# Complete Example (Between Subjects – 1 IV only)

A health psychologist recorded the number of close inter-personal attachments of 45-year-olds who were in excellent, fair, or poor health. People in the Excellent Health group had 4, 3, 2, and 3 close attachments; people in the Fair Health group had 3, 5, and 8 close attachments; and people in the Poor Health group had 3, 1, 0, and 2 close attachments.

**Datafile:** bn 1 anova.csv

**IVs:**

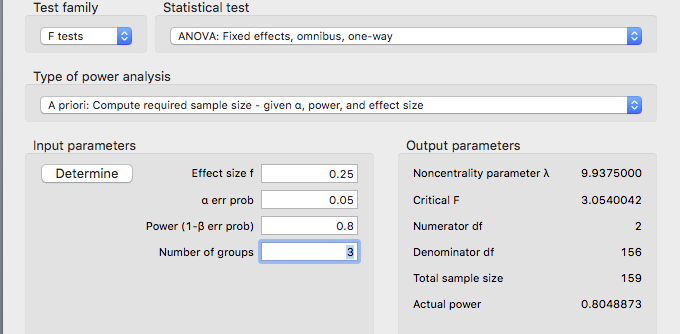
Health group – excellent, fair, poor

**DV:**

Friends - number of close inter-personal attachments

**Power:**

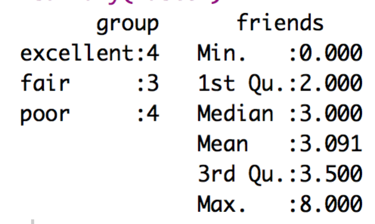
1. Open Gpower!
   1. Test family: F test
   2. Statistical test: ANOVA: fixed one way
   3. Pick an effect size or calculate
   4. Alpha = .05
   5. Power (1-beta .20) = .80
   6. Groups – number of levels.
   7. Hit ok!
2. Says we needed to run 159 people to find a significant effect with a medium effect size.
   1. Pick an effect size or calculate – hit determine 🡪 effect size from variance 🡪 direct, enter eta squared, calculate and transfer to main. Here I left it as a medium effect size.
   2. Alpha = .05
   3. Power (1-beta .20) = .80
   4. Groups 3 for excellent, fair, and poor.

****

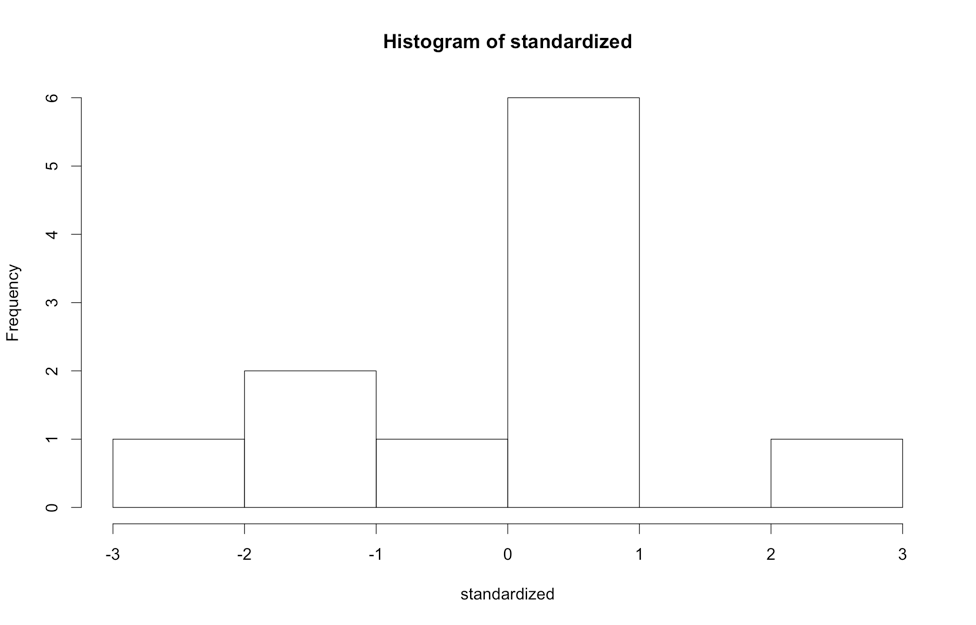
1. Check your learning:
   1. If we have a small effect size (eta square = small, use your notes to find this value), you should get a very large number of estimated participants.
   2. Can you get N = 957?

