University of Toronto Scarborough
Department of Computer and Mathematical Sciences
STAC32 (K. Butler), Midterm Exam
October 10, 2025

Aids allowed (on paper, no computers):

- My lecture overheads (slides)
- Any notes that you have taken in this course
- Your marked assignments
- My assignment solutions
- Non-programmable, non-communicating calculator

This exam has 9 numbered pages of questions plus this cover page.

In addition, you have an additional booklet of Figures to refer to during the exam.

The maximum marks available for each part of each question are shown next to the question part.

If you need more space, use the last page of the exam. Anything written on the back of the page will not be graded.

You may assume throughout this exam that the code shown in Figure 1 of the booklet of Figures has already been run.

The University of Toronto's Code of Behaviour on Academic Matters applies to all University of Toronto Scarborough students. The Code prohibits all forms of academic dishonesty including, but not limited to, cheating, plagiarism, and the use of unauthorized aids. Students violating the Code may be subject to penalties up to and including suspension or expulsion from the University.

Soybean yields

The US National Agricultural Statistics Service keeps track of the amount of different agricultural crops harvested each year in each state, going back to the early 20th century. We will be looking at harvests of soybeans. The data file contains four variables: `year`, `state`, the number of `acres` harvested (1 hectare is about 2.5 acres), and the `yield` of soybeans, measured in bushels per acre (1 tonne per hectare is about 15 bushels per acre). There are 2,528 observations altogether.

The first question below asks you to create a dataframe called `soybeans`, which you will be using for the remaining questions about these data.

(1) (3 points) Some of the data file is shown in Figure 2. The data file is called `soybean_harvest.txt`, and is in the folder where you are running R Studio. What code would read these data into a dataframe called `soybeans`, and display (at least some of) this dataframe? (Here, and elsewhere in this exam, if I ask for code, I do not need to see the output that the code produces.)

The data values in Figure 2 are separated by single vertical bars (sometimes called "pipe"; it's the symbol above the backslash on your keyboard). Hence, `read_delim` is called for, with this symbol as delimiter:

```
soybeans <- read_delim("soybean_harvest.txt", "|")
```

```
Rows: 2528 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: "|"
chr (1): state
dbl (3): year, acres, yield

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
soybeans
```

year	state	acres	yield
1948	Alabama	51000	21.5
1955	New Jersey	34000	19.0
1958	Oklahoma	48000	21.0

year	state	acres	yield
1938	North Carolina	162000	11.5
1986	North Dakota	470000	35.0
1939	South Carolina	13000	6.7
1957	Michigan	246000	22.0
1998	Texas	270000	22.0
1949	Illinois	3287000	25.5
1990	Virginia	525000	32.0

Check. Don't forget to include the code for displaying the dataframe (in practice, the first thing you would do after reading in data is to take a look at what you read in). Anything else, like `glimpse`, that displays at least some of the data and that we learned about in class (so *not* `head`), is also acceptable.

Here, and elsewhere, I display the output from my code, so that you can see that it works. As I said at the end of the question, *you* don't need to, and in fact you often won't know exactly what it will be.

This in fact also works:

```
soybeans <- read_delim("soybean_harvest.txt")
```

```
Rows: 2528 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: "|"
chr (1): state
dbl (3): year, acres, yield

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
soybeans
```

year	state	acres	yield
1948	Alabama	51000	21.5
1955	New Jersey	34000	19.0
1958	Oklahoma	48000	21.0
1938	North Carolina	162000	11.5

year	state	acres	yield
1986	North Dakota	470000	35.0
1939	South Carolina	13000	6.7
1957	Michigan	246000	22.0
1998	Texas	270000	22.0
1949	Illinois	3287000	25.5
1990	Virginia	525000	32.0

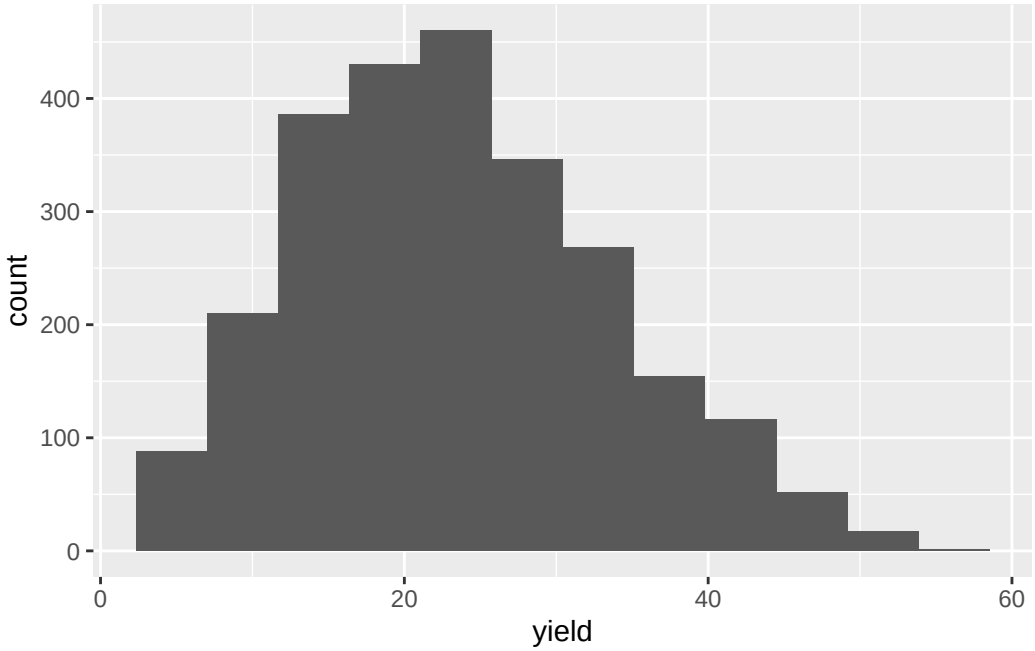
but if you go this way, you must *explain why it would work*; specifically, `read_delim` would infer that the data values were separated by `|`, which you would know from the `Delimiter: "|"` line under Column Specification. Say both what `read_delim` would infer, and how you know it would do so.

