t-Tests

William H. Knapp III

March 26, 2016

1 When to Use t-tests

If your design compares two groups of people and the dependent variable is approximately continuous (i.e. can take a wide range of values), the t-test is probably what you need. For example, imagine that you have an experiment with an experimental group and a control group. Further imagine that the dependent variable is how many questions they got right on .

2 Different Types of t-tests

2.1 Independent t-tests

When the two groups of observations are independent from one another (i.e. the participants haven't been paired on any variables), the appropriate test to use is an independent t-test. We'll use the t.test() function for all of our t-tests, but we'll see how changing the information that we give to that function can enable us to perform different types of t-tests. Of course we should set our working directory to the location that has the data. Next we need to read in our data.

```
> dat<-read.csv("example5.csv")
```

Of course, it's always a good idea to take a look at the structure of the data.

```
> str(df)
```

```
'data.frame':	50 obs. of  2 variables:
 $ group: Factor w/ 2 levels "control","experimental": 2 2 2 2 2 2 2 2 2 2 ...
 $ rt   : num  273 321 294 321 271 ...
```

As we can see, we have a data frame of 50 observations. The important variables are group, which has two levels: experimental and control, and rt (i.e. response time), which indicates how much time it took the participants to perform the task.

To perform an independent t-tests we use the following template. The t-test assumes that the variances between the two groups are equal, so we'll include that in our template.

```
t.test(group1,group2,var.equal=TRUE)
```

So let's perform a t-test to compare our experimental and control groups.

```
> t.test(df$rt[df$group=="experimental"],
+        df$rt[df$group=="control"],
+        var.equal=TRUE)

        Two Sample t-test

data:  df$rt[df$group == "experimental"] and df$rt[df$group == "control"]
t = -3.1478, df = 48, p-value = 0.002826
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -34.356624  -7.574029
sample estimates:
mean of x mean of y 
 289.5517  310.5170 
```

We can see from the test that the two groups differ. To see the direction of the difference let's get the means for the experimental and control conditions.

```
> mean(df$rt[df$group=="experimental"])

[1] 289.5517

> mean(df$rt[df$group=="control"])

[1] 310.517
```

With this information, we can conclude that the individuals in the experimental condition completed the task faster than those in the control condition, t(48) = -3.15, p = .0028.

2.2 Paired t-tests

Again, we start with two groups of observations of some variable. If your participants have been matched on some variable (e.g. IQ) and the pairs were randomly assigned to the different conditions, or if you observe the same participants in multiple conditions (i.e. you're using a repeated measures design), you should perform a paired t.test. The template for the paired t-test is as follows:

```
t.test(group1,group2,var.equal=TRUE,paired=TRUE)
```

So lets pretend that the data we had earlier represent paired data. If you take a look at how I generated the data in Data Generation.r, you'll see that I created the data to be paired. Thus, we'll be able to see how including this information in our t-test will make our test more powerful (i.e. we'll observe a smaller p-value).

```
> t.test(df$rt[df$group=="experimental"],
+        df$rt[df$group=="control"],
+        var.equal=TRUE,
+        paired=TRUE)

        Paired t-test

data:  df$rt[df$group == "experimental"] and df$rt[df$group == "control"]
t = -8.0147, df = 24, p-value = 3.055e-08
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -26.36419 -15.56646
sample estimates:
mean of the differences
              -20.96533
```

From the results of the analysis, we can see that the paired t-test was much more powerful than it's independent brother. That said, you can only use this test if the observations were actually paired in some way. So let's draw our conclusions and back them up with the appropriate statistical statement. Individuals in the experimental condition completed the task more quickly than individuals in the control condition, t(24) = -8.01, $p < .05$.

Some of you might be confused by the notation for the p-value which indicates that the p-value is 3.055e-8. The e-08 means we'd need to shift the decimal point to the left by 8. Thus 3.055e-8 really represents .00000003055, which is highly significant. When the p-value is less than .0001, just use p < .05 instead of reporting the exact p-value.

2.3 One and Two-tailed t-tests

Depending on your null hypothesis you can run a one- or two-tailed t-test (independent or paired). With two groups there are three potential null hypothesis statements.