**Data screening:**

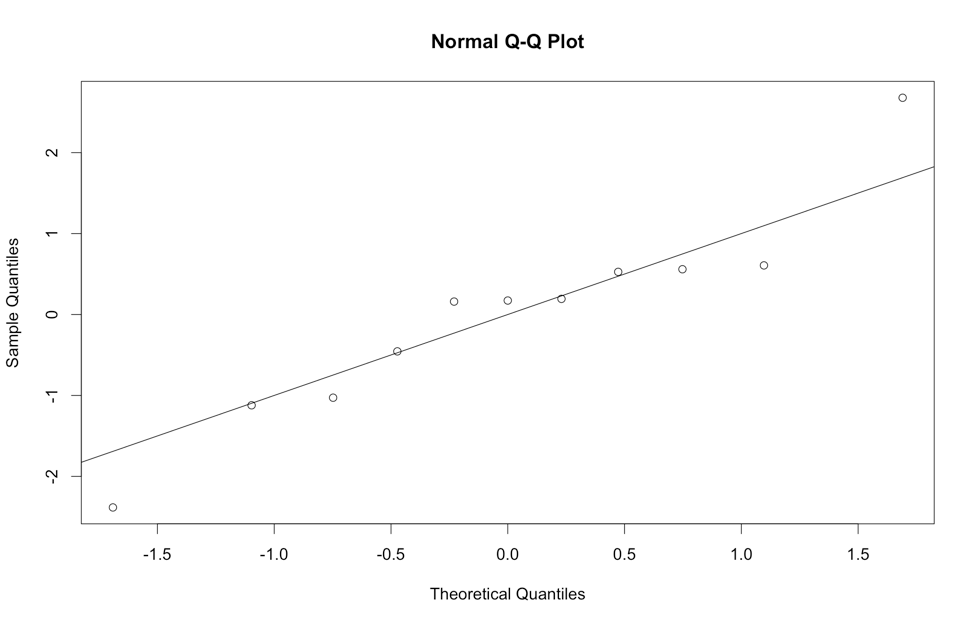
1. Accuracy:
   1. Use the summary(*dataset name*) function to get the basic information for the data.
   2. Everything here appears fine:
      1. No weird groups.
      2. No negative friends, no real maximum expected.



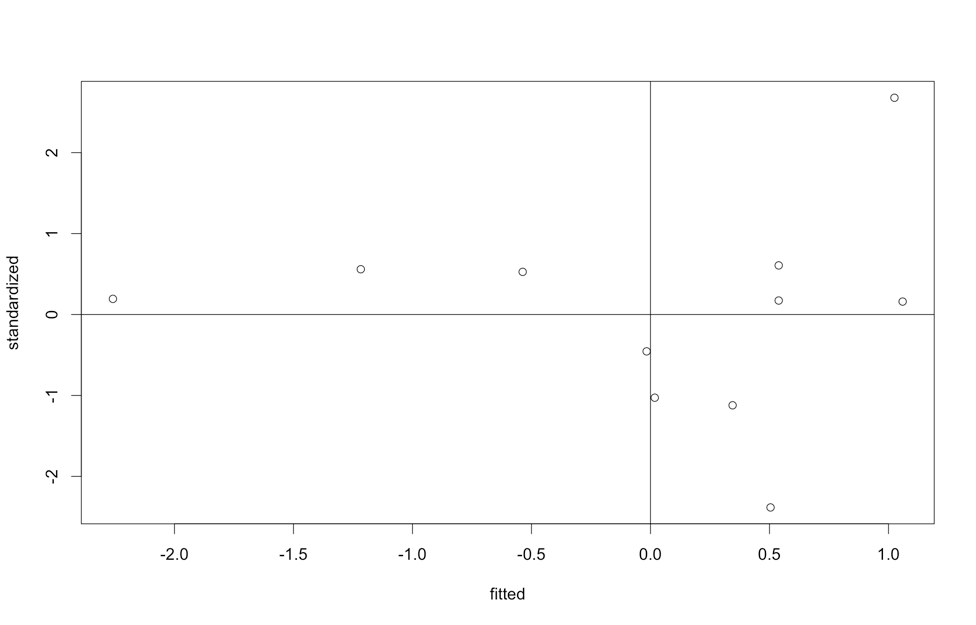
1. Missing:
   1. With the summary function, I can also see that I don’t have any missing data, because there are no NA values shown. Therefore, I can skip the missing data step.
2. Outliers:
   1. One main issue with univariate (that means one DV) between-subjects ANOVA, is that we have only one column to screen for outliers since the rest are assigned by the experimenter (that’s the IV).
   2. Therefore, we cannot use Mahalanobis because it requires there be at least two columns.
   3. Instead, we will use the scale() function to create Z-scores.
      1. Code: zscore = scale(*dataset$column*)
      2. That will save you a set of z scores for the data. Z scores are the same idea as Mahalanobis – the distance from the center, but now they are the distance from the mean of the one variable you are working with.
   4. What cut off should you use? It’s always the absolute value of 3, which is the *p* < .001 cut off for z-scores.
   5. In this data set, we shouldn’t have any outliers, remember that FALSE is bad.
      1. summary(abs(zscores) < 3)
   6. But we can exclude them in the same way we did for Mahalanobis.
      1. noout = subset(notypos, abs(zscore) < 3)
      2. The abs(*column*) creates the absolute value of z score, so you don’t forget to exclude the negatives.
      3. This code keeps all the people who are less than 3.
3. Assumptions:
   1. Additivity: not necessary, since we have only one continuous variable.
   2. Set up for the rest of the assumptions:
      1. Make a random variable:
         1. random = rchisq(nrow(*dataset*), 7)
      2. Run a fake regression:
         1. fake = lm(random~., data = *dataset*)
      3. Create the standardized residuals:
         1. standardized = rstudent(fake)
      4. Create the fitted values:
         1. fitted = scale(fake$fitted.values)
   3. Normality:
      1. hist(standardized)
      2. Eh? This graph isn’t great – the data is centered close to zero, but the real issue is the lack of people to help fill in to determine if it’s normal or not.