Points: 2 for the code, minus 1 per error (eg. forgetting to say what the data values are separated by). 1 point for displaying the data that you read in, via the name of the dataframe or `glimpse`; `head` is not something that I taught you, so no credit for that.

(2) (2 points) What code will create a suitable graph of the variable `yield` (only)?

This is one quantitative variable, so the right graph is a histogram. There are over 2,500 observations, so you can use more bins than you normally would, something like this:

```
ggplot(soybeans, aes(x = yield)) + geom_histogram(bins = 12)
```



Make sure you have *a* number of bins, somewhere between about 8 and 20. My choice comes roughly from Sturges' rule, since $2^{12} = 4096$:

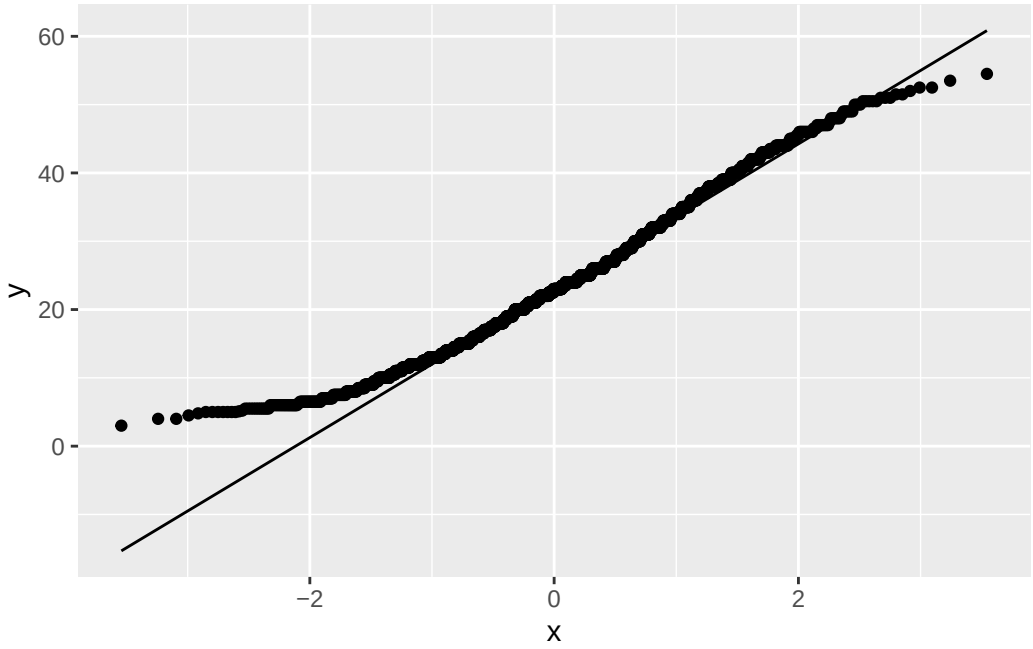
```
nclass.Sturges(soybeans$yield)
```

```
[1] 13
```

but when I tried it, I liked 12 bins better, so that's what I used. On an exam, you have no way to know this, of course, so you will have to pick a number of bins and go with it. Anything not obviously crazy will do. (For example, something like 5 bins will not give a clear picture of the shape; with so many observations, you need more bins than that.)

The second-best choice here is a normal quantile plot:

```
ggplot(soybeans, aes(sample = yield)) + stat_qq() + stat_qq_line()
```



This is, however, a second-best choice here because this plot is really for assessing *normality*, and we don't know whether we are interested in normality yet. (If normality had been the concern here, I would have said or implied that.)

Your code should make it clear what kind of graph you are drawing (you can make it easier for the grader by *stating* this up front).

Points: 2 for a correctly-drawn histogram; 1 for a correctly-drawn normal quantile plot or a histogram with an error (eg. an inappropriate or missing number of bins).

