Assignment 6

Instructions (the same as for Assignment 1): Make an R Notebook and in it answer the two questions below (one Notebook for both questions). When you are done, hand in on Quercus the *output* from Previewing (or Knitting) your Notebook. Do *not* hand in the Notebook itself. You want to show that you can (i) write code that will answer the questions, (ii) run that code and get some sensible output, (iii) write some words that show you know what is going on and that reflect your conclusions about the data. Your goal is to convince the grader that you *understand* what you are doing: not only doing the right thing, but making it clear that you know *why* it's the right thing.

Do *not* expect to get help on this assignment. The purpose of the assignments is for you to see how much *you* have understood. You will find that you also learn something from grappling with the assignments. The time to get help is after you watch the lectures and work through the problems from PASIAS, via tutorial and the discussion board. The only reason to contact the instructor while working on the assignments is to report something missing like a data file that cannot be read.

You have 3 hours to complete this assignment after you start it.

My solutions to this assignment, with extra discussion, will be available after everyone has handed in their assignment.

1. A programming course, Comp Sci 101, can be taken either in-person, by attending a class at fixed days and times, or online, by doing sessions that can be taken at times the student chooses. The course coordinator wants to know whether students taking the course in these two different ways learn a different amount, as measured by their scores on the final exam for the course. This example comes from the before-times, so the final exam was taken in person by all students. The final exam was out of 45 marks. A total of 18 students took part in the study. Each student was allowed to choose the section they preferred. The data are in http://ritsokiguess.site/STAC32/proggo.csv.

   Write a report of a complete and appropriate analysis of these data. Your report should include a description of the data in your own words, any necessary pre-processing steps, appropriate graphs, statistical analysis, assessment of assumptions for your preferred analysis, and a statement of conclusions. Imagine that your report will be read by the department Chair, who does not know about this study, and who still remembers some of their first-year Statistics course.

   Solution:

   My report follows. The occasional appearance of "(see note x)" from time to time indicates some extra discussion, at the end. Ignore those on your first read-through. Your report doesn't have to be the same as mine, but it should be complete and coherent, and readable by someone with a first course in Statistics.

   ## Introduction

   Do students learn programming more or less effectively from an online course, completed on their own time, compared with a regular in-person lecture course that meets at the same times every week? In Comp Sci 101, a study was carried out in which 18 students, 9 each in the online and in-person sections, were assessed for learning by means of the course final exam (out of 45 marks). We compare the mean final exam scores for the students in the two sections.

Data and pre-processing

We begin by reading in the data:

```
library(tidyverse)
```

```
my_url <- "http://ritsokiguess.site/STAC32/proggo.csv"
prog0 <- read_csv(my_url)
```

```
## Parsed with column specification:
## cols(
##   online = col_double(),
##   classroom = col_double()
## )
```

```
prog0
```

```
## # A tibble: 9 x 2
##   online classroom
##    <dbl>     <dbl>
## 1     32        35
## 2     37        31
## 3     35        29
## 4     28        25
## 5     41        34
## 6     44        40
## 7     35        27
## 8     31        32
## 9     34        31
```

The nine students in each section are in separate columns. This is not an appropriate layout for analysis because each row contains data for *two* separate, unrelated students, and we need to have each student's score in its own row. (See note 1.) This means making the data longer (see note 2):

```
prog0 %>% pivot_longer(everything(),
                       names_to = "instruction",
                       values_to = "mark") -> prog
prog
```

```
## # A tibble: 18 x 2
##    instruction  mark
##    <chr>       <dbl>
##  1 online         32
##  2 classroom      35
##  3 online         37
##  4 classroom      31
##  5 online         35
##  6 classroom      29
##  7 online         28
##  8 classroom      25
##  9 online         41
## 10 classroom      34
## 11 online         44
## 12 classroom      40
```

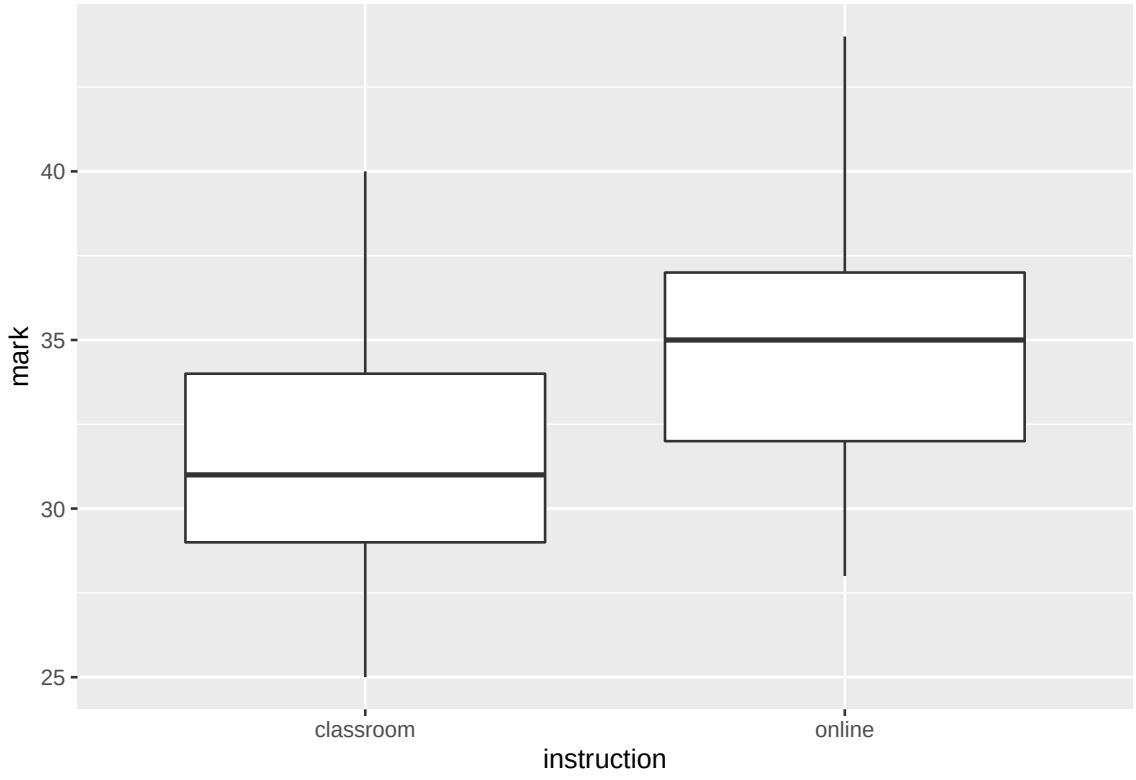
```
## 13 online         35
## 14 classroom      27
## 15 online         31
## 16 classroom      32
## 17 online         34
## 18 classroom      31
```

Now everything is arranged as we need, and we can proceed to analysis.

Analysis

We begin by visualizing the data. With one quantitative variable `mark` and one categorical variable `instruction`, a boxplot will give us an overall picture (see note 3):

```
ggplot(prog, aes(x= instruction, y = mark)) + geom_boxplot()
```



It looks as if the average (median) mark is somewhat higher for the students in the online section. In addition, both distributions look reasonably symmetric with no outliers, and they appear to have similar spread. (see note 4.)

With that in mind, it seems sensible to compare the mean final exam marks in the two groups by a pooled two-sample t-test, in order to see whether the apparent difference in performance is any more than chance. The aim of the course coordinator was to see whether there was any difference between the two sections, without having any prior idea about which teaching method would be better, so a two-sided test is appropriate: (see note 5.)