* 1. Linearity:
     1. qqnorm(standardized)
     2. abline(0,1)
     3. This graph looks pretty good.



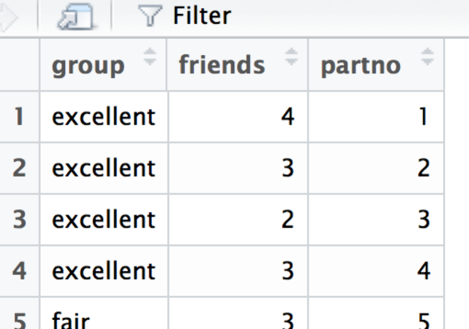
* 1. Homogeneity:
     1. plot(fitted,standardized)
     2. abline(0,0)
     3. abline(v = 0)
     4. Here the data is not so hot – it’s really one data point that’s throwing off my graph. Otherwise, the spread is ok.
     5. Now, most people do not talk about homoscedasticity for ANOVA, because homogeneity sort of equals homoscedasticity when one variable is categorical, and the other is continuous (aka the ANOVA set up).
     6. If we did, it would be bad though, mostly because of the spread on the right side of my graph.



* 1. Homogeneity: Take 2 Levene’s Test
     1. Levene’s is a test for homogeneity between groups, so it looks to see if the variances are equal across your IV levels.
     2. It is notoriously **oversensitive**, but can be a good place to start if you want to check a real number, rather than this scatterplot.
     3. With large sample sizes, it is often significant (remembering the big important rule, p<.001), and with large sample sizes it matters less. Ergo, if you have big *n* in each group, then don’t worry about it so much.
     4. You will have to run the ANOVA to get Levene’s Test, see below.

**Running the ANOVA:**

1. Load the ez library.
   1. library(ez)
2. Add a participant number to your data if it does not have one.
   1. The ez package requires a participant number, so we will have to add one.
   2. *dataset$partno* = 1:nrow(*dataset*)



1. Run the ANOVA (all these lines):
   1. ezANOVA(data = *dataset*,

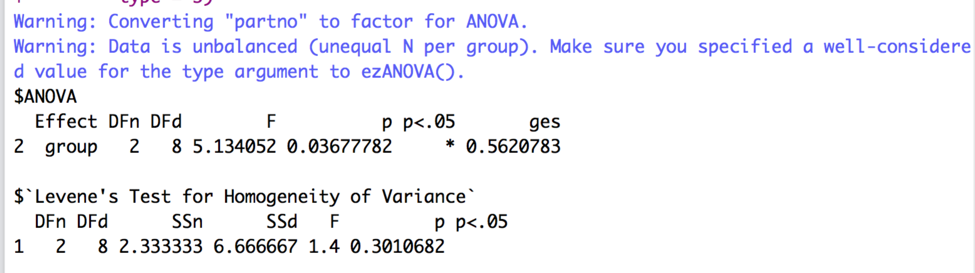
wid = partno,

between = *column of IV,*

dv = *column of DV*,

type = 3)

* 1. I got a warning, but that’s because my groups do not have the same number of people in them (that’s ok).
  2. The $ANOVA part is the ANOVA output.
  3. The $Levene’s is the Levene’s output.

