Creativity2.sas: Explanation of code

Goals of code:

- Read a space delimited file using a data step
- Randomization test

Reading a space delimited (.txt) file: `data creativity;` through `run;`
The data step is the major way to convert a data file into a SAS data set. In fact, the actual work done by `proc import` is actually done by a data step. That data step code created by proc import is shown in the log window.

Data steps are used to create or modify SAS data sets. Remember that all the proc steps (the things that do statistical analyses) work on SAS data sets. So, most SAS programs start with a data step, either written by you or written by proc import. You can also use data steps to create new variables or reorganize a data set.

`data creativity;`
This tells SAS that you want to create a SAS data set (`data`) and gives it a name, here `creativity`. All the statements from data ; to the run; are part of the data step. They will not executed until after you submit the `run;` command.

`infile` names where to find the raw data. The file name and extension, e.g., `creativity.txt` are in quotes. These can be single or double quotes, but they need to match. This file will be looked for in your working directory. Remember the working directory is named in the bottom edge of the SAS window. You can change it by double-left clicking on that folder name. The creativity.txt file has a header line with variable names (like all my class data files), so the real data starts on the second line. The `firstobs=2` option tells SAS to start reading data from the second line.

`input` describes to SAS how to convert one line in the raw data file into one observation in the SAS data set. The input line tells SAS how many values to read on each line, what names to give the variables, and how to read the values. This is described in the Introduction to SAS document. The values of `treatment` are character strings, so they must be stored as character strings. This is indicated by the $ after the variable name in the input line. The values of `score` are numeric, so **do not** put $ after score.

Reminder: To save a variable as character strings, add $ after the variable name. The default is to save a variable as numbers.

If you try to read a character string ("A", "red", "usual") as a number, SAS can't interpret it and stores a missing value (a lone decimal point). If you read a number as a character string, SAS happily saves the string of characters, but you can't do any arithmetic on a string of characters. So if you save a number with $, you can't calculate means or do any other computations.

I generally check the line in the log that says "The data set WORK.CREATIVITY has 47 observations and 2 variables". I check that SAS read the number of observations I expected and produced the number of variables I expected.

Reading space delimted data saved "in" the SAS program `cards;` or `datalines;`
If you are using a remote desktop connection to the SAS, you will need to save data files in the myfiles space accessible from the SAS server. I will try to find out how to do that. Here is a work around. That is to include the data in the SAS program. That's what the second and third data steps do.

If the file is space delimited (i.e., a .txt file), all you have to remove the infile statement (because you don't need to identify an external file) and add a cards; statement after everything else in the data step. datalines; is a synomyn for cards;. Either works identically. Then copy the data and paste it below the cards statement. SAS identifies the end of the data when it sees a semicolon. I add run; That both ends the data (because of the ;) and ends the data step (because of the run;). Some folks put a lonely semicolon after the last data line. They then need a line with run; to signal the end of the data step.

Reading comma delimited data saved "in" the SAS program
`infile cards delimiter = ',';`
Some class data comes in .csv files. These have a comma, not a space, between values for an observation. proc import will read these files from an external file. Here's how you can read them when you have pasted the contents into a SAS program. You use cards; followed by the data, just as done above for space delimited files. You need to tell SAS that the separator between variables is a comma. That is done by an infile statement. The code word cards (not in quotes) tells SAS that your modifying how to read the cards; data. delimiter = specifies the character (or multiple characters) that are delimiters between fields.

Permutation test: `proc npar1way scores=data;`
The procedure name has the digit 1 in it. It is shorthand for NonPARametric ONE WAY. We'll talk later about nonparametric and one-way. The `scores=data` goes *before* the semi-colon. It is part of the proc statement and tells SAS to permute the data. In a few weeks we'll use proc npar1way for non-parametric tests using ranks.

Just as with proc univariate (described in creativity.sas, last week and repeated this week), the class statement specifies the grouping variable and the var statement specifies the response. The exact statement specifies that we want to enumerate all possible permutations of treatments to observations. For large data sets, this can be very time consuming (and not necessary). The `/maxtime=10` option tells SAS to spend no more than 10 seconds on the problem. This is a failsafe in case you ask for a computation that will take months. If you request a "very long" computations, SAS will stop after 10 seconds and you will see a note in the log window telling you that has stopped before finishing. If you see this, you can increase 10 to a larger maxtime, but I recommend changing to a random sample of permutations (next proc step).

Including the exact statement is important. If you omit it, SAS uses something called a normal

approximation, which in this case is no different from the usual t-test.

Randomization test: `proc npar1way scores=data; exact /mc n=10000;`
If the problem is very large, the p-value can be estimated from a random sample of permutations (what I call a randomization test). It isn't necessary to enumerate them all. The `/mc` option requests a Monte-Carlo analysis, i.e. a random sample of possible permutations. The `n=10000` requests 10000 samples.

The most difficult part of this analysis is finding the result you really want. You want the two-sided p-value from the Exact test (or Monte Carlo approximation). That number is the Two-sided ..., Estimate number in the box labeled Monte Carlo Estimates for the Exact Test.