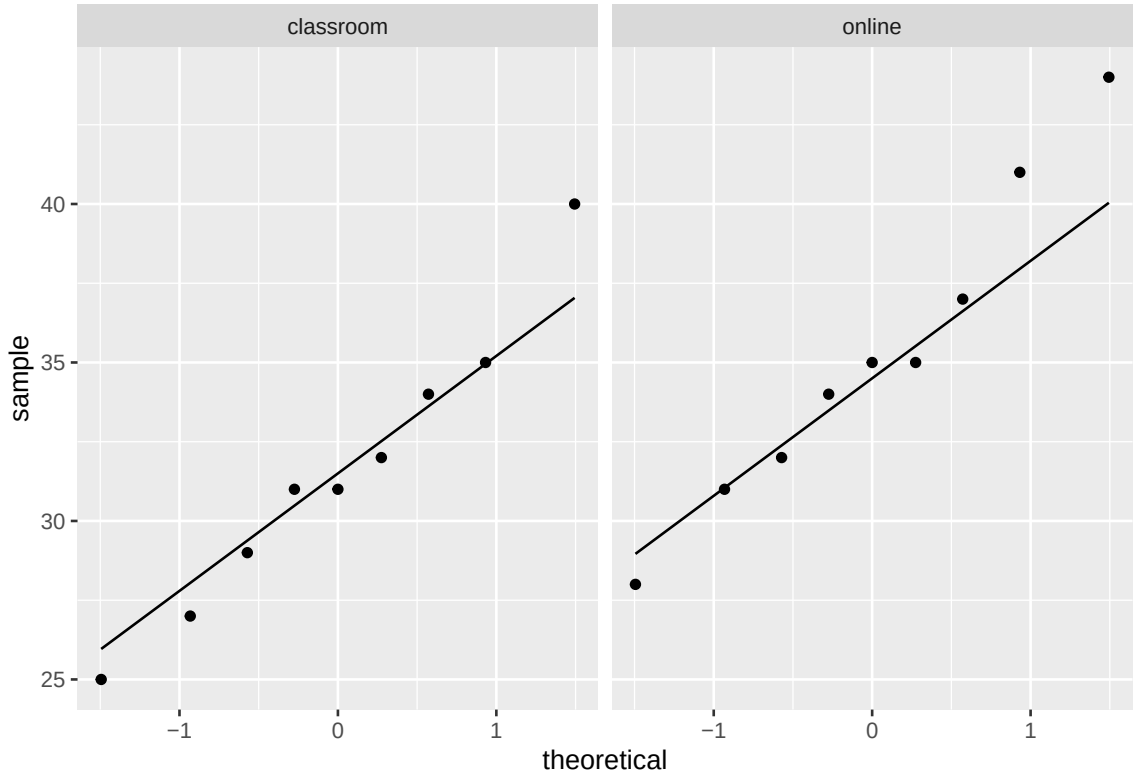
```
t.test(mark~instruction, data = prog, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  mark by instruction
## t = -1.6495, df = 16, p-value = 0.1185
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -8.379039  1.045706
## sample estimates:
## mean in group classroom    mean in group online
##                31.55556                35.22222
```

The P-value of 0.1185 is not smaller than 0.05, so there is no evidence of any difference between the mean final exam marks of the students in the two sections. The difference between the two groups shown on the boxplot is the kind of thing that could be observed by chance if the students were performing equally well under the two methods of instruction. (See notes 6 and 7.)

This t-test comes with the assumption that the data in each group come from a normal distribution, at least approximately. With two small samples, this will not be easy to assess, but we can at least look for any gross violations, using a normal quantile plot for each group:

```
ggplot(prog, aes(sample = mark)) + stat_qq() + stat_qq_line() +
  facet_wrap(~instruction)
```



The largest observation in the classroom group and the largest two observations in the online group are a little bigger than expected, but they were not big enough to be flagged as outliers on the boxplots, so I conclude that the normality is good enough to justify the t-test that we did. (See

note 8.)

Conclusions (see note 9)

We found that there is no significant difference in the performance of the students learning in the classroom compared to those learning online. Before leaping to generalize to other classes, however, we should note two limitations of the study (see note 10):

- the sample sizes were very small; if we had observed this size of difference between the two groups in larger samples, we might have been able to show that the difference was significant.
- we have no information about how the students were allocated to the groups, and it seems likely that the students were allowed to choose their own method of instruction. If the students had been randomly allocated to instruction method, we could have been more confident that any differences observed were due to the instruction method, rather than also having something to do with the relative ability of the students who chose each instruction method.

We feel that it would be worth running another study of this type, but with larger sample sizes and randomly allocating students to instruction types. This latter, however, risks running into ethical difficulties, since students will normally wish to choose the section they are in.

Thus ends the report.

Notes:

1. Say something about the kind of data layout you have, and whether it's what you want.
2. You can do a t-test without rearranging the data (the method is that of the "all possible t-tests" discussion in the ANOVA section), but if you try to draw plots with the data laid out that way, you will at best be repeating yourself a lot and at worst get stuck (if you try to make side-by-side boxplots). The right format of data should give you no surprises!

I split the pivot-longer onto several lines so that it wouldn't run off the right side of the page. Recall that R doesn't mind if you have it on one line or several, as long as it can tell that the current line is incomplete, which it must be until it sees the closing parenthesis on the `pivot_longer`.

3. A normal quantile plot is also justifiable here, but I think it would take more justification, because you would have to sell your reader on the need for normality this early in the story. A standard plot like a boxplot needs no such justification; it just describes centre, spread and shape, exactly the kind of thing your first look at the data should be telling you about.
4. Say something about what the graph is telling you. It makes sense to do as I did and look forward to the kind of inference you are going to try. If you do a normal quantile plot here, you can formally assess the assumptions before you do the test, which you might argue makes more sense, rather than doing a second plot and assessing the assumptions later as I do.
5. You might be able to justify a one-sided test here, along the lines of "is the online instruction significantly *worse*?", but in any case, you need to justify the one-sided or two-sided test that you do.
6. Conclusion *in the context of the data*, as ever. Writing "we fail to reject the null hypothesis" and then stopping invites your reader to ask "so what?".
7. You might have chosen to do a different test, but your choice needs to be properly justified. I think the pooled t-test is the best choice here. If you thought those marks were not normal enough, then you need to do Mood's median test, explaining why. That comes out like this:

```
median_test(prog, mark, instruction)
```

```
## $table
##            above
## group       above below
##   classroom     3     6
##   online        6     3
##
## $test
##        what     value
## 1 statistic 2.0000000
## 2        df 1.0000000
## 3   P-value 0.1572992
```

The (two-sided) P-value is a little bigger than the one for the pooled t-test, but the conclusion is the same. (If you think the test should be one-sided, justify dividing your P-value by 2, if you can. In the case that your one-sided alternative was that classroom teaching was *better*, you cannot reject the null in favour of that, because the online students actually have better marks in these samples. In that case, halving the P-value would not be justifiable.)