****

1. Interpret the output:
   1. Check Levene’s for Homogeneity – especially if your residual plot was not evenly distributed. You want p > .001.
      1. Look at the last number under *p*, it says .30, which is greater than .001, so we are ok.
   2. Check the Omnibus (overall) test for your IV:
      1. Under effect it says *group* – that’s because my IV is named group. Therefore, it will say whatever the IV name is.
      2. The DFn = df numerator or model.
      3. The DFd = df denominator or error.
      4. F = F
      5. p = p value.
      6. p < .05 helpfully tells you if it’s significant at *p* < .05, which is what we want to find.
      7. ges = generalized least squares or η2.
      8. It’s significant yay!
      9. Write that up:
         1. APA: *F*(2,8) = 5.13, *p* = .04, η2 = .56.
         2. AMA: *F*2,8 = 5.13, *p* = .04, η2 = .56.
2. If the overall test is significant, you will have to run post hocs to figure out what happened.
   1. First, I find it easiest to create a table to figure out my effects – and what is being compared. Remember that we are going to calculate each *pairwise* combination, which is every mean compared to every other mean.
   2. To get the means and SDs, we can use tapply.
      1. tapply(*dataset$DV*, list(*dataset*$*IV*), mean)
      2. tapply(*dataset$DV*, list(*dataset*$*IV*), sd)

excellent fair poor

3.000000 5.333333 1.500000

excellent fair poor

0.8164966 2.5166115 1.2909944

* 1. Let’s put that into a table.

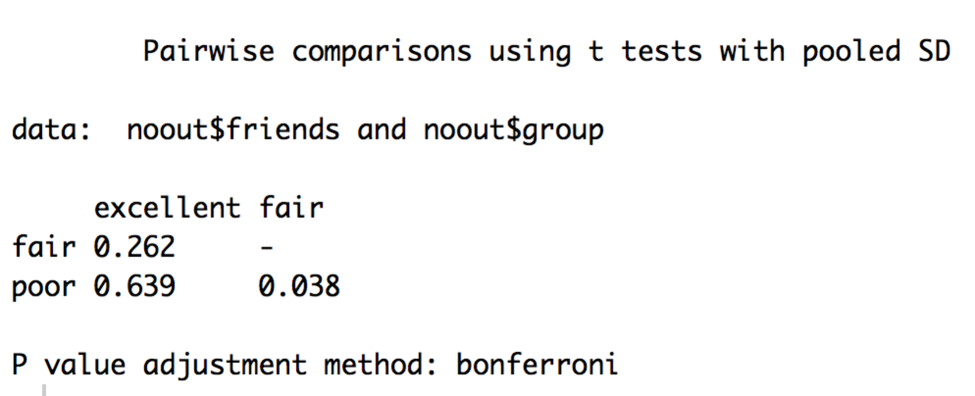
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mean 1 | Mean 2 | P-value | Explain? | Effect size |
| Excellent *M* = 3.00, *SD* = .82 | Fair *M* = 5.33, *SD* = 2.52 |  |  |  |
| Excellent *M* = 3.00, *SD* = .82 | Poor *M* = 1.50, *SD* = 1.29 |  |  |  |
| Fair *M* = 5.33, *SD* = 2.52 | Poor *M* = 1.50, *SD* = 1.29 |  |  |  |

* 1. Now, we have to calculate the *post hoc test* and *post hoc correction* to find out what’s going on. Is 3 friends different than 5 friends? Is 3 friends different from 1.5 friends? Etc.
  2. Use the pairwise.t.test() function to run t.test you learned earlier on all groups at once.
     1. Remember, you use paired = F for independent t-tests, which is what we want to use for between-subjects ANOVA.
     2. Independent t-tests require var.equal = T.
     3. P.adjust.method is the *correction*.
     4. pairwise.t.test(*dataset*$*DV*, *dataset*$*IV*,

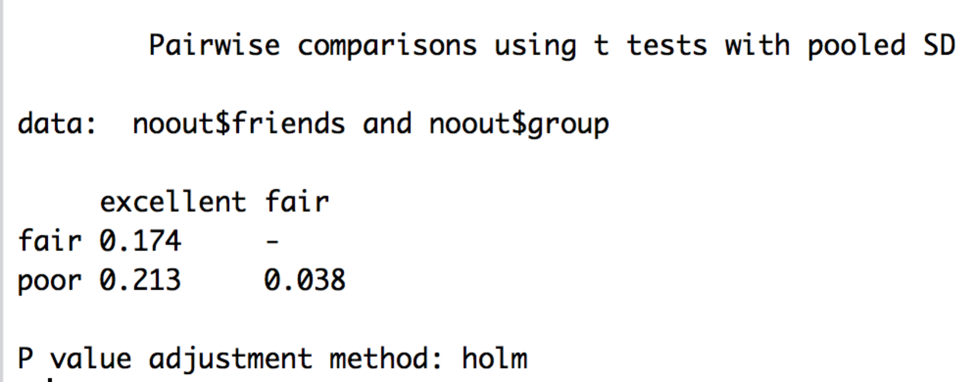
paired = F,

var.equal = T,

p.adjust.method = "bonferroni")



