Week 9 R tutorial

Aaron Shaw and Jeremy Foote

November 8, 2020 (revised)

Contents

Analyzing continuous data with t-tests and ANOVA

t-tests

You learned the theory/concepts behind t-tests last week, so here's a brief run-down on how to use built-in functions in R to conduct them and interpret the results. As a reminder, t-tests are used to estimate the difference-in-means between two continuous distributions. You might be working with a one-sample test, two (independent samples, pooled or paired samples, or difference scores. R can handle most of this stuff and, if not, remember that you know how to write your own functions!

The usual procedure for conducting t-tests in R is pretty straightforward: you use the `t.test()` function. For my example, I'll simulate two groups by randomly sampling the `rivers` dataset. I will then conduct a two-sample t-test for difference of means under a null hypothesis of no difference.

```
set.seed(20201107)
group.a <- sample(rivers, 35, replace=TRUE)
group.b <- sample(rivers, 42, replace=TRUE)

t.test(group.a, group.b)
```

```
##
##  Welch Two Sample t-test
##
## data:  group.a and group.b
## t = 1.0931, df = 46.296, p-value = 0.28
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -99.30366 335.42747
## sample estimates:
## mean of x mean of y
##  573.9429  455.8810
```

Notice everything that is contained in the output: It tells me what test I just ran (a two sample t-test). It also tells me the names of the two groups. Then it reports the test statistic (t) value, the degrees of freedom, and a p-value. It states my alternative hypothesis explicitly and includes a 95% confidence interval before also reporting the sample means for each group (which you can use to directly calculate your point estimate).

Go read the documentation for `t.test()` in R as the function has many possible arguments to help you conduct exactly the sort of t-test you want given the particulars of your data, hypotheses, etc. The documentation will also tell you quite a lot about the output or "values" returned by the function. As with most tests and operations in R, you can store the output of a `t.test()` call as an object. In the code below, notice what values the object has and how you might use the object to call specific values later.

```
t.result.ab <- t.test(group.a, group.b)

t.result.ab ## same as before
```

```
##
##  Welch Two Sample t-test
##
## data:  group.a and group.b
## t = 1.0931, df = 46.296, p-value = 0.28
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -99.30366 335.42747
## sample estimates:
## mean of x mean of y
##  573.9429  455.8810
```

```
names(t.result.ab)
```

```
##  [1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
##  [6] "null.value"  "stderr"      "alternative" "method"      "data.name"
```

```
t.result.ab$conf.int
```

```
## [1] -99.30366 335.42747
## attr(,"conf.level")
## [1] 0.95
```

You never need to calculate a confidence interval for a difference in means by hand again. You can even set other α values in the call to t-test if you want to estimate another interval (see the documentation).

ANOVAs

Here's a very brief introduction to how you can perform an ANOVA in R. Let's use the `iris` dataset (look up `help(iris)` to learn about it) to build an example in which we test whether irises of different species have different average sepal width. Remember that the purpose of an ANOVA is to compare some continuous outcome across two or more categories to test the global hypothesis of any differences between groups.

In the code block below, I explicitly store the `iris` dataset and then call `aov()` to conduct the test. In the call to `aov()` I indicate the variables and dataset I want to use. Notice the use of the tilde character ("~"). The tilde indicates that you're providing R some information as a "formula." This particular command tells R that I'm using the variables before and after the tilde as the dimensions of my ANOVA. The `aov()` function follows a convention that the dependent or outcome variable has to go first (on the left-hand side of the tilde) in the formula (just try it the other way if you want to see what happens — since ANOVAs assume a continuous outcome it won't work to try and use the categorical measure as the DV).

```
iris <- iris

aov(Sepal.Width ~ Species, data=iris)
```

```
## Call:
##    aov(formula = Sepal.Width ~ Species, data = iris)
##
```

```
## Terms:
##                   Species Residuals
## Sum of Squares  11.34493  16.96200
## Deg. of Freedom        2       147
##
## Residual standard error: 0.3396877
## Estimated effects may be unbalanced
```

Hmm, wait a minute. That doesn't look quite right. Where's the F statistic? Where's the p-value? Why doesn't this table look like the ANOVA tables the book showed us?

Not to worry. As with many statistical tests in R, the `aov()` function did everything it promised, you just need to ask it explicitly to show you the "summary" of the test to get the information you're looking for. You can also store the results and call `summary()` as a separate command.

```
summary(aov(Sepal.Width ~ Species, data=iris))
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## Species       2  11.35   5.672   49.16 <2e-16 ***
## Residuals   147  16.96   0.115
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
aov1.result <- aov(Sepal.Width ~ Species, data=iris)

summary(aov1.result)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## Species       2  11.35   5.672   49.16 <2e-16 ***
## Residuals   147  16.96   0.115
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

That should look more familiar. It's good to get accustomed to storing your model results and then calling `summary()` on the output. Such a routine is typical for many other types of statistical tests in R (such as regression models).

Note that the object returned by `aov()` has values that you can also call directly just like we did with the output of `t.test()` above. In the example below, I've asked R to show me the residuals (deviation from the sample mean across all groups) for each row of the dataset. I'm not sure what you might use this for, but it's available.