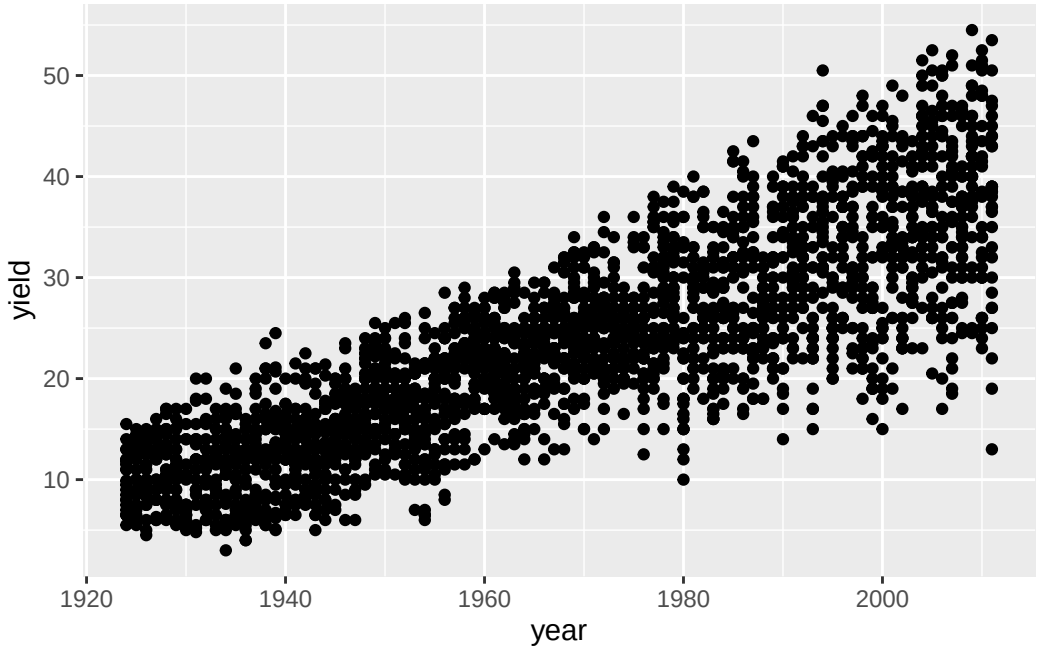
Extra: I didn't give you either of these two graphs and ask for interpretation, but if I had, you would have said (I hope) that the histogram was (slightly) skewed to the right. The normal quantile plot, however, you could interpret either of two ways:

- short tails (the low observations are not low enough and the high ones are not high enough, compared to the normal)
- or, you could say that the observations at the upper end are "close to the line", and then you have right-skewness as on the histogram, but *because the lower tail is short compared to a normal*. (Here is how the normal quantile plot is more informative than the histogram: it is very hard to tell by looking at a histogram how the lengths of the tails compare *to a normal*. All you can really do here is to compare the tails with each other.)

(3) (2 points) Someone suggests that yield has been increasing over time overall. What code would make a graph to assess this?

The two relevant variables are `year` (the measure of time here) and `yield`. These are both quantitative, so a scatterplot is called for, with `year` as explanatory and `yield` as response. (This is actually not a time graph, because there is more than one observation per year: one for each state.) Thus:

```
ggplot(soybeans, aes(x = year, y = yield)) + geom_point()
```



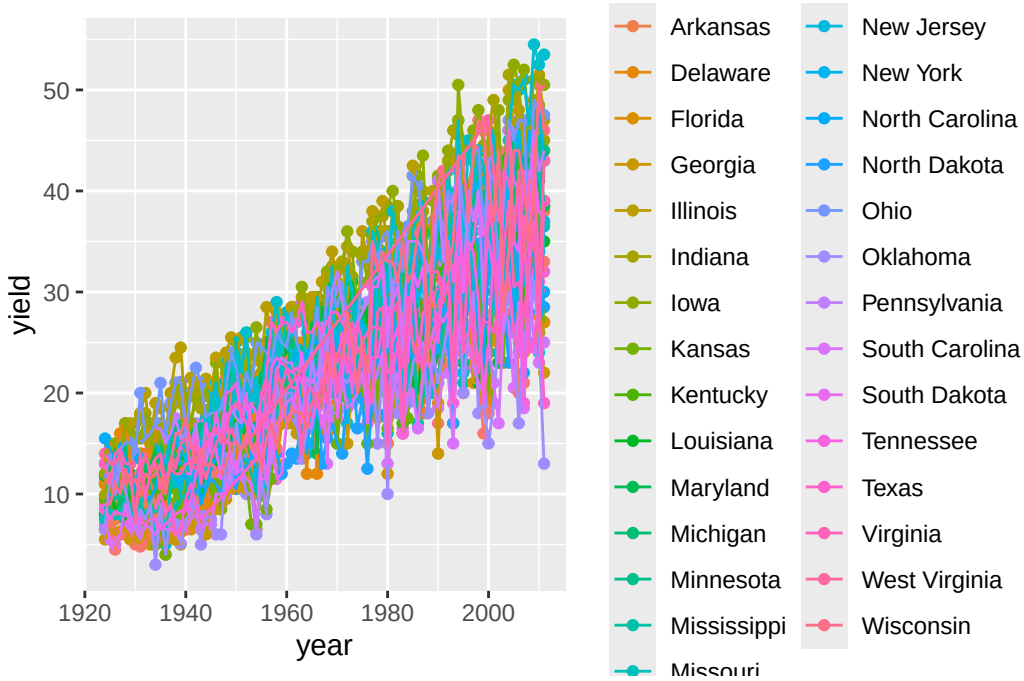
Points: two for this, or with an appropriate `geom_smooth` (with a smooth trend or a regression line). You are expected to understand that the time variable is called `year` (it is the only variable that measures time). Minus a half if you join the points by lines (see comment above and Extra 2 below).