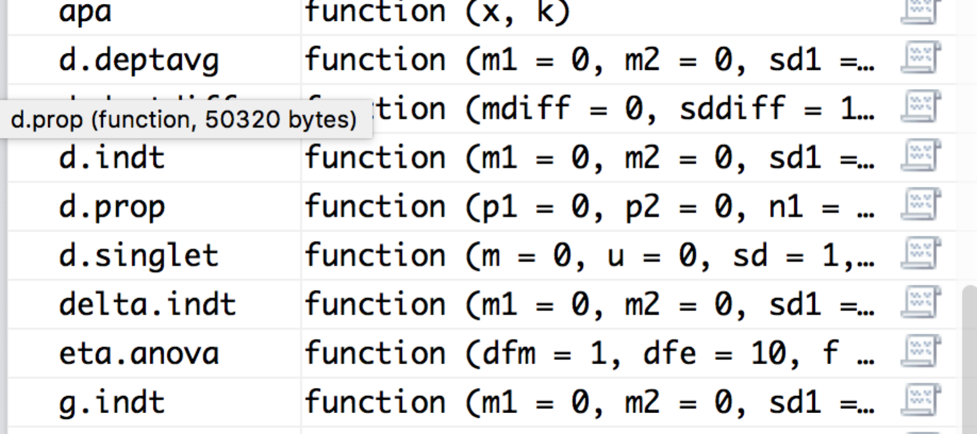
* 1. Check your learning here – try running the pairwise t-test WITHOUT the bonferroni correction.
     1. Just take out the last line and comma.



* 1. You can see that the p values are different – what Bonferroni does is change the p values biased on the number of tests you are running. That’s good for us, because then we can use p<.05 again to determine if it is significant.
  2. Use the Bonferroni output to fill in your p-values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mean 1 | Mean 2 | P-value | Explain? | Effect size |
| Excellent *M* = 3.00, *SD* = .82 | Fair *M* = 5.33, *SD* = 2.52 | .26 | Not significant, groups are equal |  |
| Excellent *M* = 3.00, *SD* = .82 | Poor *M* = 1.50, *SD* = 1.29 | .64 | Not significant, groups are equal |  |
| Fair *M* = 5.33, *SD* = 2.52 | Poor *M* = 1.50, *SD* = 1.29 | .04 | Significant,  Fair has more friends than poor |  |

* 1. You can use MOTE to calculate the effect sizes OR the R script Dr. B just wrote!
     1. Load the effsize.R script and run the whole thing, so that you get new functions listed in the Environment window.



* 1. We will use d.indt for these calculations (yes, three times).
     1. d.indt(m1 = #, m2 = #, sd1 = #, sd2 = #, n1 = #, n2 = #, a = .05, k = 2)
     2. We have means and SDs in our table.
     3. You can get n with the tapply function as well – change mean/sd to length.
     4. Here’s the output you should get:

M1 = 3.00, SD = 0.82, SE = 0.41, 95%CI[1.70 - 4.30]

M2 = 5.33, SD = 2.52, SE = 1.45, 95%CI[-0.93 - 11.59]

t(5) = -1.78, p = 0.14, d = -1.36, 95%CI[-3.01 - 0.40]

M1 = 3.00, SD = 0.82, SE = 0.41, 95%CI[1.70 - 4.30]

M2 = 1.50, SD = 1.29, SE = 0.64, 95%CI[-0.55 - 3.55]

t(6) = 1.96, p = 0.10, d = 1.39, 95%CI[-0.24 - 2.93]

M1 = 1.50, SD = 1.29, SE = 0.64, 95%CI[-0.55 - 3.55]

M2 = 5.33, SD = 2.52, SE = 1.45, 95%CI[-0.93 - 11.59]