1. The two groups are equal (two-tailed)
2. Group one is less than or equal to group 2 (one-tailed)
3. Group one is greater than or equal to group 2 (one-tailed)

The alternative hypotheses for these null hypotheses are as follows:

1. The two groups are not equal
2. Group one is greater than group 2
3. Group one is less than group 2

When performing two-tailed t-tests, we don't have to do anything special to our t.test() statement. However, for our one-tailed tests we need to specify the alternative. You can specify three different alternatives: "two.sided," "less," or "greater." The default is two.sided, which is why we don't have to do anything special for our two-tailed tests.

So let's pretend that our null hypothesis is that group one is less than than or equal to group two. If you remember that group one was actually less than group two, you should expect that the results of our t-test will be insignificant. To keep things simple, we'll just use the unpaired t-test to examine how specifying different tails affects significance.

```
> t.test(df$rt[df$group=="experimental"],
+        df$rt[df$group=="control"],
+        var.equal=TRUE,
+        alternative="greater")
```

```
        Two Sample t-test

data:  df$rt[df$group == "experimental"] and df$rt[df$group == "control"]
t = -3.1478, df = 48, p-value = 0.9986
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 -32.13603       Inf
sample estimates:
mean of x mean of y 
 289.5517  310.5170 
```

From this t-test, we can see that the p-value is very close to one. Thus, we should conclude that group one is less than or equal to group two, t(48) = -3.15, p = .999. Notice how this p-value is much less than the p-value we observed for the two-tailed independent t-test. Here we fail to reject the null, when we would have rejected it in the two-tailed situation.

Now let's run a one-tailed test under the null hypothesis that group one is greater than or equal to group two.

```
> t.test(df$rt[df$group=="experimental"],
+        df$rt[df$group=="control"],
+        var.equal=TRUE,
+        alternative="less")

        Two Sample t-test

data:  df$rt[df$group == "experimental"] and df$rt[df$group == "control"]
t = -3.1478, df = 48, p-value = 0.001413
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
      -Inf -9.794621
sample estimates:
mean of x mean of y 
 289.5517  310.5170 
```

Here' you can see that with an appropriate directional hypothesis, the results are more significant. Specifically, group one completed the task faster than group two, t(48) = -3.15, p = .0014.

We can also run directional tests on paired data, but I'll leave that as an exercise for the readers.

3 Graphing the Results

It's said that a picture is worth a thousand words. Thus, we should create a graph for any significant finding we uncover. To do this, we'll use several different packages that you'll need to install and load. To install packages, we use the following template:

```
install.packages("packagename")
```

You only need to install the packages once and it will be available to you from that point on anytime you ask R to load the package. To load the package we use the following template:

```
library(packagename)
```

One note, often when installing packages, you'll see red text in the console that looks like errors. However, if at the end of the installation process you see that the package was successfully installed, you're good to go. You can check that the package was successfully installed by trying library(packagename). Again you might see some errors that various functions or variables are masked. This is normal, so don't worry. If you're returned to the command prompt without an indication that the package wasn't found, you should be good to go.

So let's install the packages that you'll need to complete this, and the remainder of the assignments so we can create pretty graphs.

```
> install.packages("ggplot2")
> install.packages("gplots")
> install.packages("dplyr")
```

Although you only need to install the packages once. You'll have to load the packages anytime you reopen RStudio to start working on your homework. Let's load the packages now.

```
> library(ggplot2)
> library(gplots)
> library(dplyr)
```

We'll use ggplot2 to create figures. We'll use gplots to specify colors more easily. Finally, we'll use dplyr to group our data and create summary statistics (e.g. the means and standard errors) for our data that we'll use to create our plots with ggplot2.

ggplot2 is pretty cool, it creates figures by adding and manipulating layers. If you're familiar with layers in photoshop or other programs this should be pretty easy to understand. Let's start with a bare-bones plot and then add and edit layers to make a plot that's informative and visually appealing.

One important bit of information. Although I told you that you should delete any > or + symbols at the beginning of the lines when copying R code, you should keep the + symbols at the end of the lines. These tell R that we're adding more layers to our plot or modifying existing ones. If you omit these plus signs at the end of lines, ggplot2 will think that you're done giving it information and will stop where it is.

Additionally, we'll assign the output of our ggplot2 work into a variable. I like using names like f, which stands for figure, but you can use whatever variable names you like.