```
names(aov1.result)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "contrasts"     "xlevels"       "call"          "terms"
## [13] "model"
```

```
aov1.result$residuals
```

```
##      1      2      3      4      5      6      7      8      9     10     11
##  0.072 -0.428 -0.228 -0.328  0.172  0.472 -0.028 -0.028 -0.528 -0.328  0.272
##     12     13     14     15     16     17     18     19     20     21     22
## -0.028 -0.428 -0.428  0.572  0.972  0.472  0.072  0.372  0.372 -0.028  0.272
##     23     24     25     26     27     28     29     30     31     32     33
##  0.172 -0.128 -0.028 -0.428 -0.028  0.072 -0.028 -0.228 -0.328 -0.028  0.672
##     34     35     36     37     38     39     40     41     42     43     44
##  0.772 -0.328 -0.228  0.072  0.172 -0.428 -0.028  0.072 -1.128 -0.228  0.072
```

```
##     45     46     47     48     49     50     51     52     53     54     55
##  0.372 -0.428  0.372 -0.228  0.272 -0.128  0.430  0.430  0.330 -0.470  0.030
##     56     57     58     59     60     61     62     63     64     65     66
##  0.030  0.530 -0.370  0.130 -0.070 -0.770  0.230 -0.570  0.130  0.130  0.330
##     67     68     69     70     71     72     73     74     75     76     77
##  0.230 -0.070 -0.570 -0.270  0.430  0.030 -0.270  0.030  0.130  0.230  0.030
##     78     79     80     81     82     83     84     85     86     87     88
##  0.230  0.130 -0.170 -0.370 -0.370 -0.070 -0.070  0.230  0.630  0.330 -0.470
##     89     90     91     92     93     94     95     96     97     98     99
##  0.230 -0.270 -0.170  0.230 -0.170 -0.470 -0.070  0.230  0.130  0.130 -0.270
##    100    101    102    103    104    105    106    107    108    109    110
##  0.030  0.326 -0.274  0.026 -0.074  0.026  0.026 -0.474 -0.074 -0.474  0.626
##    111    112    113    114    115    116    117    118    119    120    121
##  0.226 -0.274  0.026 -0.474 -0.174  0.226  0.026  0.826 -0.374 -0.774  0.226
##    122    123    124    125    126    127    128    129    130    131    132
## -0.174 -0.174 -0.274  0.326  0.226 -0.174  0.026 -0.174  0.026 -0.174  0.826
##    133    134    135    136    137    138    139    140    141    142    143
## -0.174 -0.174 -0.374  0.026  0.426  0.126  0.026  0.126  0.126  0.126 -0.274
##    144    145    146    147    148    149    150
##  0.226  0.326  0.026 -0.474  0.026  0.426  0.026
```

As with all functions in R, I *strongly* encourage you to read the documentation before you set out to conduct your own ANOVAs in the wild. There are details and settings that it may be important to consider. In addition, you might notice in the documentation for ANOVAs that R actually conducts the ANOVA by performing a linear regression (whaaaat??!?!?). Turns out that ANOVA is just a special case of linear regression with some idiosyncratic assumptions and standards for the type of summary information that needs to be reported. No need to worry about the details of that for now, but it can be good to have in the back of your mind for future reference in case you find yourself in a situation where you had planned to run an ANOVA, but your data just doesn't fit the assumptions necessary to identify the model. (Also, this is the point where Nick Vincent will want to remind you that most statistical tests can, in some way or another, be represented/understood as a special case of a linear model).

Multiple comparisons

Alright, you've run your ANOVA or whatever and now you want to compare a bunch of different group means against each other. How do you avoid issues with multiple comparisons? The two approaches you've read about so far (the Bonferroni Correction and the Benjamini-Hochberg procedure) are not difficult to conduct by-hand, but R also has a handy `pairwise.t.test()` function that can take care of this for you.

You can (and should) read the documentation for `pairwise.t.test()` yourself, but to get you started here's a simple example of how it works. Let's imagine that I wanted to perform all possible pairwise comparisons between mean sepal width across the different species of irises included in the ANOVA analysis I just conducted above. Since there are three different species, that results in ($\binom{3}{2} = 3$) possible pairs (remember those binomial coefficients from a few weeks ago? I knew you would!). I can use the `with()` command to provide the dataframe and then call `pairwise.t.test()` across the same dimensions that I used to conduct my ANOVA. I can also provide a number of specific arguments to `pairwise.t.test()` including one for `p.adjust.method`, which helps manage corrections for multiple comparisons. In the example below, I've demonstrated how you can use the Bonferroni and Benjamini-Hochberg corrections by supplying the appropriate arguments to `p.adjust.method`. If you want to learn more, you can also check out the documentation for another function called `p.adjust()` that offers additional correction methods.

```
with(iris,
     pairwise.t.test(Sepal.Width, Species, p.adjust.method = "bonferroni"))
```

```
##
##  Pairwise comparisons using t tests with pooled SD
```

```
##
## data:  Sepal.Width and Species
##
##            setosa  versicolor
## versicolor < 2e-16 -
## virginica  1.4e-09 0.0094
##
## P value adjustment method: bonferroni
```

```
with(iris,
     pairwise.t.test(Sepal.Width, Species, p.adjust.method = "BH"))
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  Sepal.Width and Species
##
##            setosa  versicolor
## versicolor < 2e-16 -
## virginica  6.8e-10 0.0031
##
## P value adjustment method: BH
```

The three values in the little matrices there correspond to the p-values for the pairwise differences of means. Depending on your alpha value, all of these differences seem to meet standard thresholds for statistical significance, indicating that we can reject the null hypothesis of no differences in every case. (The example is a little crude/simple since there are only three groups, the bar would get higher with additional groups to compare!).