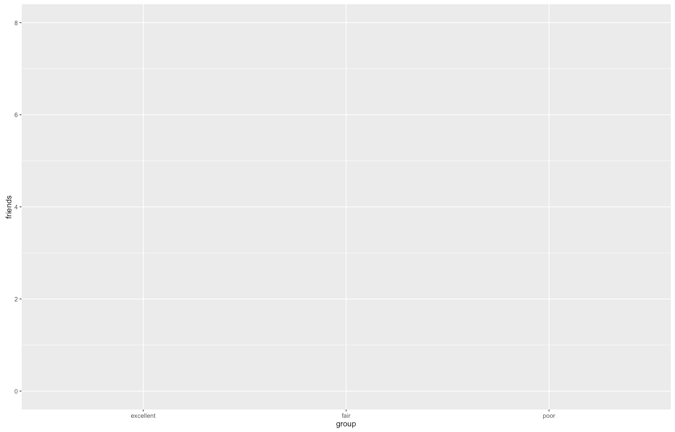
t(5) = -2.67, p = 0.04, d = -2.04, 95%CI[-3.91 - -0.05]

* 1. Make sure each M, SD, and N look correct.
  2. Enter *d* only into your table. You get *t*-values, but those are *uncorrected*, so we don’t want to use them.
  3. You can make *d* values positive or negative – I tend to report them as always positive because the negative just indicates that you subtracted the smaller mean first, not anything about the actual effect size.
  4. Even though some of our tests were not significantly different, they do have big effect sizes, which indicates maybe we should run more people for the power to find that significant difference.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mean 1 | Mean 2 | P-value | Explain? | Effect size |
| Excellent *M* = 3.00, *SD* = .82 | Fair *M* = 5.33, *SD* = 2.52 | .26 | Not significant, groups are equal | 1.36 |
| Excellent *M* = 3.00, *SD* = .82 | Poor *M* = 1.50, *SD* = 1.29 | .64 | Not significant, groups are equal | 1.39 |
| Fair *M* = 5.33, *SD* = 2.52 | Poor *M* = 1.50, *SD* = 1.29 | .04 | Significant,  Fair has more friends than poor | 2.04 |

**Graphs:**

1. The best type of chart for anything analyzing group means is a bar chart with error bars.
2. We are going to use ggplot2 to build all our graphs.
   1. The package works like a transparency machine – you build layers and add them to the graph. You will really want to learn to stack your code, so that it’s easy to troubleshoot any problems you have.
3. First, load the ggplot2 library.
   1. library(ggplot2).
4. Create a blank graph with the right variables.
   1. X = IV, Y = DV.
   2. bargraph = ggplot(*datasetname,* aes(*Xcolumn, Ycolumn*))
   3. Check that it worked – try running just bargraph. You should get a blank plot like this:



1. Add things to the plot:

bargraph +

stat\_summary(fun.y = mean,

geom = "bar",

fill = "White",

color = "Black") +

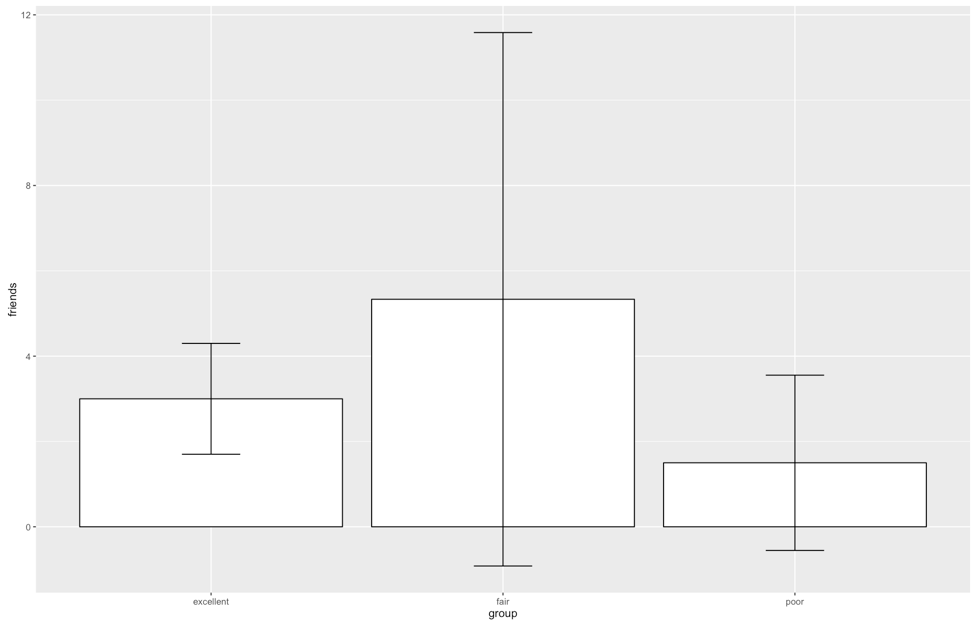
stat\_summary(fun.data = mean\_cl\_normal,

geom = "errorbar",

position = position\_dodge(width = 0.90),

width = 0.2)