Extra 1: There are *lots* of points in this graph, but an upward trend actually turns out to be pretty clear.

Extra 2: You might connect the observations for the same state together with lines, but that would make a very cluttered graph,[1] and in any case our interest here is in the overall picture:

[1] You can guess this, because I told you in the data description how many observations there were.

```
ggplot(soybeans, aes(x = year, y = yield, colour = state)) +
  geom_point() + geom_line()
```

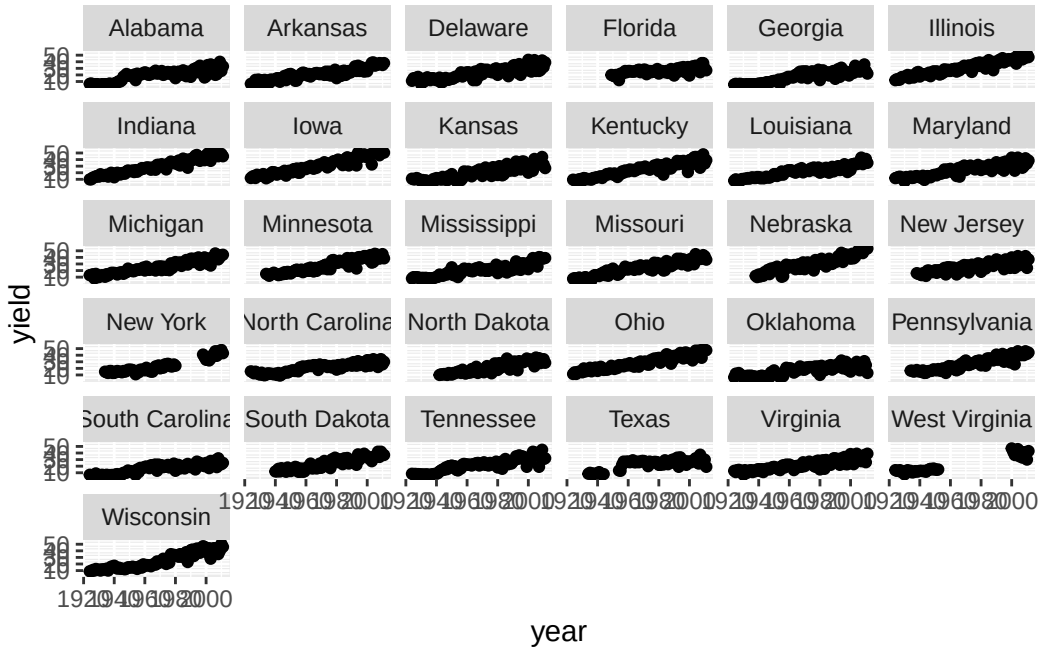


and therefore this kind of graph does not help much (if at all) in answering this question, and does not answer the next one either.

(4) (2 points) The person at the Agricultural Statistics Service asks you to make a graph that shows any trends in yield over time clearly for each state. What code would do this? (Hint: there are a *lot* of observations.)

The Extra to the last question shows you why you need to pay attention to the hint. To show the trends clearly for each state, you need to use facets, but the graph is otherwise the same as the one you just drew:

```
ggplot(soybeans, aes(x = year, y = yield)) + geom_point() +
  facet_wrap(~ state)
```



