/dataset name contains characters that can't be read by SAS - i.e. Make sure that you do not have parentheses "()" in your filename.**

Example 1:

```
libname perm 'C:/…';
```

BAD - there is a () in the dataset name:

```
data new;
set perm.original_data(2);
run;
```

GOOD:

```
data new;
set perm.original_data;
run;
```

Common Coding Mistakes: SAS

Example 2:

BAD - there is a () in the filename:

```
proc import datafile='C:/.../mydataset(1).csv'
```

GOOD:

```
proc import datafile='C:/.../mydataset.csv'
```

Note: Make sure your file, mydataset.csv, is closed before trying to import into SAS (common coding mistake below)

6. **Trying to import a file into SAS using `proc import`, but cannot because the file is currently open. Make sure any files you are trying to import into SAS are closed first.**

Example:

```
proc import datafile='C:/.../mydataset.csv'
out=newdata
dbms=csv replace;
run;
```

If I have the file, mydataset.csv, open on my computer and run the SAS code above it will not work until I close the file first.

7. **Dataset location (`libname`) not specified - hint: put all library specifications at the top of your program**

Example: Instead of listing libname statements throughout your SAS program list all libraries at the top of your SAS program so you can easily keep track of them all.

[NEW SAS PROGRAM]

```
libname dem 'C:/.../mySASdata/demographics';
libname rfs 'C:/.../mySASdata/riskfactors';
libname outc 'C:/.../mySASdata/outcomes';
```

Now that I've listed all of the libname statements at the top of my SAS program I can start creating datasets and running SAS procedures.

8. **Single quotes used in a label that contains an apostrophe - we suggest using double quotes throughout programming, reserve the single-quotation-mark for contractions only. Bonus: this is the best form of quotations to use when invoking macro variables (macros are not covered in BS723).**

Common Coding Mistakes: SAS

Example:

BAD:

```
data one;
set old;
label SBP='Subject's SBP';
run;
```

GOOD:

```
data one;
set old;
label SBP="Subject's SBP";
run;
```

9. **When using `if/else` statements to create new variables always start with `if oldvar = . then newvar = <missing` (. if newvar is to be numeric or "" if newvar is to be a character variable)>; so that when a "less than" statement is used, missing values are not assigned to that group.**

Example: I want to create a new character variable, BMICAT, from the existing numeric variable, BMI.

BAD - all missing BMI values will be in the BMICAT group "Normal or less".

```
data new;
set original_data;
if BMI<25 then BMICAT="Normal or less";
else BMICAT="Overweight or more";
run;
```

GOOD - all missing BMI values will have missing values for BMICAT.

```
data new;
set original_data;
if BMI=. then BMICAT="                          ";
else if BMI<25 then BMICAT="Normal or less";
else BMICAT="Overweight or more";
run;
```

Note: The double quotes are spaced far apart in order to avoid needing a `length` statement. Remember that when creating a new character variable in SAS it is automatically given a length of 8 characters or the length of the first value (if more than 8 characters). By spacing the double quotes by at least 18 characters I've insured that none of my remaining character values (`Normal or less` and `Overweight or more`) get cut off.

10. **Mixing numeric missing with character variable, i.e. when creating a character variable, newcat, and assigning the missing group to `newcat=.` but trying to create other categories using quotations like `newcat="Diseased"` or `newcat="1"`. Hint: Always use quotes for character variables, except when using `datalines` or `cards`.**

Common Coding Mistakes: SAS

Example 1: I want to create a new numeric variable, BMICAT, from the existing numeric variable, BMI. Note that quotes are not used anywhere.

```
data new;
set original_data;
if BMI=. then BMICAT=.;
else if BMI<25 then BMICAT=1;
else BMICAT=2;
run;
```

Example 2: I want to create a new character variable, BMICAT, from the existing numeric variable, BMI. Note that quotes are always used for denoting the values of the character variable.

```
data new;
set original_data;
if BMI=. then BMICAT="                          ";
else if BMI<25 then BMICAT="Normal or less";
else BMICAT="Overweight or more";
run;
```

Note: The double quotes are spaced far apart in order to avoid needing a `length` statement. Remember that when creating a new character variable in SAS it is automatically given a length of 8 characters or the length of the first value (if more than 8 characters). By spacing the double quotes by at least 18 characters I've insured that none of my remaining character values (`Normal or less` and `Overweight or more`) get cut off.

11. **Missing `else` for logic statements (if/then statements). Make sure that you account for all possible values.**

Example:

BAD - all BMI values of 25 or greater will have a BMICAT of missing (.)

```
data new;
set original_data;
if BMI=. then BMICAT=.;
else if BMI<25 then BMICAT=1;
run;
```

GOOD - all possible BMI values are taken into account in the creation of BMICAT

```
data new;
set original_data;
if BMI=. then BMICAT=.;
else if BMI<25 then BMICAT=1;
else BMICAT=2;
run;
```

Common Coding Mistakes: SAS

Hint: Use SAS to help you check your coding. For example, I could use the following code to make sure all missing BMI values are contained in BMICAT=., all non-missing BMI values less than 25 are in BMICAT=1, and all BMI values greater than or equal to 25 are in BMICAT=2.

```
proc means data=new min max;
class BMICAT;
var BMI;
run;
```

12. **If you get a SAS Error that says "variable does not exist" you have either:**
    1) spelled the variable name wrong (check your dataset), or
    2) accidentally deleted the variable or dataset. Check the dataset in your folder to make sure the dataset and/or the variable exist!

13. **Data not sorted by the variable. Hint: Whenever you need to use a `by` statement make sure to sort by that variable first.**

Example: merging 2 datasets by subjectid.

```
proc sort data=one; by subjectid; run;
proc sort data=two; by subjectid; run;
data three;
merge one two;
by subjectid;
run;
```

14. **Format errors - add `options nofmterr;` at the top of your program to produce warnings instead of errors in your log.**

Example:

[NEW SAS PROGRAM]

```
options nofmterr;
libname dem 'C:/.../mySASdata/demographics';
libname rfs 'C:/.../mySASdata/riskfactors';
libname outc 'C:/.../mySASdata/outcomes';
```

Now that I've listed the nofmterr option and all of the libname statements at the top of my SAS program I can start creating datasets and running SAS procedures.

15. **Format names (`value` [fmtname]) must end in an alphabetic character or "_". Ending a format name with a number will produce errors.**

Example:

BAD - format name ends in a number, this will produce an error in SAS.

```
proc format;
value sexf2 0='Female'
            1='Male';
run;
```

Common Coding Mistakes: SAS

GOOD - format name ends in a letter or underscore (_).

```
proc format;
value sexf 0='Female'
           1='Male';
run;
```

16. **In linear regression analysis, it is easy to think that the df=1 because it is written in the table where you get the slope, but this is not correct. The df for the beta estimate is the error df listed in the F table.**

Example:

The REG Procedure
Model: MODEL1
Dependent Variable: fev

Number of Observations Read	654
Number of Observations Used	654

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	29.56968	29.56968	41.79	<.0001
Error	652	461.35015	0.70759		
Corrected Total	653	490.91984			

Root MSE	0.84119	R-Square	0.0602
Dependent Mean	2.63678	Adj R-Sq	0.0588
Coeff Var	31.90198		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	2.56614	0.03466	74.04	<.0001
smoke	1	0.71072	0.10994	6.46	<.0001

Results for the variable smoke:
Beta estimate= 0.71
t = 6.46 with 652 df
p<0.0001