* 1. Please note:
     1. That code above stays exactly the same, but remember that “” doesn’t copy correctly sometimes.
     2. What does it do?
        1. The first stat\_summary adds the bars to the graph by graphing the mean for each group.
        2. The second stat\_summary adds the error bars of the confidence interval (approximately 2\*SE). These bars help you see how much the variance is spread around each group.
     3. You should have this now:



* 1. That is the right graph, but it is **hideous.**
  2. First, we are going to clean up the gray background, the nondiscriminate axes, and the tiny type.
  3. Separate from the graph code, run this code exactly:

theme = theme(panel.grid.major = element\_blank(),

panel.grid.minor = element\_blank(),

panel.background = element\_blank(),

axis.line = element\_line(colour = "black"),

legend.key = element\_rect(fill = "white"),

text = element\_text(size = 15))

* 1. This code saves a whole bunch of settings as theme, which then we can add to our graph.
  2. NOTE: In this demo, we are walking through one part at a time, but you will run the entire graph code again to recreate the graph. It isn’t quite cool enough to remember what you did a minute ago.

bargraph +

stat\_summary(fun.y = mean,

geom = "bar",

fill = "White",

color = "Black") +

stat\_summary(fun.data = mean\_cl\_normal,

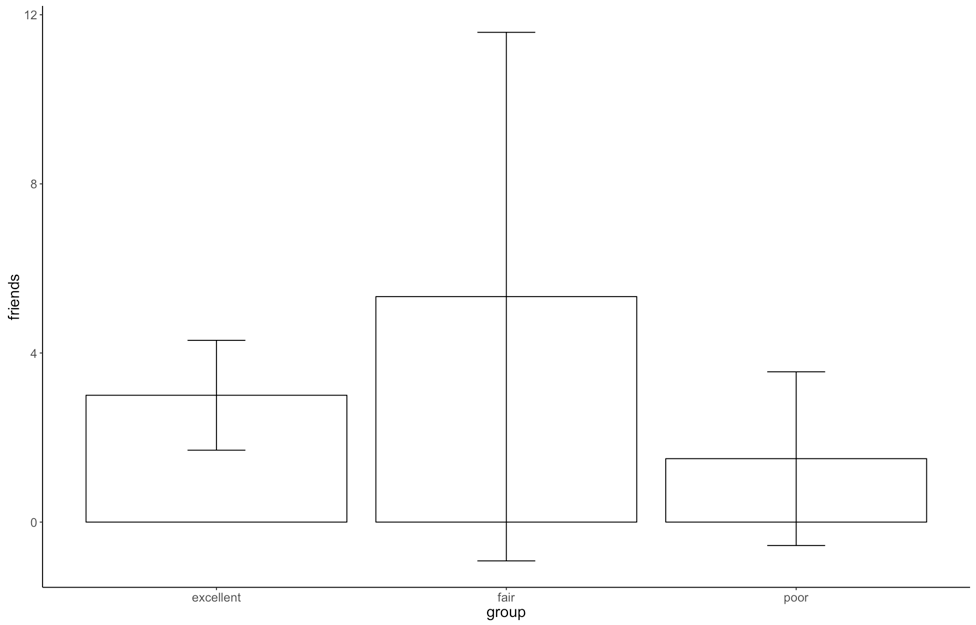
geom = "errorbar",

position = position\_dodge(width = 0.90),

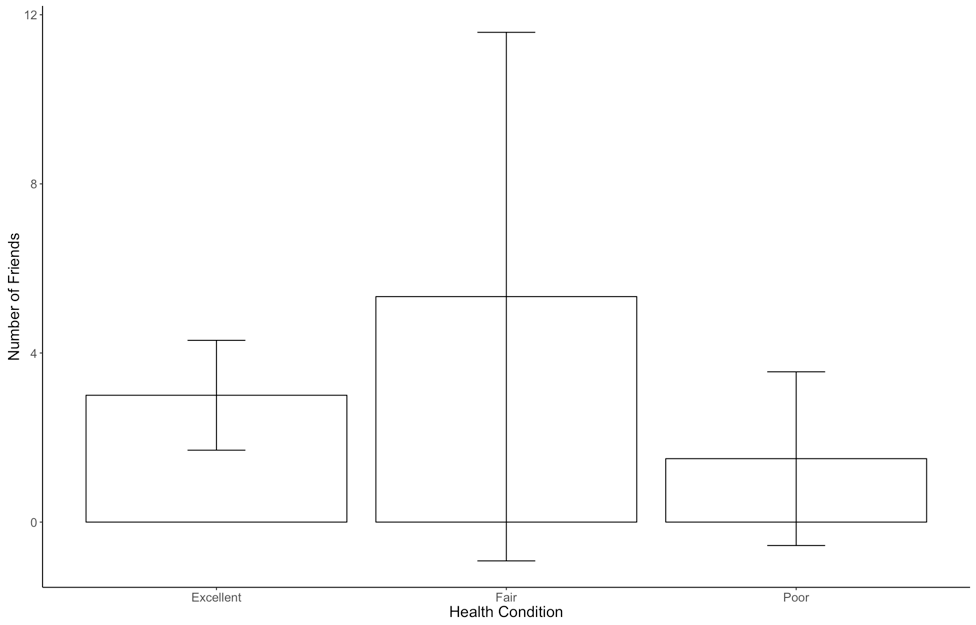
width = 0.2) +

theme

Should give you:



* 1. Ok, that’s better, but now we have two issues:
     1. The x and y axes labels are terrible – what do they even mean?
     2. The group labels are not capitalized.
