Creativity3.r: Explanation

Goals of code:

- Your workspace and R Studio projects
- Calculate two-sample t-test
- Calculating pooled sd and the se of the difference

Note: I only describe what is new. The code file includes preparatory steps (e.g., reading the data file) to make the code standalone. Explanations of reading the data file are in the creativity1 and creativity2 documents.

Retaining work from one R session to another
By default, R offers to save your workspace when you quit. Answering Y is almost always a good idea. This saves all objects (variables, data frames, ...) in your workspace in a file named .Rdata in your current working directory. This means that your progress is saved when you stop work. To resume, all you need to do load that saved workspace.

If you have used setwd() to set the working directory to something specific for your 587 work, the workspace will be saved in that working directory. This creates a problem when R restarts. Initially, the working directory is set to the default directory. On my Windows PC, that default is `c:/Users/pdixon/Documents`. When R loads the saved workspace, it loads the one in there, not the workspace from your 587 folder.

I illustrate two ways to get R to load your 587 working folder. The first (rm, load) works in plain R and RStudio, but many folks suggest never using this way. The preferred way is to create an RStudio project, but that only works in RStudio.

Saving and loading your workspace (both R and R Studio)
`rm(list=ls());  load('.Rdata');`

`setwd('stat 587');` Use the specified folder as the working directory. You will probably need to change the folder name to something appropriate for you. Or, use File / Change dir, or setwd(choose.dir()) to set the appropriate working directory.

choose.dir() isn't available on a Mac (I presume this is an operating system thing, but I don't know the details).

`rm(list=ls())` Remove any automatically loaded objects from the workspace. There may be a .Rdata file in the initial working directory. This clears away everying already loaded. This step is not essential.

`load('.Rdata')` Load the .Rdata file from your new working directory. If you see an error message, that means there isn't anything saved. Not a problem if you starting something new. Is a problem if you expected to have something saved.

At this point, your workspace is restored to its status when you last quit R. If you have loaded any R packages (discussed later), you will need to load any libraries that your code needs.

R Studio projects
R Studio projects simplify saving and loading previous work, especially if you have many simultaneous projects. A project is a special file with a .Rproj extension that identifies a working directory and set of libraries that you want for an analysis (one data set or one course, or even everything you do).

This only applies to R Studio. The equivalent in plain R is saving and loading your workspace.

Creating a project: File / New Project
The popup dialog allows you to create a project in a new working directory, or in a currently existing directory. If you have already done some work in a folder, you want the currently existing directory option. You will have to enter or browse to that working directory. If you are starting something new, choose new directory, then select empty project and provide a name for the new folder (directory) and where to put it in your folder system. R Studio will create an empty folder for your project. (You do not want the new option if you already have stuff in a folder).

You will see various options that may interest you in the future. R Studio makes it easy to link to material stored on GitHub (but not CyBox, yet) and create new R packages and web applications. We will not discuss any of these options.

Using a project:
By default, R Studio saves the last project name and restores that project (working directory, saved workspace, and loaded libraries) when you restart R Studio. If you

quit RStudio while working in your 587 project, this means that your 587 work is automatically restored when you restart RStudio.

You can see the name of the current project by looking in the top right of the entire RStudio window. Look for Project: . If you aren't currently working in a project, it will say Project: (None). The drop-down menu allows you to Close a project, Open a new project, or select a project (after you've worked in a project or two).

If you want to change projects, File / Open Project or File / Open Project in new session will do that. The new session choice opens a second instance of R Studio.

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 response variable is on the left-hand side (score, here) of the $\sim$ and the grouping variable is on the right-hand side (treatment, here). The `data=creativity` specifies the data frame with the variables names in the formula.

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. This is the customary practice, at least in the areas Dr. Dixon works in. The default for t.test is the unequal variance (Welch or Welch-Satterthwaite) t-test.

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".)
with error df and the t-statistic.
95% confidence interval for the difference of means
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).

`t.test(score ~ treatment, data=creativity, var.equal=T, conf.level=0.90)`
will provide a 90% confidence interval for the difference.

Computing the pooled sd and the se of the difference

The t.test() function doesn't report the pooled sd or the se of the difference. Here's how to compute them from a data set. The code assumes that there are only two levels of the treatment variable, but it checks this.

My code is written in the same way as my code for a randomization test. You copy treatment and response information into two specifically-named variables. The rest of the code can be ran without changes.
Aside: Those who know about R functions will see that this code could easily be converted into a function.

```
group <- creativity$treatment
response <- creativity$score
```

Copy the treatment information into the variable named group and the response information into the variable names response.

The rest of the code can be run without modification. My explanations are only relevant if you want to know some useful R tricks.

```
if ...
```

Counts the number of unique values in group. If not 2, stop and write out an error message that includes the names of the groups.

```
n <- table(group)
```

Count the number of observations in each group and save in a vector called n. This will have 2 values, one for each group.

```
s <- tapply(response, group, sd)
```

Calculate the within-group sd for each group. The arguments to tapply are the variable to worked on, the variable defining groups, and the function to apply to all the values in each group. This call computes the sd for each group. Again, the result is a vector with 2 values.

```
pooled.sd <- ...
```

Compute the pooled sd. R allows math on vectors. So `(n-1)*s^2` results in a vector of 2 values, with each being the sample size (n) times the variance (s^2) for each group. `sum()` adds those. The denominator is the total sample size, `sum(n)` minus 2 df.

```
se.diff <- ...
```

Computes the standard error of the difference, either by working separately on each sample size, or by using the vector math equivalent.