Lab 6 Tutorial

Janette Avelar & Anwesha Guha
1/25/2022

One-Way ANOVA

Intro

This week we will go over one-way analysis of variance and post-hoc pairwise comparisons.

Note: **All material is taken from Dr. Zopluoglu's 2021 EDUC 614 materials and adapted for this year's course.**

We have adapted it to slide format using R.

Want to learn how we created R slide presentations? This tutorial *is helpful.*

Importing Data

Our first step is to import the data we need. We'll use the `Absence` dataset for this walkthrough, which you can access via Canvas. We'll also use the `factor()` and `str()` functions to ensure the data is in the proper format to run an ANOVA.

```
abs <- read.csv(file   = "/Users/janetteavelar/Desktop/absence.csv",
                header = TRUE)
```

Our dataset consists of two columns, with `absence` indicating the number of absences and `class` indicating the grade the individual is in. We'll thus want to convert the `class` column into a factor with 4 levels to specify the class level and aid our analysis.

```
abs$class <- factor(abs$class,
                    levels = c(1, 2, 3, 4),
                    labels = c("freshman", "sophomore", "junior", "senior"))
```

Data Prep

Next, we'll double-check the structure of our data to ensure `class` is a factor with four specified levels and `absence` is an integer…

```
str(abs)
```

```
## 'data.frame':    72 obs. of  2 variables:
##  $ class  : Factor w/ 4 levels "freshman","sophomore",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ absence: int  4 3 2 4 5 2 1 0 2 1 ...
```

…and we'll familiarize ourselves with the data by looking at the first five lines of the dataset.

```
head(abs, 5) #5 = number of rows
```

```
##      class absence
## 1 freshman       4
## 2 freshman       3
## 3 freshman       2
## 4 freshman       4
## 5 freshman       5
```

Research Hypothesis

Suppose you have a hypothesis that students in different class levels have different average number of absences. Using verbal language, we'd write our null and alternative hypothesis as follows:

- **Null hypothesis:** The average number of absences are equal for all class levels.
- **Alternative hypothesis:** There is at least one class level with the average number of absences different than the other class levels.

Remember that the same null hypothesis can all be written using symbolic notation:

$$H_0 : \mu_{freshman} = \mu_{sophomore} = \mu_{junior} = \mu_{senior}$$

Data Checks

Our next step is to check the independent variable. It doesn't matter if it's a factor or numeric variable, the function will work for either type. If the independent variable is an integer, the functions will treat the numbers as distinct categorical variables. For example:

```
typeof(abs$class)
```

```
## [1] "integer"
```

```
table(abs$class)
```

```
##
##  freshman sophomore    junior    senior
##        18        18        18        18
```

We also want to make sure our dependent variable is a numeric variable.

```
typeof(abs$absence)
```

```
## [1] "integer"
```

Notice this returns `integer`, which indicates that R recognizes it as a numeric variable.

It's also a good idea to check the descriptive statistics for the variable to make sure it looks as expected. We'll take a look at the dependent variable first:

```
require(psych)
```

```
describe(x = abs$absence)
```

```
##    vars  n mean  sd median trimmed  mad min max range skew kurtosis  se
## X1    1 72 3.33 1.7      3    3.28 1.48   0   7     7 0.16    -0.66 0.2
```

We'll also want to look at the conditional descriptive statistics by group:

```
describeBy(x = abs$absence,
           group = abs$class)
```

```
##
##  Descriptive statistics by group
## group: freshman
##    vars  n mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 18 2.67 1.46    2.5    2.69 2.22   0   5     5 0.01    -1.18 0.34
## ------------------------------------------------------------
## group: sophomore
##    vars  n mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 18 2.72 1.56      3    2.75 1.48   0   5     5    0    -1.33 0.37
## ------------------------------------------------------------
## group: junior
##    vars  n mean   sd median trimmed  mad min max range  skew kurtosis   se
## X1    1 18 2.78 1.06      3    2.81 1.48   1   4     3 -0.14    -1.44 0.25
## ------------------------------------------------------------
## group: senior
##    vars  n mean   sd median trimmed  mad min max range  skew kurtosis   se
## X1    1 18 5.17 1.25      5    5.19 1.48   3   7     4 -0.12    -1.05 0.29
```

Conducting one-way ANOVA using **aov()**

One-way ANOVA

There are different ways of running a one-way ANOVA using R. We'll first explore the `aov()` function from base R.

The formula argument is similar to what we've seen in the past. It's composed of 3 parts:
* our dependent variable `absence` on the left side. * our independent grouping variable `class` on the right side. * the two variables are separated by `~`, which signifies our argument to R.