  2. How to fix that:
     1. xlab(“Text that you want”) + ylab(“Text that you want”) will fix the axes labels.
     2. scale\_x\_discrete(labels = c(“Label1”, “Label2”, …)) will add labels – be sure to type them in the same order you have your bars currently – it does not rearrange, just relabels.
  3. If you run all the code from before and that code, you should get:

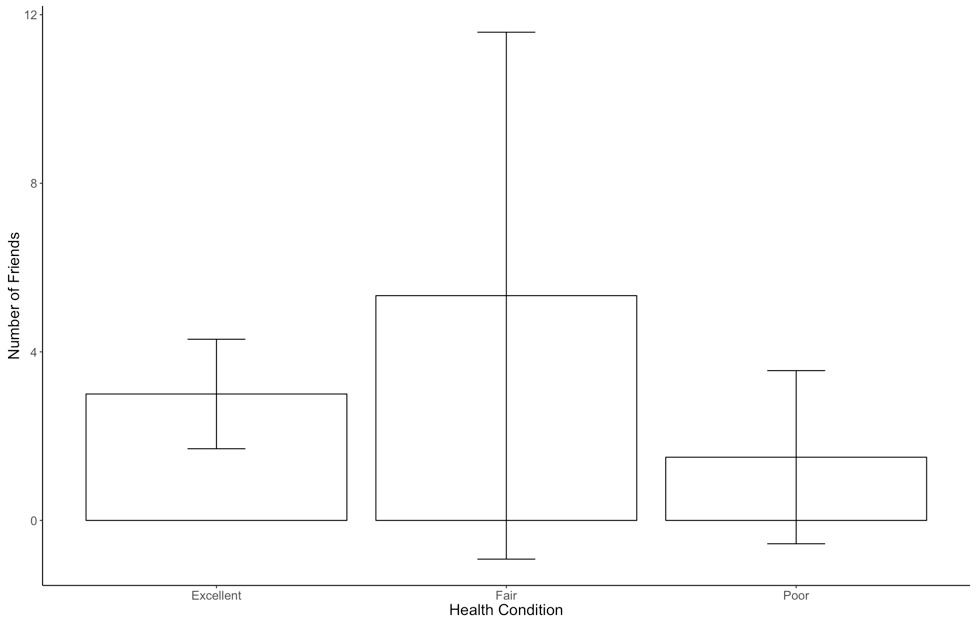


* 1. Now, to be fair, we really should reorder the groups as Poor – Fair – Excellent, but we’ll save that adventure for another lecture.

Example write up:

**Results**

Participants’ numbers of inter-personal connections were examined in relationship to their overall health. The data was screened for assumptions and outliers and found to be satisfactory (Levene’s *F*(2,8) = 1.40, *p* = .30). A between subjects ANOVA was used to analyze the number of inter-personal attachments with excellent, fair, and poor health groups. These group means were found to be significantly different, *F*(2,8) = 5.13, *p* = .04, η2 = .56. An independent t-test with Bonferroni correction was used to examine differences between groups. As shown in Figure 1, the excellent health group (*M* = 3.00, *SD* = 0.82) was not different from the fair health group (*M* = 5.33, *SD* = 2.52, *p* = .19, *d* = 1.36). The excellent and poor health groups (*M* = 1.50, *SD* = 1.29) were also found to have equal numbers of inter-personal attachments (*p* = .41, *d* = 1.39). However, the fair health group was found to have significantly more attachments than the poor health groups (*p* = .03, *d =* 2.04).



*Figure 1.* Average number of friends by health condition with confidence interval error bars.