If you thought that the normality was all right, but the equal spreads was not, you will need to justify the unequal spreads. Perhaps the best way here is to say that you are unsure whether the spreads are close enough to equal, so you are doing the Welch test to be safe. That looks like this:

```
t.test(mark~instruction, data = prog)
```

```
##
##  Welch Two Sample t-test
##
## data:  mark by instruction
## t = -1.6495, df = 15.844, p-value = 0.1187
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -8.382820  1.049486
## sample estimates:
## mean in group classroom    mean in group online
##                31.55556                35.22222
```

As you see, the P-values of the pooled and Welch t-tests are almost identical (which is what I was guessing), but to do the Welch test without comment reveals that you are *not thinking* about which of the two tests is more appropriate here. In the real world, you might get away with it (for these data, the conclusions are the same), but in this course I need to see that your thought process is correct.

A final observation in this note: all three tests give you similar P-values and the same conclusion, so that in the end it didn't really matter which one of them you did. When that happens, it's usually (as here) a sign that a t-test will be best, because it makes the best use of the data (and thus you will get the most power in your test). I cannot stop you doing all three tests behind the scenes and using this to help decide, but strictly speaking your P-value will not be what you say it is, because these tests (really, any tests) are designed so that the test you choose is the only one you do. In any case, your *report* should include only the *one* test you thought was most appropriate. Your reader does not have the time or patience for a detailed comparison of the three tests.

8. You might have combined the assessment of assumptions with your first plot, particularly if you chose a normal quantile plot instead of a boxplot there. As long as you have assessed normality (and equal spreads, if you are happy with the normality) somewhere, this is fine. In

particular, if you ended up doing a Mood median test, you have presumably already decided that the normality was not good enough, and (I hope) you already discussed that somewhere.

Again, bear in mind who is (in the setup that we have) reading your report: someone who might remember about t-tests and normality. Ideas like the bootstrap are *far* too advanced to go into a report like this. If you must include something like that, you need to put it in an appendix, not the main body of the report, so that the person reading your report can get the main ideas without having to wade through that.

The rest of this note is an Extra:

Here is a slightly different way to approach the bootstrap in this case. I learned about it yesterday and in some ways it's clearer than the other way with the `map`s in it. It makes use of a thing called `rowwise` which is a variant of `group_by`: everything that follows works for each row of the dataframe. I'll take you through it.

The first step is `nest_by`, which does two things, one of which is invisible:

```
prog %>% nest_by(instruction)
```

```
## # A tibble: 2 x 2
## # Rowwise:  instruction
##   instruction               data
##   <chr>       <list<tbl_df[,1]>>
## 1 classroom              [9 x 1]
## 2 online                 [9 x 1]
```

This (visibly) creates a list-column containing two mini dataframes `data`. They are the original data with all the data except `instruction` in each one: that is, the other column `mark`. The top one is the nine marks for the students in the `classroom` section, and the bottom one is the nine marks for the students in the `online` section. The invisible thing is that `nest_by` includes a `rowwise`, so that what we do after this is one for *each* row, one at a time.

What we do next is to generate a lot of bootstrap samples, for each group. This almost works, but not quite:

```
prog %>% nest_by(instruction) %>%
  mutate(samples = rerun(10000, sample(data$mark, replace = TRUE)))
```

```
## Error: Problem with `mutate()` input `samples`.
## x Input `samples` can't be recycled to size 1.
## i Input `samples` is `rerun(10000, sample(data$mark, replace = TRUE))`.
## i Input `samples` must be size 1, not 10000.
## i Did you mean: `samples = list(rerun(10000, sample(data$mark, replace = TRUE)))` ?
## i The error occurred in row 1.
```

When you see something like this, "can't be recycled to size 1", that means that it is expecting a number each time, but you have something else, in this case a `list` of samples, and so what you do is to wrap the thing on the right of the equals sign in `list()`. This will work whenever you are generating something other than a number or a piece of text:

```
prog %>% nest_by(instruction) %>%
  mutate(samples = list(rerun(10000, sample(data$mark, replace = TRUE))))
```

```
## # A tibble: 2 x 3
## # Rowwise:  instruction
##   instruction               data samples
```

```
##   <chr>       <list<tbl_df[,1]>> <list>
## 1 classroom              [9 x 1] <list [10,000]>
## 2 online                 [9 x 1] <list [10,000]>
```

Now, I rather more clearly have 10,000 bootstrap samples, one for each method of instruction.

The next thing is that I want the mean of each bootstrap sample, and they are rather too deeply buried at the moment, so let's `unnest` the samples and see what happens:

```
prog %>% nest_by(instruction) %>%
  mutate(samples = list(rerun(10000, sample(data$mark, replace = TRUE)))) %>%
  unnest(samples)
```

```
## # A tibble: 20,000 x 3
## # Groups:   instruction [2]
##    instruction               data samples
##    <chr>       <list<tbl_df[,1]>> <list>
##  1 classroom              [9 x 1] <dbl [9]>
##  2 classroom              [9 x 1] <dbl [9]>
##  3 classroom              [9 x 1] <dbl [9]>
##  4 classroom              [9 x 1] <dbl [9]>
##  5 classroom              [9 x 1] <dbl [9]>
##  6 classroom              [9 x 1] <dbl [9]>
##  7 classroom              [9 x 1] <dbl [9]>
##  8 classroom              [9 x 1] <dbl [9]>
##  9 classroom              [9 x 1] <dbl [9]>
## 10 classroom              [9 x 1] <dbl [9]>
## # ... with 19,990 more rows
```

Now we have one row for each bootstrap sample (20,000 rows in the dataframe altogether), which is good, because now we can find the mean of each bootstrap sample. There is one more thing first, though: we need to say that the calculation of the mean is to be done on each row of this big dataframe, so we need to say `rowwise` here (as well, even though `nest_by` implicitly includes a `rowwise`: it's the wrong `rowwise`, because that one was on the *two* rows of the dataframe we had before). In return for that, working out the mean of each bootstrap sample is much simpler, because the `rowwise` is a kind of group-by-rows, so that the calculation is done for each row, in the same spirit as `group_by` and `summarize`:

```
prog %>% nest_by(instruction) %>%
  mutate(samples = list(rerun(10000, sample(data$mark, replace = TRUE)))) %>%
  unnest(samples) %>%
  rowwise() %>%
  summarize(sample_mean = mean(samples))
```

```
## `summarise()` regrouping output by 'instruction' (override with `.groups` argument)
```