Don't forget to also specify the data argument with the name of the data object in your environment. This argument tells R where to find the variables you've specified in the formula argument.

```
aov(formula = absence ~ class,
    data = abs)


## Call:
##    aov(formula = absence ~ class, data = abs)
##
## Terms:
##                     class Residuals
## Sum of Squares   80.77778 123.22222
## Deg. of Freedom         3        68
##
## Residual standard error: 1.346139
## Estimated effects may be unbalanced
```

You'll notice in the output that `aov()` does not include all the numbers we want to see from the ANOVA analysis. Thus, a better method to extract these numbers is to assign `aov()` to an object, which we'll call `mod` for Model 1.

```
mod <- aov(formula = absence ~ class,
           data = abs)
```

We'll then use `summary()` on the new object to pull up more convenient output.

```
summary(mod)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## class        3  80.78  26.926   14.86 1.54e-07 ***
## Residuals   68 123.22   1.812
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Reading the output

The `summary()` output has two rows. The first row will be labeled after our independent variable `class` and will always represent what we've referred to in our lectures as `Between Groups`.

The second row labeled `Residuals` will always represent what we've referred to as `Within Groups`.

Please refer to the lecture notes and textbook for more details about how these numbers are computed and what they mean.

Conducting one-way ANOVA using `Anova()`

Anova()

We're going to explore another function, `Anova()`, which may be more convenient in the future when you start learning about more complex ANOVA designs with multiple independent variables. The `Anova()` function comes from the `{car}` package.

To use `Anova()`, we'll create an object using the `aov()` function like before. Rather than using the `summary()` function to generate our output, we'll use `Anova()` instead and specify the type of ANOVA test we want. In this class, we will not learn about two-factor or three-factor ANOVA, which would require a specific number for the `type` argument, thus, you can leave `type = 3` to specify one-way ANOVA.

For this example, I'll just use the `mod` object we created earlier, but note that you will specify the name of the object you want when using the function in the future.

```
require(car)
```

```
Anova(mod, type = 3)
```

```
## Anova Table (Type III tests)
##
```

```
Anova(mod, type = 3)
```

```
## Anova Table (Type III tests)
##
## Response: absence
##              Sum Sq Df F value    Pr(>F)
## (Intercept) 128.000  1  70.637 4.038e-12 ***
## class        80.778  3  14.859 1.538e-07 ***
## Residuals   123.222 68
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Reading the output

In this output, you can ignore the first line labeled `(Intercept)`.

The second and third lines are identical to what we obtained before with the `summary()` function.

Calculating effect sizes

You can compute Cohen's *f* by extracting the information from the `aov()` object. First assign the summary to an object so we can extract the values we'll need:

```
s.mod <- summary(mod)[[1]]
```

We'll then extract the values we need to plug into a formula, including degrees of freedom, the F statistic, and N, the number of observations:

```
df = s.mod[1,]$Df #assigns degrees of freedom
F = s.mod[1,]$`F value` #assigns F value
N = sum(s.mod$Df)+1 #assigns N value
```

Finally, we'll plug everything into a formula that will give us a value for Cohen's *f*:

```
sqrt(F*df/N)
```

```
## [1] 0.7868458
```

Calculating variance explained

You can also compute a value for η^2 in a similar method. After assigning the summary to an object `s.mod` (which we created on the last slide), we'll directly plug the values we need into the following formula that will give us a value for η^2:

```
s.mod[1,2]/sum(s.mod[,2])
```

```
## [1] 0.3959695
```

Summarizing the findings

Finally, you can summarize the results using a paragraph, like the one below:

A one-way ANOVA was conducted to test whether or not there is a significant difference in the number of absences between freshman, sophomore, junior, and senior students. The findings indicated that there was a significant difference among the four groups in average number of absences, $F(3, 68) = 14.86$, $p < 0.001$, Cohen's $f = 0.787$. The class level explained 39.6% of the variance in the number of absences.

Conducting post-hoc pairwise comparisons

Pairwise comparisons using `pairwise.t.test()`

When you find a significant result from the overall one-way ANOVA, we reject the null to argue that there is at least one mean that is different than the others at the population level. That is, that there is a significant difference between the groups we are observing.

The second step after finding a significant result is to run post-hoc comparisons to get more insights about where the difference may lie. We'll use the `pairwise.t.test()` function from base R to run our post-hoc pairwise comparisons.

Structuring the argument

The `pairwise.t.test()` function has three arguments we need to specify:

- the first, `x = abs$absence`, is the column for the dependent variable from the data object.
- the second, `g = abs$class`, is the column for the independent grouping variable from the data object.
- the third, `p.adjust.method = "bonferroni"` indicates that we want to use Bonferroni procedure to adjust our p-values for multiple comparisons. You can check other methods you can use to adjust p-values with this argument by running `?p.adjust`.