Before we get started, there's one more thing that I'd like to mention. You can create comments in R by using the # sign. Thus, as I'm including the R code in a code chunk, you'll see that I explain some of what I'm doing using R comments. Hopefully, this will help you to better understand exactly what the code is doing.

Ok, let's get started. Before we can create our figures we need to calculate the means and standard errors of the means for our figures. We'll use dplyr to do this. Dplyr is pretty cool, because it allows you to chain together various commands that will be completed in sequence. We'll assign the output of the work completed with dplyr into a variable called temp (you could use other names, but temp works for me). Let's create the code we need to calculate the means and standard errors of the means for our two conditions and then break it down.

```
> temp<-df%>%group_by(group)%>%
+   summarize(means=mean(rt),
+             sems=sd(rt)/sqrt(length(rt)))
```

In plain English what this statment says is use the dataframe df, group the data frame by the group variable, and summarize the different groups by creating means and standard errors of the means for the different groups and assigning these values to a new variable called temp.

We can see that temp has summarized the data by calling temp in the console.

```
> temp

Source: local data frame [2 x 3]

         group    means     sems
        (fctr)    (dbl)    (dbl)
1      control 310.5170 4.271169
2 experimental 289.5517 5.110365
```

Now that we have our data in summarized form, we can use this information to create our plots. The plots we create will all begin with an assignment statement for a ggplot2 command.

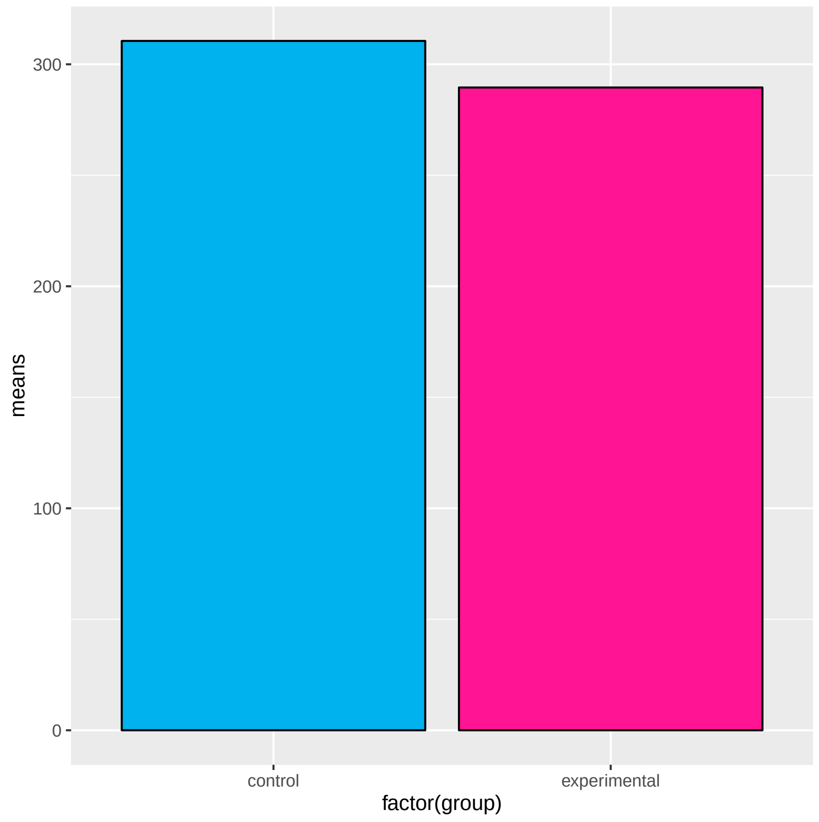
```
> f<-ggplot(temp,aes(x=factor(group),y=means))+
+   geom_bar(stat="identity", color="black",
+             fill=c("deepskyblue2", "deeppink"))
```

The first line tells R to create a ggplot using the temp date. “aes” is short for aesthetic, which tells ggplot2 roughly how we want the figure to look. Here we specfify which data are to be plotted on the x and y axes. Notice how we included a plus sign at the end of the first line. This tells R that we’re going to add or edit more layers. If you get rid of this plus, your work won’t look like mine.

The second line says that we want to use a bar plot. By indicating stat=”identity”, we’re saying use the raw values from the means to create the bar graph. Here color indicates the color we want the bars to be outlined in. Finally, fill indicates what colors we want the bars to be.