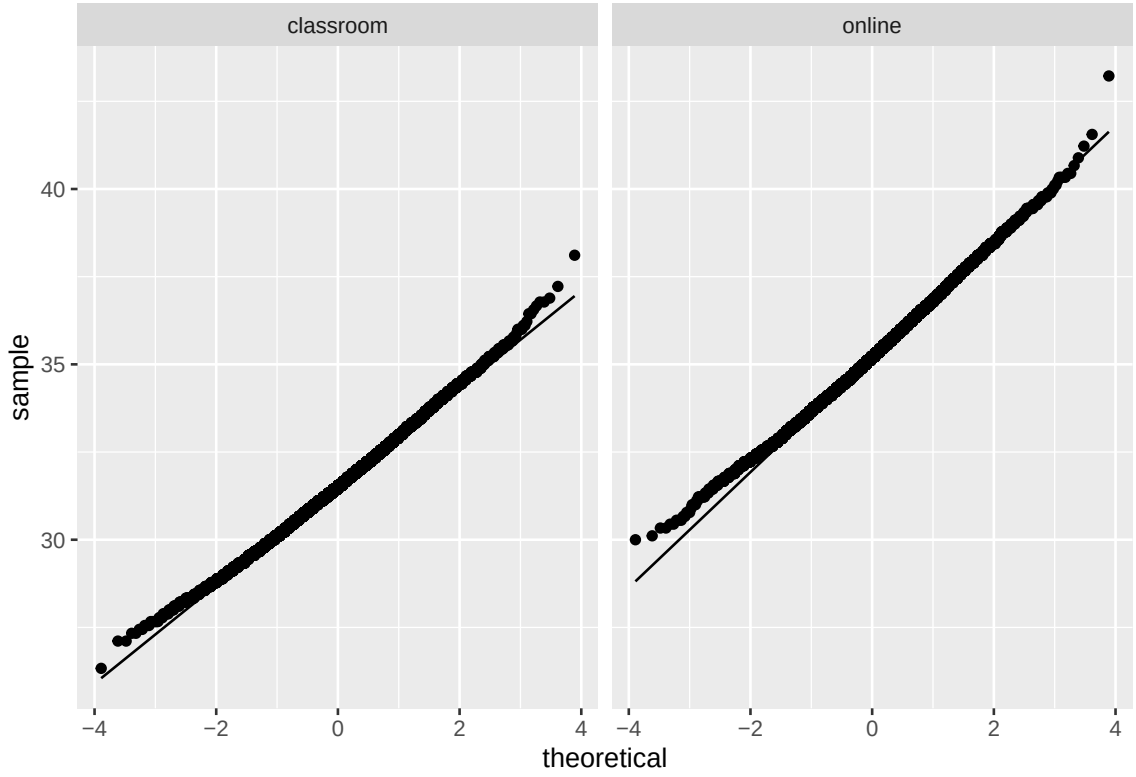
```
## # A tibble: 20,000 x 2
## # Groups:   instruction [2]
##    instruction sample_mean
##    <chr>             <dbl>
##  1 classroom          32.2
##  2 classroom          33.7
##  3 classroom          34.2
##  4 classroom          30.2
```

```
##  5 classroom          33.2
##  6 classroom          29.9
##  7 classroom          30.3
##  8 classroom          30.7
##  9 classroom          31.7
## 10 classroom          30.7
## # ... with 19,990 more rows
```

and now we can make normal quantile plots of these, separately for each form of instruction:

```
prog %>% nest_by(instruction) %>%
  mutate(samples = list(rerun(10000, sample(data$mark, replace = TRUE)))) %>%
  unnest(samples) %>%
  rowwise() %>%
  summarize(sample_mean = mean(samples)) %>%
  ggplot(aes(sample = sample_mean)) + stat_qq() + stat_qq_line() +
  facet_wrap(~instruction)
```

```
## `summarise()` regrouping output by 'instruction' (override with `.groups` argument)
```



These are both very much normal. This is further support for the t-test (pooled or whatever you thought was best) being fine; even with only nine observations in each group, the possible upper-end outliers were not outlying enough to be a problem.

As I say, though, this is very much for *you*, and not for the report, given who would be reading it. If you were doing a presentation on this, the bootstrap stuff is something you would keep in reserve in case someone asks about the appropriateness of the t-test, and *then* you could talk about it, but otherwise you wouldn't mention it at all.

9. You definitely need some conclusions. If the department chair is busy (quite likely), this is the only part of the entire report they may be able to read.

10. This is the place to put limitations, and recommendations for next time. The sample size one is (I hope) obvious, but you can also get at the other one by asking yourself how the students were assigned to instruction methods. The classical comparative study assigns them at random; that way, you know that the two groups of students are at least supposed to be about equal on ability (and anything else) before you start. But if the students choose their own groups, it might be that (say) the weaker students choose the classroom instruction, and in that case the reason the online group came out looking better might be that they were stronger students: it could have nothing to do with the effectiveness of the learning environment at all.

2. Insurance adjusters are concerned that Jocko's Garage is giving estimates for repairing car damage that are too high. To see whether this is indeed the case, ten cars that had been in collisions were taken to both Jocko's Garage and another garage, and the two estimates for repair were recorded. The data as recorded are in http://ritsokiguess.site/STAC32/jocko.txt.

(a) Take a look at the data file (eg. by using your web browser). How are the data laid out? Do there appear to be column headers?

Solution:

The data are laid out in aligned columns, so that we will need to use `read_table` to read it in. There are no column headers, since there is no line at the top of the file saying what each column represents. (The fact that I was asking about column headers is kind of a clue that something non-standard is happening there.)

(b) Read in and display the data file, bearing in mind what you just concluded about it. What names did the columns acquire?

Solution:

As mentioned above, you'll need `read_table`, plus `col_names=FALSE` to *not* read the first row as column names:

```
my_url <- "http://ritsokiguess.site/STAC32/jocko.txt"
cars0 <- read_table(my_url, col_names = FALSE)
```

```
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_character(),
##   X3 = col_double(),
##   X4 = col_double(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double()
## )
```

```
cars0
```

```
## # A tibble: 6 x 7
##   X1    X2       X3    X4    X5    X6    X7
```

```
##   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 a     Car       1     2     3     4     5
## 2 a     Jocko  1375  1550  1250  1300   900
## 3 a     Other  1250  1300  1250  1200   950
## 4 b     Car       6     7     8     9    10
## 5 b     Jocko  1500  1750  3600  2250  2800
## 6 b     Other  1575  1600  3300  2125  2600
```

The column names have become `X1` through `X7`. You'll need to work with these in a minute, so it is good to be aware of that now.

I used a "temporary" name for my dataframe since we are going to be tidying it before we do anything with it, and I'm saving the "good" name `cars` for the tidy one.

(c) Make this data set tidy. That is, you need to end up with columns containing the repair cost estimates at each of the two garages and also identifying the cars, with each observation on one row. Describe your thought process. (It needs to be possible for the reader to follow your description and understand why it works.) Save your tidy dataframe.

Solution:

This looks very far from tidy right now. The things in `X2` look like they will need to be variable names eventually, but there are two copies of them, and there are also five columns of data values that need eventually to become three. Having all the data values in *one* column might be a useful place to start:

```
cars0 %>% pivot_longer(X3:X7, names_to="old_cols", values_to="values")
```

```
## # A tibble: 30 x 4
##    X1    X2    old_cols values
##    <chr> <chr> <chr>     <dbl>
##  1 a     Car   X3            1
##  2 a     Car   X4            2
##  3 a     Car   X5            3
##  4 a     Car   X6            4
##  5 a     Car   X7            5
##  6 a     Jocko X3         1375
##  7 a     Jocko X4         1550
##  8 a     Jocko X5         1250
##  9 a     Jocko X6         1300
## 10 a     Jocko X7          900
## # ... with 20 more rows
```

This is tidier, but it's now *too long*: this has 30 rows but there are only 10 cars, or, depending on your point of view, there are 20 observations on 10 individual cars, so you could justify (in some way) having 20 rows, but not 30.

Now, therefore, we need to pivot *wider*. But to get to this point, we had to try pivoting longer to see what it did, and then go from there. I don't think it's at all obvious that this is what will happen, so I think you need to do a pivot-longer first, *talk about it*, and then move on.

From here, we want to make columns whose names are the things in `X2`, and whose values are the things in `values`. This is exactly what `pivot_wider` does, so add that to our pipe:

```
cars0 %>% pivot_longer(X3:X7, names_to="names", values_to="values") %>%
  pivot_wider(names_from = X2, values_from = values) -> cars
cars
```

```
## # A tibble: 10 x 5
##    X1    names   Car Jocko Other
##    <chr> <chr> <dbl> <dbl> <dbl>
##  1 a     X3        1  1375  1250
##  2 a     X4        2  1550  1300
##  3 a     X5        3  1250  1250
##  4 a     X6        4  1300  1200
##  5 a     X7        5   900   950
##  6 b     X3        6  1500  1575
##  7 b     X4        7  1750  1600
##  8 b     X5        8  3600  3300
##  9 b     X6        9  2250  2125
## 10 b     X7       10  2800  2600
```

This is now tidy: one row for each of the 10 cars, one column containing the repair estimates for each car at each of the two garages, and a column identifying the cars. I think this is best because this is a matched-pairs study, and so you want the two measurements for each individual car in columns next to each other (for `t.test` with `paired=TRUE`).