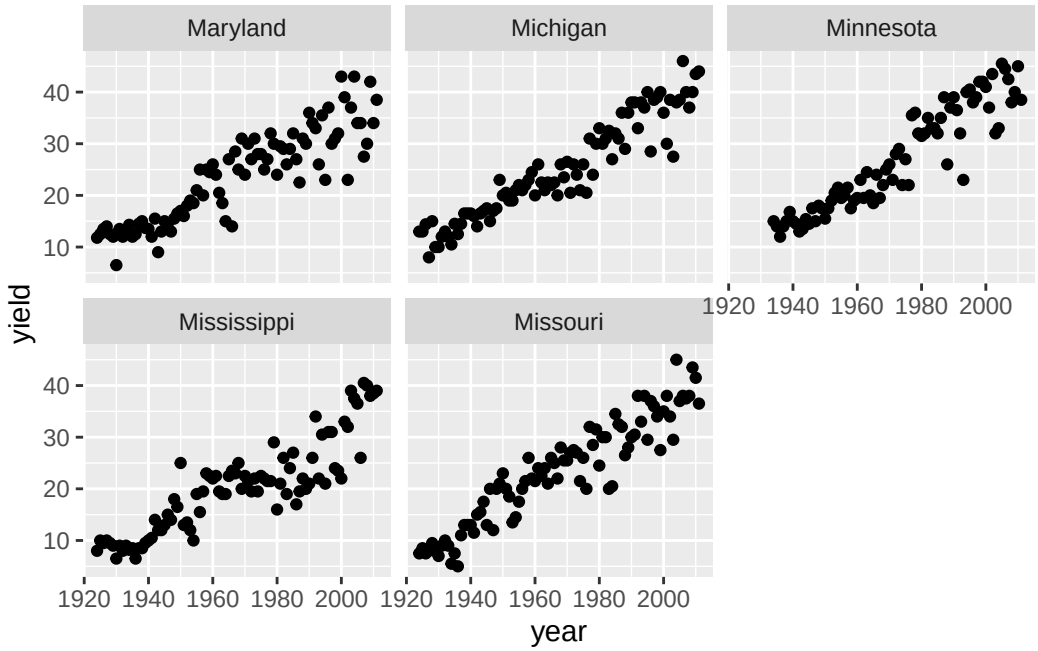
This is still pretty hard to read, because there are so many states, but that's not relevant for you (because you have little way on an exam to know that it will be hard to read). As it turns out, adding a `scales = "free"` will make it even harder to read, but if you can make the case for doing so (for example, you expect yields in some states to be different from yields in others, or you want to focus on the trend rather than the actual values), go ahead.

These are now genuine time trends (one observation per year in each facet), so you are *now* entitled to join the points with lines if you wish, using `geom_line`. See the end of the Extra below.

Two points for saying to add the `facet_wrap` to your previous graph (or for repeating your previous graph code and adding the `facet_wrap` to the end), optionally now with a `geom_line`. Expect to lose at least one if you introduce any *additional* errors beyond what you had before.

Extra: an actual way to get a better graph, one that you can actually read, is to limit the number of states being shown in the array of graphs, such as only displaying the states whose names begin with M:

```
soybeans %>% filter(str_detect(state, "^M")) %>%
  ggplot(aes(x = year, y = yield)) + geom_point() +
  facet_wrap(~ state, ncol = 3)
```



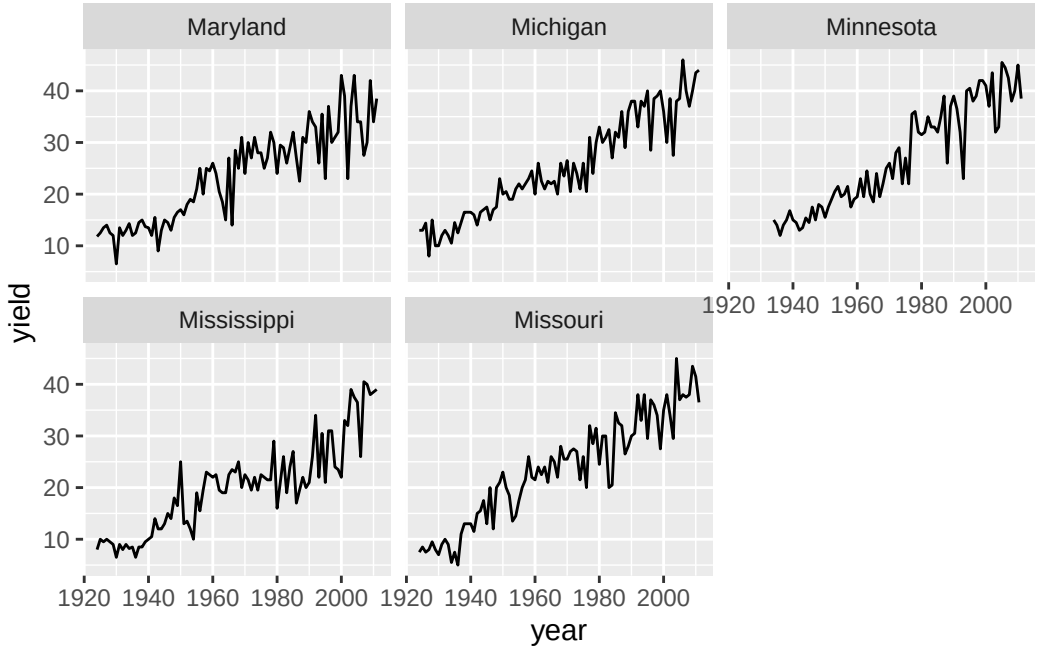
and now you get a better sense that in these states, yields are going up over time with some variability. These, remember, are bushels *per acre* (equivalent to tonnes per hectare), so it not that more land is having soybeans grown on it, but that more soybeans are being grown *on the same amount of land*.

Picking the states whose names start with M is kind of dopey, but you could imagine that the states might be arranged into regions, like "South" or "Midwest", and showing the plots together for the states in the same region might be insightful.