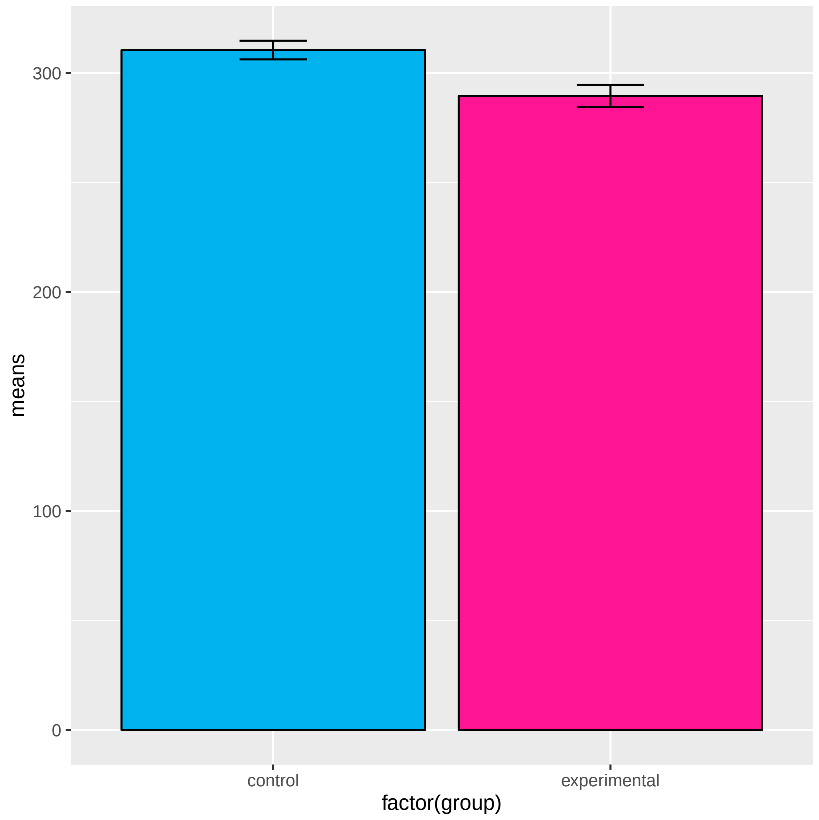
To see what our figure looks like thus far we can just call f like any other variable in R.

```
> f
```



Now let's add some error bars to our figure and see what we get.

```
> f<-f+geom_errorbar(aes(ymax=means+sems,
+                        ymin=means-sems),
+                    width=.2)
> f
```



Let's give the figure a meaningful title. *NOTE*: I'm going to show you the commands one by one, but I won't replot the figure until we're finished. If you want to take a look at what the figure looks like after any of the steps, type "f" into the console after you make a change. First I'm going to add a title to the figure. This title is fairly generic, because I didn't really describe the differences in the two groups. In a real-world project their should be a much more meaningful title.

```
> f<-f+ggtitle("Response Times For the Groups")
```

Let's change the axis labels.

```
> f<-f+labs(x="Group", y="Response Time (ms)")
```

Now let's change names of our groups on the x-axis.

```
> f<-f+scale_x_discrete(breaks=c("control","experimental"),
+                        labels=c("Control","Experimental"))
```

Let's change the title size, font, and vertical adjustment.

```
> f<-f+theme(plot.title=element_text(size=15,
+                                    face="bold",
+                                    vjust=.5))
```

Let's do the same thing for the axis labels and text.

```
> f<-f+theme(axis.title.x=element_text(size=12,
+                                      face="bold",
+                                      vjust=-.25))+
+   theme(axis.title.y=element_text(size=12,
+                                      face="bold",
+                                      vjust=.25))+
+   theme(axis.text.x=element_text(size=10,
+                                      face="bold",
+                                      color="black"))+
+   theme(axis.text.y=element_text(size=10,
+                                      face="bold",
+                                      color="black"))
>
```

Let's change the limits of the y-axis so we can more clearly see the effects. I piced values that would contain the means plus and minus two times the standard error of the means.

```
> f<-f+coord_cartesian(ylim=c(min(temp$means)-2*max(temp$sems),
+                              max(temp$means)-2*max(temp$sems)))
```

Let's make the axes look a little prettier. Remember you can call f in the console after you make any of these changes to see what the figure now looks like.