I think it is best to show the whole pipeline here, even though you are making R work a little harder, rather than having to make up a temporary variable name for the output from `pivot_longer` (that you are never going to look at again after this).

If you thought there were 20 observations, you have a bit more work to do (that you will have to undo later to get the right graph), namely:

```
cars %>% pivot_longer(c(Jocko, Other), names_to="garage", values_to="estimate") -> cars1
cars1
```

```
## # A tibble: 20 x 5
##    X1    names   Car garage estimate
##    <chr> <chr> <dbl> <chr>     <dbl>
##  1 a     X3        1 Jocko      1375
##  2 a     X3        1 Other      1250
##  3 a     X4        2 Jocko      1550
##  4 a     X4        2 Other      1300
##  5 a     X5        3 Jocko      1250
##  6 a     X5        3 Other      1250
##  7 a     X6        4 Jocko      1300
##  8 a     X6        4 Other      1200
##  9 a     X7        5 Jocko       900
## 10 a     X7        5 Other       950
## 11 b     X3        6 Jocko      1500
## 12 b     X3        6 Other      1575
## 13 b     X4        7 Jocko      1750
## 14 b     X4        7 Other      1600
## 15 b     X5        8 Jocko      3600
## 16 b     X5        8 Other      3300
## 17 b     X6        9 Jocko      2250
```

```
## 18 b     X6        9 Other      2125
## 19 b     X7       10 Jocko      2800
## 20 b     X7       10 Other      2600
```

This would be the right thing to do if you had independent observations (that is, *20 different* cars, and you randomly choose a garage to send each one to). But you can have a car assessed for repair without actually repairing it, so it makes more sense to send each car to both garages, and compare like with like. Compare the kids learning to read; once a child has learned to read, you can't teach them to read again, so that study had to be done with two independent samples.

Extra: I thought about starting by making the dataframe even wider:

```
cars0 %>% pivot_wider(names_from = X2, values_from = X3:X7)
```

```
## # A tibble: 2 x 16
##   X1    X3_Car X3_Jocko X3_Other X4_Car X4_Jocko X4_Other X5_Car X5_Jocko
##   <chr>  <dbl>    <dbl>    <dbl>  <dbl>    <dbl>    <dbl>  <dbl>    <dbl>
## 1 a          1     1375     1250      2     1550     1300      3     1250
## 2 b          6     1500     1575      7     1750     1600      8     3600
## # ... with 7 more variables: X5_Other <dbl>, X6_Car <dbl>, X6_Jocko <dbl>,
## #   X6_Other <dbl>, X7_Car <dbl>, X7_Jocko <dbl>, X7_Other <dbl>
```

This is sort of the right thing, but there repeat columns now, depending on where the data values came from in `cars0`. What we want to do now is *some* kind of `pivot_longer`, creating three columns called `Car`, `Jocko`, and `Other`. If we only had *one* kind of thing to make longer, this would be a standard `pivot_longer`. But we have three. There are two "extras" to `pivot_longer` that will get you to the right place. The first one is to give multiple inputs to `names_to`, because the column names encode two things: where in the original data frame the value came from (which is now junk to us), and what the value actually represents, which we definitely *do* want to keep. I don't have a good name for it, though, so I'll call it `z` for now. Note that we need a `names_sep` that says what the two things in the column names are separated by, the underscore that the `pivot_wider` put in there:

```
cars0 %>% pivot_wider(names_from = X2, values_from = X3:X7) %>%
  pivot_longer(-X1, names_to = c("junk", "z"), names_sep="_")
```

```
## # A tibble: 30 x 4
##    X1    junk  z     value
##    <chr> <chr> <chr> <dbl>
##  1 a     X3    Car       1
##  2 a     X3    Jocko  1375
##  3 a     X3    Other  1250
##  4 a     X4    Car       2
##  5 a     X4    Jocko  1550
##  6 a     X4    Other  1300
##  7 a     X5    Car       3
##  8 a     X5    Jocko  1250
##  9 a     X5    Other  1250
## 10 a     X6    Car       4
## # ... with 20 more rows
```

This is now exactly what I got by *starting* with `pivot_longer`, and so the same `pivot_wider` that I finished with before will tidy this up:

```
cars0 %>% pivot_wider(names_from = X2, values_from = X3:X7) %>%
  pivot_longer(-X1, names_to = c("junk", "z"), names_sep="_") %>%
  pivot_wider(names_from = z, values_from = value)
```

```
## # A tibble: 10 x 5
##    X1    junk    Car Jocko Other
##    <chr> <chr> <dbl> <dbl> <dbl>
##  1 a     X3        1  1375  1250
##  2 a     X4        2  1550  1300
##  3 a     X5        3  1250  1250
##  4 a     X6        4  1300  1200
##  5 a     X7        5   900   950
##  6 b     X3        6  1500  1575
##  7 b     X4        7  1750  1600
##  8 b     X5        8  3600  3300
##  9 b     X6        9  2250  2125
## 10 b     X7       10  2800  2600
```

This is now tidy, so you have achieved what you set out to do, but you have not done it the best way, so you should expect to lose a little something.

This kind of longer-then-wider happens often enough that there is an option in `pivot_longer` to do it in one step. Let's remind ourselves of where we were:

```
cars0 %>% pivot_wider(names_from = X2, values_from = X3:X7)
```

```
## # A tibble: 2 x 16
##   X1    X3_Car X3_Jocko X3_Other X4_Car X4_Jocko X4_Other X5_Car X5_Jocko
##   <chr>  <dbl>    <dbl>    <dbl>  <dbl>    <dbl>    <dbl>  <dbl>    <dbl>
## 1 a          1     1375     1250      2     1550     1300      3     1250
## 2 b          6     1500     1575      7     1750     1600      8     3600
## # ... with 7 more variables: X5_Other <dbl>, X6_Car <dbl>, X6_Jocko <dbl>,
## #   X6_Other <dbl>, X7_Car <dbl>, X7_Jocko <dbl>, X7_Other <dbl>
```

The second part of those funky column names needs to become *the names of our new columns*. To make that happen in one step, you put the special indicator `.value` in where we had `z` before:

```
cars0 %>% pivot_wider(names_from = X2, values_from = X3:X7) %>%
  pivot_longer(-X1, names_to = c("junk", ".value"), names_sep="_")
```

```
## # A tibble: 10 x 5
##    X1    junk    Car Jocko Other
##    <chr> <chr> <dbl> <dbl> <dbl>
##  1 a     X3        1  1375  1250
##  2 a     X4        2  1550  1300
##  3 a     X5        3  1250  1250
##  4 a     X6        4  1300  1200
##  5 a     X7        5   900   950
##  6 b     X3        6  1500  1575
##  7 b     X4        7  1750  1600
##  8 b     X5        8  3600  3300
##  9 b     X6        9  2250  2125
## 10 b     X7       10  2800  2600
```

and as if by magic, we have tidiness. It's best to discover this and do it in two steps, though by starting with `pivot_wider` you have made it more difficult for yourself. By starting with `pivot_longer`, it is a very standard longer-then-wider, and there is nothing extra you have to learn. (The column `X1` I added to the data so that `pivot_wider` would work smoothly. See what happens if you remove it with `select(-X1)` before you start pivoting.)

