Creativity.sas: Explanation of code

Lines from creativity.sas are repeated here so you know what each explanation refers to.

Because SAS generally produces lots of results, including many you don't want/need, these explanations include both what code to write and where to find desired results.

Do not copy and paste code from a pdf file. That works 98% of the time but not 100% of the time. The problem are those characters or character combinations that pdf files make "look pretty". Quotes are one example. They do not copy and paste correctly.

The .sas file has the run; commands to make the

Goals of code:

- Calculate summary statistics
  - For all observations, or for each treatment
- Display the creativity data (dotplot, histogram, boxplot)
  - As one group or for each treatment
- Saving results to copy to Word

Confusing things:
Copying code from the pdf file. Copy it from the .sas file or load the entire file into the program editor window.
Forgetting to submit/run after correcting errors in the code.
Forgetting to submit the run; command.
Forgetting to submit the ods rtf close; to finish saving results in a Word rtf file.

`/* anything */` **Comments:**
Any text between `/*` and `*/` is a comment and ignored. This can span many lines. The SAS program editor window will show the comment in green. If code you want to run is in green, it probably won't be run. Figure out why your comment didn't end where it was supposed to.

`proc import` **Reading a .csv file**
Some proc are set up so that follow on statements (lines with ; at their end) specify what needs to be done. proc import wants its information in the proc statement. The necessary pieces are where the data comes from and the name of the SAS data set that will store the data.
`datafile='name in quotes'` tells where the data comes from. The file extension (e.g. `.csv`) is necessary because that's how SAS knows how the information is stored. The file name and extension are a string of characters inside quotes. Those may be double quotes (`"`) or single quotes (`'`). They must match, i.e. if the string starts with a double quote, it **must** end with a double quote.

`out=name` specifies the name of the SAS data set. SAS will store this in the WORK library, so the full name will be WORK.NAME. The data set name can be the same as the file name or different. Names of data sets only become important when you save more than one.
`replace` is an option I recommend. Without it, SAS is very conservative. It checks whether the data set you specify already exists. If it does, then proc import stops; the old data set is not overwritten. This is a problem when you have corrected data and really do want to overwrite the old version. replace gives SAS permission to overwrite the data set.

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.

`proc univariate;` **Descriptive statistics, possibly for multiple groups**: PROC UNIVARIATE calculates and reports many different statistics. The VAR statement names the variable to be summarized. You can name multiple variables if you want more than one summary. The optional title statement provides a title (in quotes).

SAS reports a lot of numbers, including many that we won't talk about. You will probably be interested in:
In the box of results labeled Moments:
N: number of observations
Mean: sample average
Std Deviation: sample standard deviation
In the box of results labeled Basic Statistical Measures:
Mean: a repeat of the sample average
Median: sample median Std Deviation: a repeat of the sample standard deviation
Interquartile Range: the IQR
Coeff Variation: the coefficient of variation

You may be interested in:
In the Quantiles box: 100% Max, 75% Q3, 25% Q1 and 0% Min:
the max, 75'th percentile, 25'th percentile, and minimum.

Some of the other numbers will be used later. The rest have specialized uses.

`proc univariate; class treatment;` **Descriptive statistics and histograms for groups**:
Adding a class statement to the proc univariate code tells SAS to do everything multiple times, once for each unique value of the treatment. Same format for the numeric output, but the top of the page is labeled treatment = to indicate which treatment is being described.

Note: Remember that the grouping variable goes in the class statement and the result variable (to be described) goes in the var statement. If you reverse these, e.g. class score; var treatment;, you get mush or worse.

Saving a copy of results in a rtf file: `ods rtf file='creativity.rtf';`

This line is optional. It tells SAS to save a copy of the output in the RTF format file called creativity.rtf. This file is easily readable by WORD, and makes it easy to copy selected results to another document.

When you are done, you need to submit `ods rtf close;` (at the end of the code file), but can be anywhere if you want to turn on and off to selectively save results. When you close the output, SAS will (should) start Word so you can see the file. To copy parts of the output to another file, just select the part you want, copy and paste into your file.

graphs in SAS:
SAS graphics are not fantastic. They are much better then they were 10 years ago, but I (and many others) use other programs for graphics. I show how you can use SAS to draw homework quality graphs. There are three different sorts of SAS graphics: base plots, which date from the days of lineprinters (really coarse graphs), SAS/GRAPHICS functions and SAS/StatisticalGraphics (SG functions). The SG functions are the first to be both easy to produce and reasonable plots.

`proc sgplot; dot score;` **Dot plots of all the data**
The Y axis shows the values; the X axis, labelled Frequency, indicates the number of repetitions of that value.

`proc sgplot; scatter x=treatment y=score;` **Dot plots of data for each group**
Scatter draws a scatter plot of a data set (Y vs X). This produces a vertical dot plot when Y is continuous and X indicates groups.

`proc univariate; histogram score;` **Histograms, either of all values or by group**
Instead of `var score;` saying `histogram score;` produces a histogram of the values. If you add the class statement (see earlier proc univariate code), you get stacked histograms.

`proc boxplot;` **Boxplots**: Proc boxplot produces side-by-side boxplots of data from multiple groups. One quirk is that it requires each group to be all together. The easiest way to ensure this is to sort the data by treatment first. The `proc sort; by treatment; run;` does this. The plot statement specifies the response variable and the grouping variable. The response is first, then the group, separated by a *. Again, if you reverse these, the result is either mush or a lot of errors.

Note: If you expected two boxes (for two groups) and got many, the data set is probably not sorted.

`proc sgplot; vbox` **Boxplots for one group of observations**
When you only have one group of observations (so no grouping variable), the easiest way to get a boxplot is use proc sgplot. The vbox command names the variable to plot. That name goes on the y axis of the plot. It is also possible to create a grouping variable with only one level and use proc boxplot, but that's no longer necessary.

Closing the rtf output: `ods rtf close;`
This tells SAS to stop saving results in the RTF file. SAS will then open up WORD so you can look at the results.