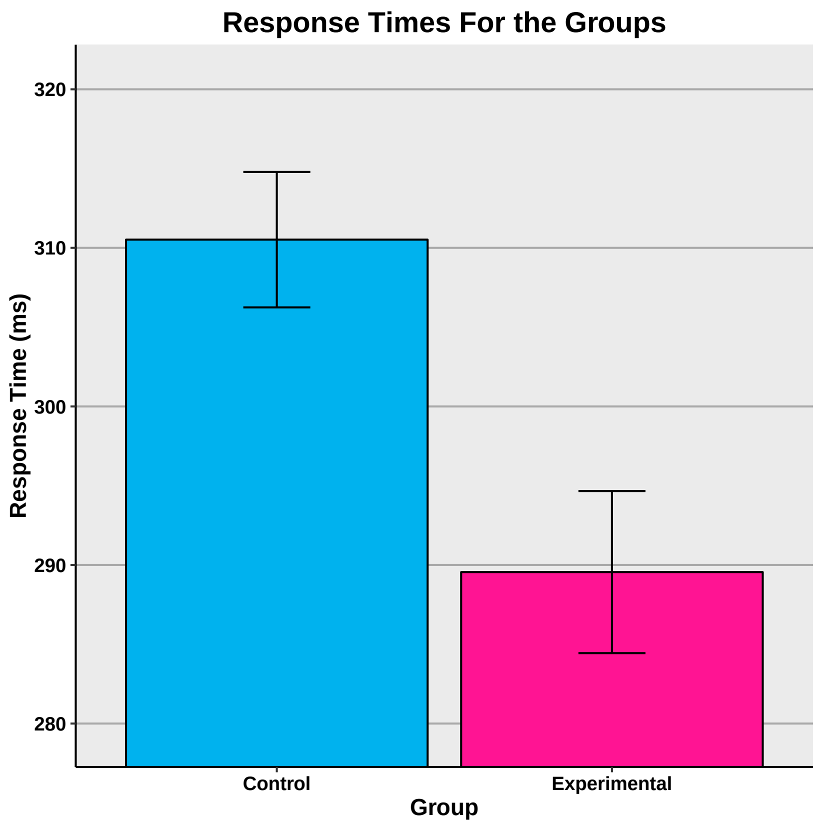
```
> f<-f+theme(panel.border=element_blank(),
+            axis.line=element_line())
```

Now let's clean up that grey grid.

```
> f<-f+theme(panel.grid.major.x=element_blank())+
+   theme(panel.grid.major.y=element_line(color="darkgrey"))+
+   theme(panel.grid.minor.y=element_blank())
```

Before we plot the figure, let's put everything together to see how creating a figure in one feel swoop would look.

```
> f<-ggplot(temp,aes(x=factor(group),y=means))+
+   geom_bar(stat="identity", color="black",
+            fill=c("deepskyblue2", "deeppink"))+
+   geom_errorbar(aes(ymax=means+sems,
+                   ymin=means-sems),
+                width=.2)+
+   ggtitle("Response Times For the Groups")+
+   labs(x="Group", y="Response Time (ms)")+
+   scale_x_discrete(breaks=c("control","experimental"),
+                   labels=c("Control","Experimental"))+
+   theme(plot.title=element_text(size=15,
+                                 face="bold",
+                                 vjust=.5))+
+   theme(axis.title.x=element_text(size=12,
+                                   face="bold",
+                                   vjust=-.25))+
+   theme(axis.title.y=element_text(size=12,
+                                   face="bold",
+                                   vjust=.25))+
+   theme(axis.text.x=element_text(size=10,
+                                  face="bold",
+                                  color="black"))+
+   theme(axis.text.y=element_text(size=10,
+                                  face="bold",
+                                  color="black"))+
+   coord_cartesian(ylim=c(min(temp$means)-2*max(temp$sems),
+                         max(temp$means)+2*max(temp$sems)))+
+   theme(panel.border=element_blank(),
+         axis.line=element_line())+
+   theme(panel.grid.major.x=element_blank())+
+   theme(panel.grid.major.y=element_line(color="darkgrey"))+
+   theme(panel.grid.minor.y=element_blank())
> f
```


Response Times For the Groups
Response Time (ms)
320
310
300
290
280
Control
Experimental
Group