Creativity.r: Numerical and Graphical summaries of data
Explanation of code

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

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.

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:
Trying to copy code from the pdf file. Don't. Copy from the associated code file or open that file and execute commands from that file.
Setting the working directory so data files can be found and saved plots go where you can locate them.
Remembering that capitalization matters

```
# creativity.r: numerical and graphical summaries of data
```
The # symbol starts a comment. All text from the # to the next end-of-line is ignored.

```
getwd()
```
Print the default working directory. If your desired working directory is a folder under this, you can use a relative path in the setwd().

```
setwd('name of folder')
```
Set the default working directory. Can use a relative path or an absolute path, i.e., starting at `c:` or some other drive. R will look for code and data or save files in the specified directory. The Rstart document describes working directories and how to set them.

```
creativity <- read.csv('creativity.csv', as.is=T)
```
Read the csv format file and create the creativity data frame. Details in the Rstart document.

```
names(creativity)
```
Print names of the variables in creativity. Not essential. Helpful in plain R; the data window in RStudio gives you this information automatically.

`summary(creativity$score)`
Print a 6 number summary of the values in creativity$score. See the Rstart document if you don't recognize creativity$score as the score variable in the creativity data frame.

`mean(creativity$score)`
`median(creativity$score)`
Functions to compute specific summary statistics

`tapply(creativity$score, creativity$treatment, summary)`
Apply a function to subgroups of data. Requires three arguments:
1) The source of data, here creativity$score
2) How to define subgroups, here using creativity$treatment
3) The function to apply to each subgroup, here summary.
Output looks slightly different when the function returns a scale (e.g., mean or median). You should be able to figure out what you get by looking at it.

R graphics: There are three different sets of graphing functions in R: base graphics, lattice graphics, and ggplot/ggplot2 graphics. I will illustrate base graphics because that's what I use. The ggplot systems are more complicated to learn but they allow an almost infinite amount of plot customization. If you're already familiar with a graphics system, feel free to use it. You don't have to use my code.

`stripchart(creativity$score)`
Basic horizontal dot plot. Overplots repeated observations.

Most R function have optional arguments to control / refine what they do. These options are described in the help file for the function. To see that help, type: `?stripchart` or `help(stripchart)` The help file appears in a new browser window (plain R) or the help window of RStudio. The options are indicated by option= value. I will introduce useful options. There are many, many options that I overlook.

`stripchart(creativity$score, method='stack', offset=0.6)`
For the stripchart, we probably prefer to separate the overplotted symbols. That is done by adding `method='stack'` to the function. Additionally, `offset=` indicates how much separation to add.

`stripchart(score~treatment, data=creativity, method='stack', offset=0.6, pch=19)`
I prefer a filled dot to the open square that is the default plot symbol. That is done by `pch=`. Some options are specific to stripchart, e.g., method= and offset=. Other options are general to all basic graphics plotting function, e.g., pch=. You'll see pch= a lot in my code. I will demonstrate other useful plotting options throughout the semester.

`stripchart(creativity$score, vertical=T, method='stack', offset=0.6, pch=19)`
More elaboration. `vertical=TRUE` makes the plot vertical. `T` is shorthand for `TRUE`. If you look in the help file, you see that the default value for this option is `FALSE`, i.e., a horizontal plot. In both cases, the capitalization really matters.

`stripchart(score~treatment, data=creativity, method='stack', offset=0.6, pch=19)`

Side-by-side dotplots produced using a formula to specify the variable and groups. ~ specifies a formula. The Y variable (values to plot) goes on the left-hand side; the grouping variable (X) goes on the right-hand side. When you have a formula, you can indicate the data frame explicitly, e.g., `creativity$score ~ creativity$treatment`, or use the more convenient `data=` format shown in the code.

`hist(creativity$score)`
Draw a histogram. Lots of options to change labels (axis or title) and number of histogram classes.

I do not know an easy base graphics way to draw side-by-side histograms. I use side-by-side boxplots instead (as do most folks).

`boxplot(creativity$score)`
Draw a vertical boxplot of all the values in creativity@score.

`boxplot(score~treatment, data=creativity)`
Side-by-side boxplots using the formula specification.

Moving plots into a Word document
There are three (at least) ways to do this. Some of the details depend on whether you are using plain R or RStudio.

`savePlot('boxplot.emf', type='emf')`
(either R or RStudio). Saves the current plot in the specified file. File goes in your working directory unless you add a path to the name. You must provide the file extension (e.g., `.emf`), which is what Windows uses to identify the contents. Type= specifies the type of file. `'emf'` is an enhanced meta file. This and TIFF files (`type='tiff'`, with a `.tiff` extension) are the two that I find go into Word most easily. In Word, Include / picture will load the picture.

RStudio alternatives:
After plotting something, the lower right window should be a Plots window and show the graph. Look for the Export button in the 2nd menu bar. If not there, click Plots and you should see the Export button.
1) Click Export / Save as image. The dialog box allows you to specify the format, the directory, and the filename. The metafile format is very useful for moving to Word. The directory is your working directory, but you change this by clicking the Directory button. The file name is Rplot by default. You should provide a more descriptive name. You can also change the plot size. You can resize plots in Word, but resizing before saving usually looks better.

plain R alternatives:
1) Click on the graph to make it active, then select File/Save As. The menu will include various file types. Metafile and TIFF are the two that I use to move pictures to Word.
2) Click on the graph to make it active, then select File / Copy to the clipboard, as a Metafile. crtl-W is the shortcut to copy the active graph to the clipboard. They you can go to Work and paste the graph into a document, using either the Paste button or ctrl-V.
In plain R, the plot size depends on the size of the plot window. If your plot is too small or too big, resize the plot window before saving it. You can resize plots in Word, but resizing before saving

usually looks better.