Code notes:

- The `^M` in my `filter` is a regular expression, that we also saw when we looked at the select-helper `matches`. It means "match anything that has an M at the beginning but can have anything else after that". This one is not a select-helper, though, because we are choosing *rows* rather than columns. The `str_detect` returns `TRUE` if the name of the state in that row matches the regular expression (ie., it starts with M), and then the `filter` chooses the rows for which it is true, so this is how to select the *rows* that match something.
- These are now genuine time plots, with one observation per state per year (though there are a lot of years, so it is hard to be sure). I could therefore have joined the points by lines, or even replaced the `geom_point` with a `geom_line`:

```
soybeans %>% filter(str_detect(state, "^M")) %>%
  ggplot(aes(x = year, y = yield)) + geom_line() +
  facet_wrap(~ state, ncol = 3)
```



These look more like time trends now, with a certain amount of variability (particularly in more recent years). It is an aesthetic decision on your part (in collaboration with whoever is going to be reading your work) whether you prefer these, or the ones above that look more like scatterplots.

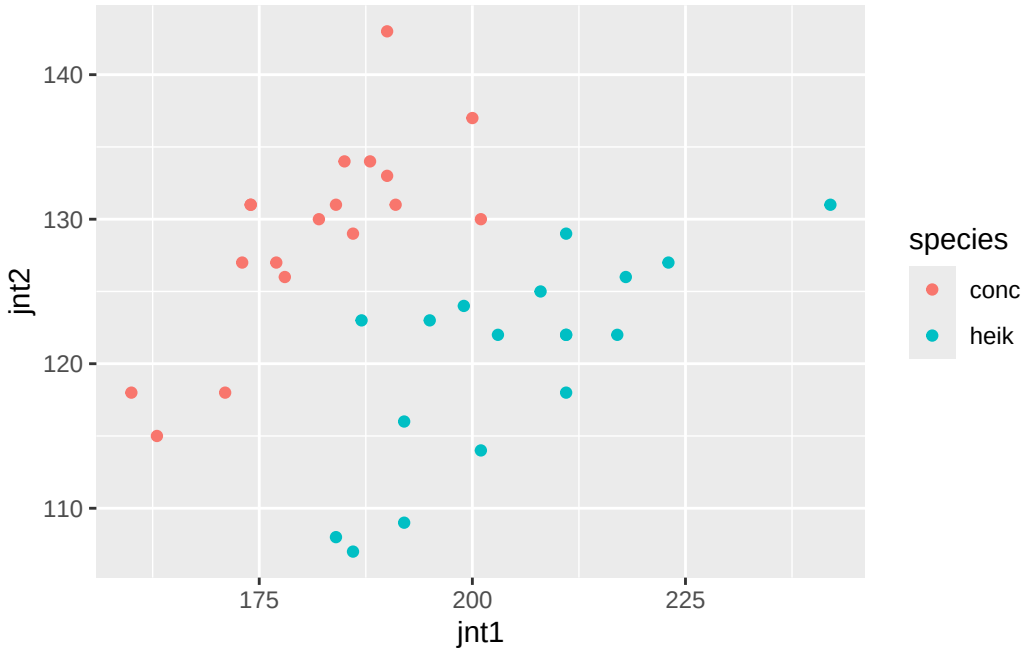
Flea beetles

Flea beetles are small, jumping beetles that feed on the leaves of plants. The data in Figure 3 are measurements of two very similar species of flea beetle, named `conc` and `heik` in column `species`. (These are abbreviations for longer names, but we use the abbreviations here.) The measurements are the lengths of the first leg joint (`jnt1`) and the second leg joint (`jnt2`) in micrometres. The dataframe is called `beetle`.

(5) (3 points) A plot is shown in Figure 4. What code was used to draw this plot?

This, exactly:

```
ggplot(beetle, aes(x = jnt1, y = jnt2, colour = species)) + geom_point()
```



This is a scatterplot with the species distinguished by colour, so the two quantitative variables go on the axes and the categorical one is colour. Look carefully to see that `jnt1` is on the x-axis and `jnt2` is on the y-axis.

Minus a point per mistake, down to 1 point if you got anything substantial correct.

Extra: there is no reason for the two quantitative variables to be this way around, since neither is a response. If I had asked you for a suitable graph of these three variables, it would have been fine to produce either the plot of Figure 4 or with the two quantitative variables the other way around. But the code to make *this* plot has to be exactly as shown. A reason for my doing it this way is so that the grader can more easily grade it correctly.

(6) (2 points) How can you distinguish the two species? Explain briefly.

The species `conc` (red) is up and to the left on the plot (one point), so a beetle with a large value of `jnt2` and/or a small value of `jnt1` is a `conc`, otherwise it's a `heik` (the second point).

You could also do it the other way around: a large value of `jnt1` and/or a small value of `jnt2` is a `heik`, otherwise it's a `conc`.

You can reasonably use either or both of the words "and" and "or", but at least one of them needs to be present, to emphasize that the decision about `species` depends on *both* of `jnt1` and `jnt2`.

Extra: this is the kind of data for which we use discriminant analysis in D29, to answer the question "what is it about the quantitative variables we measured that distinguishes the groups (species)?" In this context, we are thinking of both `jnt1` and `jnt2` as response variables and of `species` as something that affects them both, so the sort of thing that should be in your head now is something like "a two-sample t-test, but in two dimensions". This is what becomes a "multivariate analysis of variance" or MANOVA, which you will see in D29.