There is usually a relatively simple way to do these, and if your way is complicated, that is an invitation to try it again a different way. I don't think there's a way to do it in *one* step, though, because those things in `X2` have to get to column names somehow, and they can only do so by being attached to which original column the values came from.

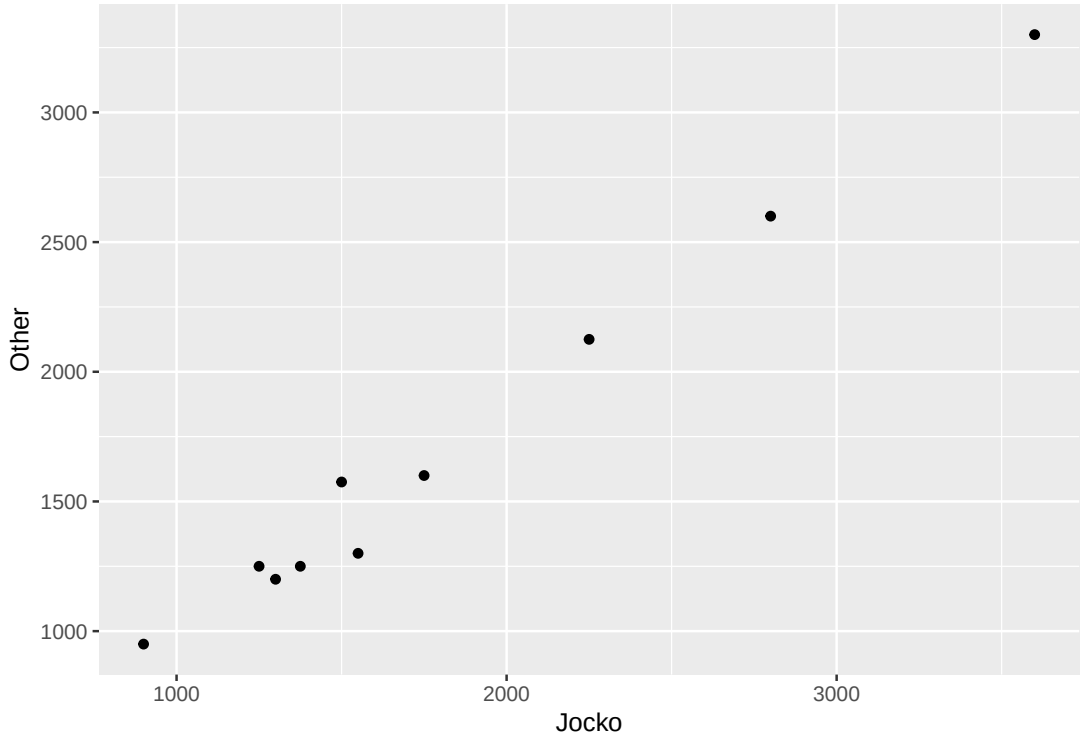
All of these ideas are here, which is a dense read, but worth working through to see what is possible. This problem is of the type in "Longer, then wider".

(d) Make a suitable graph to assess the comparison of interest, and say briefly what your graph tells you.

Solution:

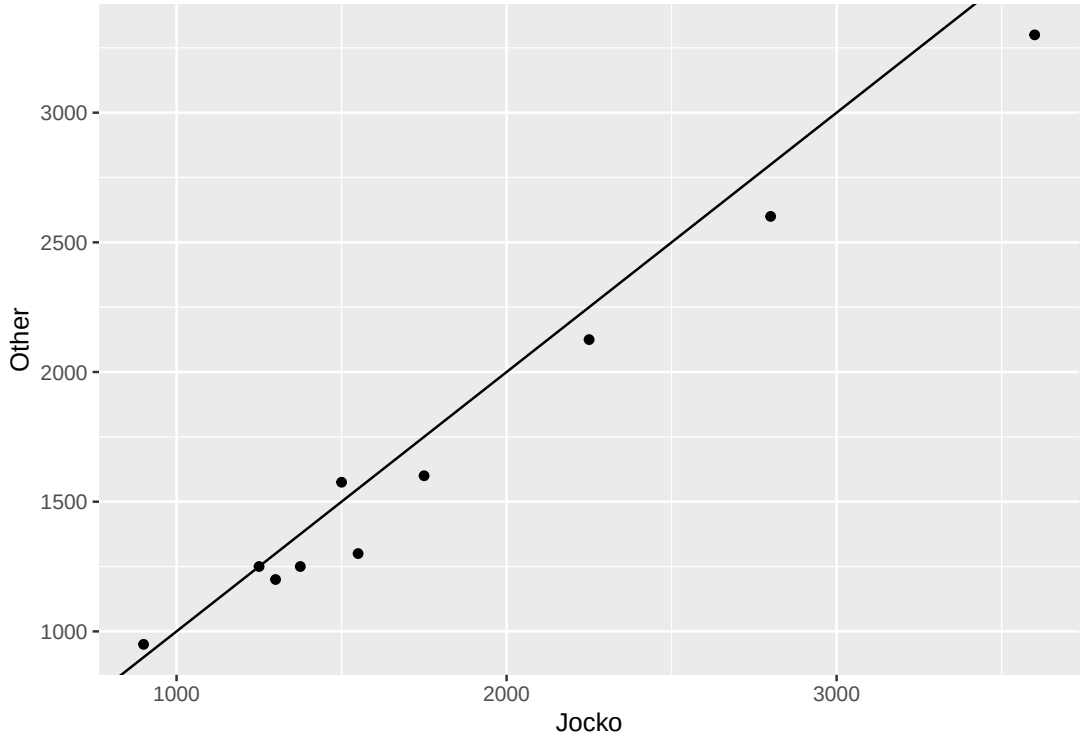
You might be tempted to look at `cars`, see two quantitative variables, and think "scatterplot":

```
ggplot(cars, aes(x=Jocko, y=Other)) + geom_point()
```



This says that a repair that is more expensive at one garage is more expensive at the other as well, which is true, but it's an answer to the *wrong question*. We care about whether Jocko's Garage is more expensive than the other one *on the same car*. To rescue the scatterplot, you can add the line $y = x$ to the graph and see which side of the line the points are, which you might have to find out how to do:

