Creative.r: Explanation

Goals of code:

- Display the creativity data (histogram, boxplot, dotplot)
  - As one group or for each treatment
- Calculate summary statistics for each group
- Randomization test
- T-test with equal variances

These notes are meant to be used while you have R open and are executing the code in creativity.r. Each command (or group of related commands) will explained and important parts of the result will be discussed. It will be easiest to use these notes if you execute each command, then read the notes.

I am describing the essential arguments to obtain the sort of results we need for this class. Almost all functions have many more options, specified by additional arguments, that are described in the help files.

**Histogram:** hist(creativity$score)
draws a histogram of the variable named in the function. Not easy to draw side-by-side histograms as in the book.

**Boxplot:** boxplot(creativity$score)
Draws a boxplot. The format follows the description in section 1.5.1, except that “far-away” points, > 3 IQR, are not distinguished from “not-so-far-away” points, > 1.5 IQR.

**Side-by-side Boxplots:** boxplot(split(creativity$score, creativity$treatment))
Functions can be nested, as illustrated here. The split() function splits the first argument (all 47 scores) into groups defined by the value of the second argument (the treatment name). The result of the split function go to boxplot(), which plots the groups side-by-side.

**Dot plots:** stripchart(score~treatment, data=creativity, pch=1)
When there aren’t too many data points, a dot plot portrays the data without summarizing it first. The stripchart() function draws a horizontal dot plot. It
accepts a formula. The left side of the ~ is the response to be plotted; the right side is the grouping variable. The data= tells R where to find the two variables. Further description of data= is in my introduction to R document.

The pch=1 argument changes the plotting character to an open circle. The default for stripchart is pch=0, which is a square box. I like circles better. pch=19 is a filled in circle. There are 25 plot symbols, as well as any character (e.g., pch=’a’). To see the 25 symbols, type plot(1:25, rep(1,25), pch=1:25). If you prefer square boxes, omit the pch= argument.

**Numeric summaries:** summary(), mean(), sd(), var(), IQR() These five functions produce numeric summaries of the data. summary() gives a 6 values: minimum and maximum, 1st quartile, median and 3rd quartile and the average. mean(), sd(), var(), IQR() give the average, standard deviation, variance, and inter-quartile range.

**Numeric summary of multiple groups:**
tapply(creativity$score, creativity$treatment, mean) The tapply() function applies the function you specify (last argument, here mean), to the values in the first argument, categorized by the second argument. The example command will compute the mean for each of the two treatment groups. To get the sd for each group, replace mean with sd. You can use tapply for functions that return more than one value. Ask if you don’t understand the format of that output.

**Randomization p-value:** The code from nobs <- to p.value There is no default function to compute the randomization test p-value. This code uses R’s programming commands to do that. In this example, creativity$score contains the response variable; creativity$treatment contains the grouping variable. These are copied into two new variables, response and group, which are the variables used elsewhere in the code. To use for a new problem, change the variables on the right-hand side of the first two commands, then execute all lines down to and including the p.value line.

The two-sided p-value is printed after you execute the last line.

I am happy to explain the code to anyone who wants to know the details. Note: Those of you familiar with computer programming will realize this code could be
easily converted to a user-defined function. I agree. We’ll define our own functions later.

**Two-sample t-tests**: `t.test(score~treatment, data=creativity, var.equal=T)`

A formula is a convenient way to specify the response and grouping variables. The `var.equal=T` argument specifies that the function should assume that the two groups have the same variance and to use the pooled standard deviation in the t-test. (We’ll talk about this in lecture). That argument is necessary because the default is unequal variance, which is not customary practice (at least in the areas PMD works in).

The output from the t-test function includes:
  - two-sided p-value: `p-value =`
  - (You know it is two-sided because of the next line, which tells you that the alternative is “not equal to 0”.)
  - 95% confidence interval for the difference of means
  - and the sample averages (estimated means) for each group

It is good practice to check that the degrees of freedom (top of output, df=) match what you expect. If not, the data weren’t read or interpreted correctly.

To get something other than a 95% confidence interval, specify the desired coverage as a decimal fraction in the `conf.level=` argument (next command).