It may be helpful to turn off scientific notation for reporting values, so we'll do that first:

```
options(scipen = 99)
```

Now we can run our pairwise comparisons with the specified arguments:

```
pairwise.t.test(x = abs$absence,
                g = abs$class,
                p.adjust.method = "bonferroni")
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  abs$absence and abs$class
##
##           freshman  sophomore junior
## sophomore 1         -         -
## junior    1         1         -
## senior    0.0000028 0.0000046 0.0000074
##
## P value adjustment method: bonferroni
```

Reading the output

Our pairwise comparison output returns a grid of p-values for all possible pairwise comparisons. For instance, this grid indicates the adjusted p-values for comparison between our groups as follows:

- *Sophomore & Freshman:* adjusted p-value is **1**; mean difference **is not** statistically significant.
- *Junior & Freshman:* adjusted p-value is **1**; mean difference **is not** statistically significant.
- *Junior & Sophomore:* adjusted p-value is **1**; mean difference **is not** statistically significant.
- *Senior & Freshman:* adjusted p-value is **0.0000028**; mean difference **is** statistically significant ($p < .05$).
- *Senior & Sophomore:* adjusted p-value is **0.0000046**; mean difference **is** statistically significant ($p < .05$).
- *Senior & Junior:* adjusted p-value is **0.0000074**; mean difference **is** statistically significant.

Pairwise comparisons using `TukeyHSD()`

Note that the `pairwise.t.test()` function only returns the adjusted p-values, and nothing else. This may not be convenient, as you have to go back and compute the mean difference between each pair of groups by hand after looking at the descriptive statistics for reporting purposes.

Another alternative is to use the `TukeyHSD()` function from base R. Tukey's Honest Significant Difference method is just another procedure for adjusting p-values like the Bonferroni procedure we used in the previous example, though there are many other procedures we can use.

For this function, you only have to provide the object obtained from the `aov()` function earlier. That is, the `mod` object we created for Model 1. We'll assign the function to another object, named `pairwise` in order to specify we want the output saved as a data frame.

```
pairwise <- as.data.frame(TukeyHSD(mod)$class)
```

We can then pull up our new object:

```
pairwise
```

```
##                          diff       lwr      upr           p adj
## sophomore-freshman 0.05555556 -1.126228 1.237339 0.999314953402
## junior-freshman    0.11111111 -1.070672 1.292895 0.994617062447
## senior-freshman    2.50000000  1.318216 3.681784 0.000002771067
## junior-sophomore   0.05555556 -1.126228 1.237339 0.999314953402
## senior-sophomore   2.44444444  1.262661 3.626228 0.000004497222
## senior-junior      2.38888889  1.207105 3.570672 0.000007270206
```

Reading the output

Notice that the output provided by `TukeyHSD()` gives us additional information for each pairwise comparison:

- The `diff` column reports the mean difference between groups.
- The `lwr` and `upr` columns report the upper and lower bounds for the 95% confidence interval for the mean difference.
- The `p adj` column reports the adjusted p-value.

Also note that the adjusted p-values from the previous Bonferroni procedure are very similar and close to those we get using Tukey's HSD procedure.

Calculating effect sizes

You can calculate Cohen's *d* effect sizes for each pairwise comparison by dividing the mean difference for each group by the square root of within-group variance. Rather than doing this by hand, we'll have R calculate each value for us.

First, we'll extract the output from our summary just like when we were calculating Cohen's *f* previously:

```
MS.within <- summary(mod)[[1]][2,3]
```

Next, we'll create a new column in our `pairwise` object that runs the formula for each comparison group:

```
pairwise$cohen.d <- pairwise$diff/sqrt(MS.within)
```

Now you can call the `pairwise` object to see the new column we just created:

```
pairwise
```

```
##                          diff       lwr      upr           p adj    cohen.d
## sophomore-freshman 0.05555556 -1.126228 1.237339 0.999314953402 0.04127028
## junior-freshman    0.11111111 -1.070672 1.292895 0.994617062447 0.08254056
## senior-freshman    2.50000000  1.318216 3.681784 0.000002771067 1.85716267
## junior-sophomore   0.05555556 -1.126228 1.237339 0.999314953402 0.04127028
## senior-sophomore   2.44444444  1.262661 3.626228 0.000004497222 1.81589239
## senior-junior      2.38888889  1.207105 3.570672 0.000007270206 1.77462211
```

Summarizing the results

Finally, you can summarize the results using a second paragraph to follow the results of your one-way ANOVA, like the one below:

The significant main effect for class type was further analyzed using Tukey's HSD procedure to examine all pairwise contrasts. The contrasts tests indicated that there was a significant difference in the number of absences between freshman and senior students (Mean difference = 2.50, 95% CI [1.32,3.68], Cohen's d = 1.86, $p < 0.001$), between sophomore and senior students (Mean difference = 2.44, 95% CI [1.26,3.63], Cohen's d = 1.82, $p < 0.001$), and between junior and senior students (Mean difference = 2.39, 95% CI [1.21,3.57], Cohen's d = 1.78, $p < 0.001$). All other pairwise contrasts were found to be insignificant (see Table XX).