Wine cultivars

Wines from three cultivars (varieties) of grape grown in Italy were assessed for chemical composition. Several bottles of wine were sampled from each cultivar. We are interested in the alcohol content of each wine (in `Alcohol`: a higher value means more alcoholic) and its colour (in `Colour`: a higher value means a more desirable colour). Some of the data are shown in Figure 5.

(7) (3 points) What *three* conclusions do you draw from Figure 6?

Boxplots are for *comparing* distributions, and the features they allow you to assess are median, spread (IQR) and shape, a point each:

- Cultivar `barbera` has the highest median Colour and `grignolino` has the lowest.
- Cultivar `barbera` has the highest IQR (spread) of `Colour` and `grignolino` again has the lowest.
- All three distributions are right-skewed for `Colour`, or `grignolino` has four upper outliers and `barolo` has one.

(8) (2 points) What do you conclude from Figure 7?

As Colour increases, Alcohol increases, but the relationship is weak (there is a lot of scatter).

If you like, think about interpreting the scatterplot in terms of form, direction, strength:

- form: apparently linear (not obviously curved)
- direction: increasing
- strength: weak

I am less interested in whether you think this is linear; the two points are for an upward trend but a weak one. One point for saying that there is no relationship, on the grounds that it captures "weak" but doesn't capture that the relationship is more up than down.

Extra: The upward trend, though weak, is actually real. Looking ahead to regression (later in the course):

```
wine.1 <- lm(Alcohol ~ Colour, data = wine)
summary(wine.1)
```

```

Call:
lm(formula = Alcohol ~ Colour, data = wine)

Residuals:
     Min       1Q   Median       3Q      Max
-1.62083 -0.50404 -0.00891  0.48728  1.80223

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 12.03286    0.12295  97.868  < 2e-16 ***
Colour       0.19133    0.02211   8.654 3.06e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6819 on 176 degrees of freedom
Multiple R-squared:  0.2985,    Adjusted R-squared:  0.2945
F-statistic:  74.9 on 1 and 176 DF,  p-value: 3.056e-15
```

The slope of `Colour` is significantly positive, even if not very big (and thus the R-squared is small but not zero). Hence it is better to say that there is a positive but weak relationship rather than to say that there is no relationship.

(9) (2 points) In the code at the top of Figure 8, the word "colour" appears twice. How are the two uses different?

`Colour` with an uppercase `C` is the colour of the *wine*, on the x-axis; `colour` with a lowercase `c` is the colour of the *points on the plot*. Make sure that you are clear about which usage of "colour" you are talking about each time; another way is like this:

- in `x = Colour`, the Colour is the colour (rating) of the wine, drawn on the x-axis

- in `colour = cultivar`, the different cultivars are drawn as different coloured points.

Extra: I previously had another question about distinguishing `grignolino` from the other cultivars, but the question in "flea beetles" asks the same thing (about distinguishing the species), and is a more interesting question, so I took the one here out.

Flea beetles revisited

Earlier, we met some data about flea beetles. See the description above Question 5; the data were shown in Figure 3.

In the following questions, provide code that will display the items requested from the dataframe, which is called `beetle`. Your code may display all of anything not explicitly mentioned; for example, if you are asked to display some columns, your code may display all the rows of those columns.

(10) (2 points) Display the quantitative columns, without using the names or numbers of any columns.

The prohibition means that you have to find something that the columns you want have in common. This could be that their names start with `jnt`:

```
beetle %>% select(starts_with("jnt"))
```

jnt1	jnt2
191	131
185	134
200	137
173	127
171	118
160	118
188	134
186	129
174	131
163	115
190	143
174	131
201	130
190	133
182	130

jnt1	jnt2
184	131
177	127
178	126
186	107
211	122
201	114
242	131
184	108
211	118
217	122
223	127
208	125
199	124
211	129
218	126
203	122
192	116
195	123
211	122
187	123
192	109

(or some subset of `jnt`).

Or a property they have, such as being `numeric`:

```
beetle %>% select(where(is.numeric))
```

jnt1	jnt2
191	131
185	134
200	137
173	127
171	118
160	118
188	134
186	129

jnt1	jnt2
174	131
163	115
190	143
174	131
201	130
190	133
182	130
184	131
177	127
178	126
186	107
211	122
201	114
242	131
184	108
211	118
217	122
223	127
208	125
199	124
211	129
218	126
203	122
192	116
195	123
211	122
187	123
192	109

Or even of *not* being text, but you have to be careful about this:

```
beetle %>% select(where(\(x) !is.character(x)))
```

jnt1	jnt2
191	131
185	134
200	137

jnt1	jnt2
173	127
171	118
160	118
188	134
186	129
174	131
163	115
190	143
174	131
201	130
190	133
182	130
184	131
177	127
178	126
186	107
211	122
201	114
242	131
184	108
211	118
217	122
223	127
208	125
199	124
211	129
218	126
203	122
192	116
195	123
211	122
187	123
192	109

The `!` for "not" has to be in an actual function, because you cannot use the *name* of a function as we did above.