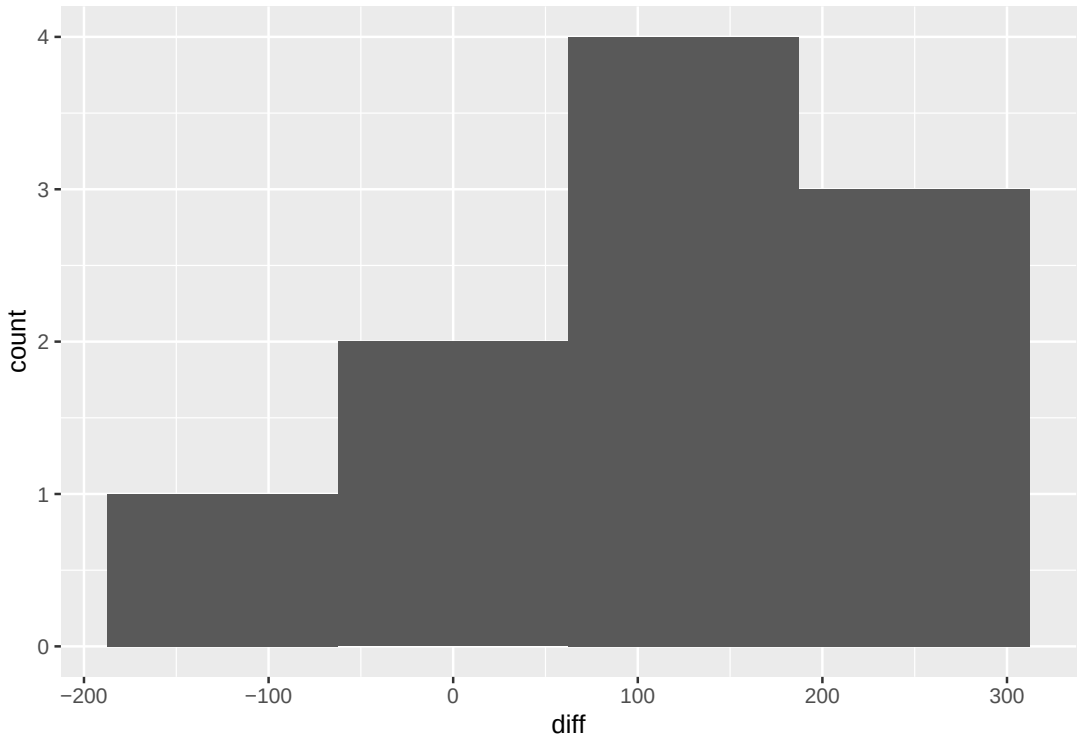
```
ggplot(cars, aes(x=Jocko, y=Other)) + geom_point() + geom_abline(slope = 1, intercept = 0)
```



More of the points are below and to the right of the line, indicating that Jocko's Garage is typically more expensive (in the cases where the other garage is more expensive, there is not much in it).

There is a more direct approach here, based on the idea that a matched pairs test looks at the differences between the two estimates for each car: work out the differences, and make a *one*-sample plot of them:

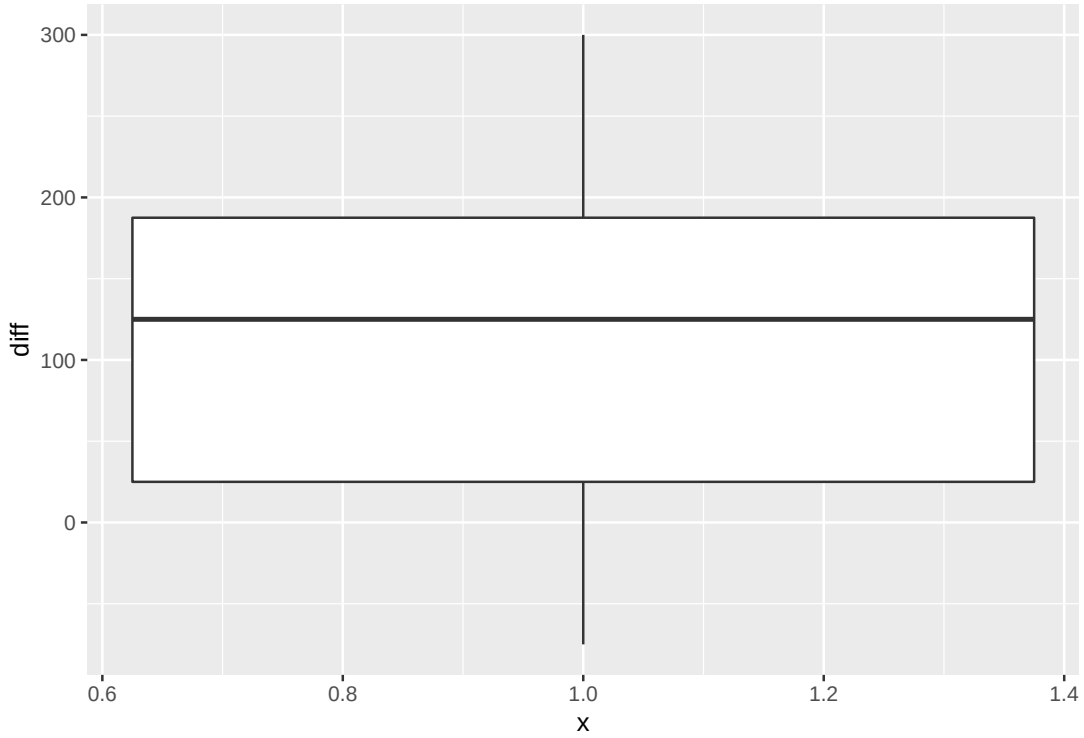
```
cars %>% mutate(diff=Jocko-Other) %>%
  ggplot(aes(x=diff)) + geom_histogram(bins = 4)
```



Most of the differences, this way around, are positive, so the indication is that Jocko's Garage is indeed more expensive. Don't have too many bins.

A one-sample boxplot of the differences would also work:

```
cars %>% mutate(diff=Jocko-Other) %>%
  ggplot(aes(x=1, y=diff)) + geom_boxplot()
```



This tells you that at least 75% of the differences are positive.

If you ended up with my `cars1`:

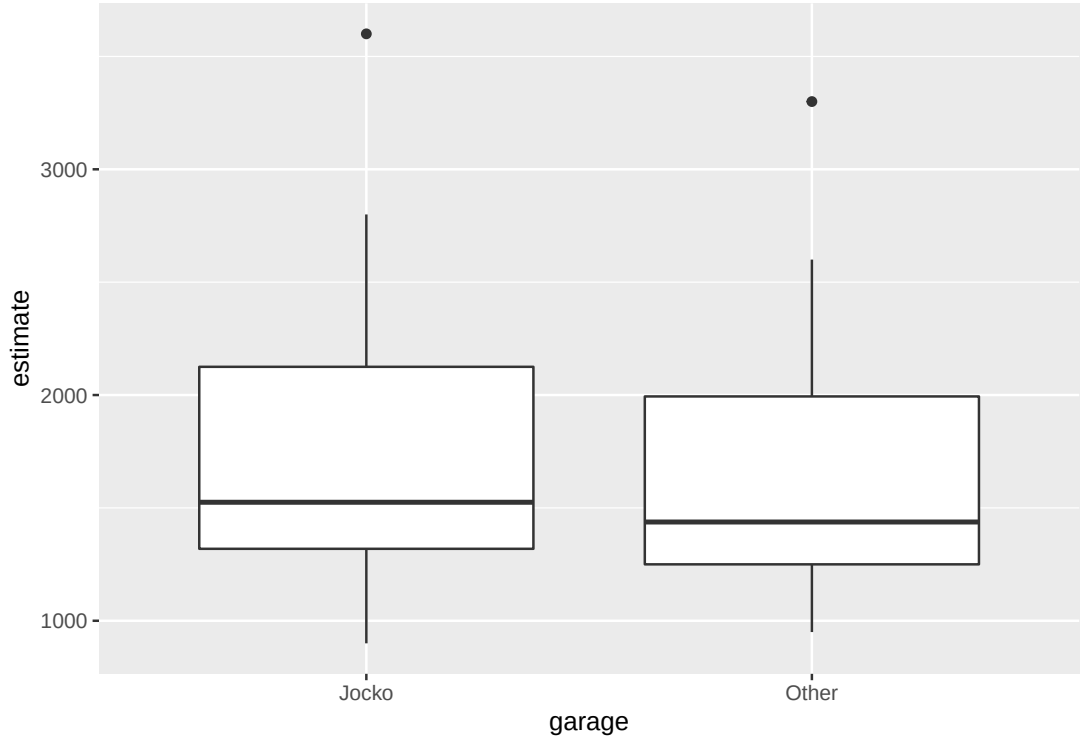
```
cars1
```

```
## # A tibble: 20 x 5
##    X1    names   Car garage estimate
##    <chr> <chr> <dbl> <chr>     <dbl>
##  1 a     X3        1 Jocko      1375
##  2 a     X3        1 Other      1250
##  3 a     X4        2 Jocko      1550
##  4 a     X4        2 Other      1300
##  5 a     X5        3 Jocko      1250
##  6 a     X5        3 Other      1250
##  7 a     X6        4 Jocko      1300
##  8 a     X6        4 Other      1200
##  9 a     X7        5 Jocko       900
## 10 a     X7        5 Other       950
## 11 b     X3        6 Jocko      1500
## 12 b     X3        6 Other      1575
## 13 b     X4        7 Jocko      1750
## 14 b     X4        7 Other      1600
## 15 b     X5        8 Jocko      3600
## 16 b     X5        8 Other      3300
## 17 b     X6        9 Jocko      2250
## 18 b     X6        9 Other      2125
## 19 b     X7       10 Jocko      2800
```

```
## 20 b     X7       10 Other      2600
```

