readcharacter.sas: Explanation of code

Goals of code:

- Potential issues when SAS reads character data
  - data step and default length of character variables
  - proc import

SAS and character data: background:
SAS stores character values in a fixed length field. By default, that is 8 characters long. That means SAS will only store the first 8 characters of a character string. The value "long string A" will be stored (by default) as "long str". The space counts as one character.

Fixed length fields are very efficient. If you want to look at the 100,500'th observation in a data file with 10,000,000 observations, SAS knows exactly where to go. They are frustrating when you need to save strings longer than 8 characters.

In general, the solution is to tell SAS when you have a long character string so it knows to reserve more than 8 characters for it. When read the data by proc import, you need to help SAS figure out the length of the longest string.

data basic1; proc means; proc print;
The first data step reads the file using the default length. You don't get any warning in the log file but you have problems after that. The proc means only sees one group and when you print the data set, you see why. The values "longname1" and "longname2" are saved as the same thing: "longname".

`length name $ 20;`
The solution is to tell SAS that name is a character variable with more than 8 characters. This length statement does that. The second piece is the variable name you are defining. The `$` tells SAS this is a character variable and the number (20 here) specifies how much space to reserve. Here, that's 20 characters. If you had really long strings (e.g. species names with the genus), you may need more than 20 characters. You specify the length.

The length statement **must** go before the input statement. That's because SAS needs to know the desired length before it saves a value in the variable called name.

proc import issues:
When you write a data step, you specify the variable names and how to read each field (character or numeric). Proc import has a more difficult job. It has to figure that information out from what it sees in the file.

The variable names are easy: they are given on the first line of the file.
How to read each field is more difficult. proc import looks at the first few lines of the file and uses

those to figure out how to store each observation. By default, proc import looks at the first 20 rows in the file. If a column has something other than numbers in any of those 20 rows, that variable will be stored as a character variable. The length of that character variable is just enough to store the longest character string in those 20 rows.

The readcharacter.csv file illustrates two ways that default mechanism may fail. The first 20 rows may not correctly indicate the length of a character variable and the first 20 rows may not correctly indicate that a variable is really a character variable.

The first column has three values: "group 1", "group 2", or "group 10". The first two values are 7 characters long; the third is 8 characters long. This would not be a problem if there was a "group 10" observations in the first 20 rows. There isn't; the first "group 10" value is in row 24 of the file. So, SAS thinks the name1 variable is 7 characters long. All the "group 10" values are truncated to 7 characters and are saved with the same value as the "group 1" observations.

The second column has three values: "5", "10", and "A". It needs to be stored as a character variable, because of the "A" values. When proc import looks at the first 20 rows, all it sees is numbers, so name2 is stored as a numeric variable. When SAS reads the last few rows, it fusses (in the log file, about invalid data)about the "A" values and stores them as a missing value (.).

One solution is to reorder observations so proc import sees what it needs to in the first 20 rows.

guessingrows = 100;:
The better solution is to tell proc import to look at more of the file. The guessingrows statement tells proc import how many rows to look at to decide on the appropriate structure. The number (100) is the number of rows to look at. This statement is a statement in proc import. It is not an option to the proc import statement. So, the proc import statement ends with a ;, then you have the guessingrows statement, then the run; statement.

proc import replace option:
One final issue that may occur when you try to import a data file more than once. proc import is conservative. If the data set already exists, proc import won't read the data file again. Doesn't matter whether the first time failed, or whether you have edited the data. The default behaviour is to check whether the data set exists and stop if it does. You do get an error message in the log window. That message tells you why proc import stopped and suggests you include the REPLACE option.

Adding the word replace to the proc import statement (i.e., between proc import and the ;) tells SAS to read the data file even if the data set already exists.