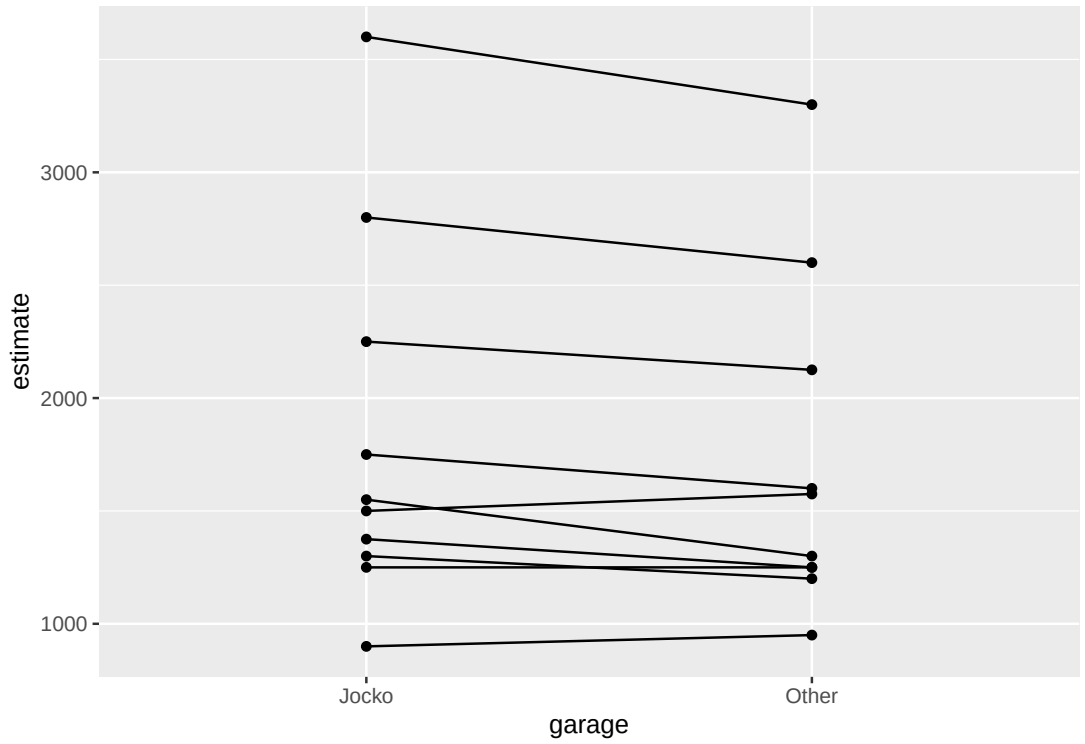
this is "obviously" a boxplot:

```
ggplot(cars1, aes(x=garage, y=estimate)) + geom_boxplot()
```



except that you have not used the fact that each group is measurements on the *same* 10 cars. Here is a way to rescue that:

```
ggplot(cars1, aes(x=garage, y=estimate, group=Car)) + geom_point() + geom_line()
```



The majority of the lines go downhill, so Jocko's Garage is more expensive most of the time. (The lines are really another way to look at the differences.) This last graph I would be good with, since it shows which pairs of measurements are related because of being on the same cars.

(e) Carry out a test to make an appropriate comparison of the mean estimates. What do you conclude, in the context of the data?

Solution:

Comparing means requires the right flavour of t-test, in this case a matched-pairs one, with a one-sided alternative, since we were concerned that the Jocko estimates were bigger. In a matched pairs test, `alternative` says how the first column you name compares with the other one. If your columns are the opposite way to mine, your `alternative` needs to be "less":

```
with(cars, t.test(Jocko, Other, paired = TRUE, alternative = "greater"))
```

```
##
##  Paired t-test
##
## data:  Jocko and Other
## t = 2.8749, df = 9, p-value = 0.009164
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  40.76811      Inf
## sample estimates:
## mean of the differences
##                   112.5
```

Remember that this flavour of t-test doesn't take a `data=`, so you need to use `with` or dollar signs.

The P-value is actually just less than 0.01, so we can definitely conclude that the Jocko estimates are bigger on average.

If you calculated the differences earlier, feel free to use them here:

```
cars %>% mutate(diff=Jocko-Other) %>%
  with(., t.test(diff, mu=0, alternative = "greater"))
```

```
##
##  One Sample t-test
##
## data:  diff
## t = 2.8749, df = 9, p-value = 0.009164
## alternative hypothesis: true mean is greater than 0
## 95 percent confidence interval:
##  40.76811      Inf
## sample estimates:
## mean of x
##     112.5
```

Saving the data frame with the differences in it is probably smart.

Again, if you got to `cars1`, you might think to do this:

```
t.test(estimate~garage, data=cars1, alternative="greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  estimate by garage
## t = 0.32056, df = 17.798, p-value = 0.3761
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  -496.4343       Inf
## sample estimates:
## mean in group Jocko mean in group Other
##              1827.5              1715.0
```

but you would be *wrong*, because the two groups are not independent (they're the same cars at each garage). You have also lost the significant result, because some of the repairs are more expensive than others (at both garages), and this introduces extra variability that this test does not account for.

I said to compare the means, so I don't want a sign test here. If you think we should be doing one, you'll need to make the case for it properly: first, calculate and plot the differences and make the case that they're not normal enough. I see left-skewness in the histogram of differences, but personally I don't find this bad enough to worry about. If you do, make that case (but, a sample of size 10 even from a normal distribution might look this skewed) and then run the right test:

```
cars %>% mutate(diff=Jocko-Other) %>%
  sign_test(diff, 0)
```

```
## $above_below
## below above
##     2     7
##
## $p_values
##   alternative    p_value
## 1       lower 0.98046875
## 2       upper 0.08984375
## 3   two-sided 0.17968750
```

The upper-tail P-value is the one you want (explain why), and this is not quite significant. This is different from the correct t-test for a couple of reasons: there is probably not much power with this small sample, and the two estimates that are higher at the Other garage are not much higher, which the t-test accounts for but the sign test does not.