Points for all of these coding questions: full marks if your code is consistent with what we have done in class and will work. Partial credit such as 1/2 or 2/3 if your code has errors

but would otherwise work; on a 3-mark question, 1/3 for something tangentially relevant, but not close to a complete answer.

(11) (2 points) Display rows 15 through 20 inclusive of the dataframe.

Displaying rows by number is `slice`:

```
beetle %>% slice(15:20)
```

jnt1	jnt2	species
182	130	conc
184	131	conc
177	127	conc
178	126	conc
186	107	heik
211	122	heik

This is the best way, but there might be partial credit if you can find a way to display the right rows without using it, such as this:

```
beetle %>% mutate(r = row_number()) %>%
  filter(r >= 15, r <= 20)
```

jnt1	jnt2	species	r
182	130	conc	15
184	131	conc	16
177	127	conc	17
178	126	conc	18
186	107	heik	19
211	122	heik	20

This would be 2/3 if done correctly.

(12) (2 points) Display only the data for `heik` beetles.

Selecting rows by something that is true is `filter`:

```
beetle %>% filter(species == "heik")
```

jnt1	jnt2	species
186	107	heik
211	122	heik
201	114	heik
242	131	heik
184	108	heik
211	118	heik
217	122	heik
223	127	heik
208	125	heik
199	124	heik
211	129	heik
218	126	heik
203	122	heik
192	116	heik
195	123	heik
211	122	heik
187	123	heik
192	109	heik

You are (as per the instructions) free to display all the columns. Don't forget the *double* equals sign for a logical condition (only one point for a single equals, because that is showing a key misunderstanding).

(13) (3 points) Display all the beetles of the `conc` species whose first leg joint is (strictly) greater than 190 micrometres.

Read the English carefully: you need *both* of the two things to be true, so this is an "and". Hence, two things in a `filter`:

```
beetle %>% filter(species == "conc", jnt1 > 190)
```

jnt1	jnt2	species
191	131	conc
200	137	conc

jnt1	jnt2	species
201	130	conc

or two filters, one after the other (in either order):

```
beetle %>% filter(species == "conc") %>%
  filter(jnt1 > 190)
```

jnt1	jnt2	species
191	131	conc
200	137	conc
201	130	conc

Once again, you can display all the columns. Two points for an error like a single equals sign, but if you did that before and lost marks for it there, you should not get punished again.

Extra: You can tell from Figure 4 that there will not be many of these, since most of the beetles that have a `jnt1` value this large are `heik` (blue on the graph) rather than `conc` (red).

(14) (2 points) Display all the beetles whose first leg joint is less than 175 micrometres or whose second leg joint is greater than 140 micrometres.

This is a somewhat odd thing to ask, but it is an either-or:

```
beetle %>% filter(jnt1 < 175 | jnt2 > 140)
```

jnt1	jnt2	species
173	127	conc
171	118	conc
160	118	conc
174	131	conc
163	115	conc
190	143	conc
174	131	conc

There is only one beetle that satisfies the second condition (the sixth one in the output), and no beetles that satisfy both. I actually decided on the values by looking at the scatterplot.

One point off per error.

(15) (2 points) Sort the beetles by species, breaking ties by the joint 1 measurement in descending order.

`arrange`, with the tiebreaking variable second:

```
beetle %>% arrange(species, desc(jnt1))
```

jnt1	jnt2	species
201	130	conc
200	137	conc
191	131	conc
190	143	conc
190	133	conc
188	134	conc
186	129	conc
185	134	conc
184	131	conc
182	130	conc
178	126	conc
177	127	conc
174	131	conc
174	131	conc
173	127	conc
171	118	conc
163	115	conc
160	118	conc
242	131	heik
223	127	heik
218	126	heik
217	122	heik
211	122	heik
211	118	heik
211	129	heik
211	122	heik
208	125	heik

jnt1	jnt2	species
203	122	heik
201	114	heik
199	124	heik
195	123	heik
192	116	heik
192	109	heik
187	123	heik
186	107	heik
184	108	heik

One point if you did something like this but with an error, of coding or logic.

(16) (3 points) Find the mean of joint 1 measurements that are (strictly) larger than 190 micrometres, for each species of beetle.

This is a group-by and summarize, but with a twist: we only need to consider the joint 1 measurements that are over 190, so we can choose them first and then summarize:

```
beetle %>% filter(jnt1 > 190) %>%
  group_by(species) %>%
  summarize(big_mean = mean(jnt1))
```

species	big_mean
conc	197.3333
heik	208.9333

It also works to do the `filter` second:

```
beetle %>%
  group_by(species) %>%
  filter(jnt1 > 190) %>%
  summarize(big_mean = mean(jnt1))
```

species	big_mean
conc	197.3333

species	big_mean
heik	208.9333

This one actually selects the `jnt1` measurements that are big enough for each group separately, but the result is the same (the same rows get selected whichever way you do it), so you can do it either way.

Give the summary a name of your choosing, since I haven't told you what to call it.

The points are for having these three things:

- the right `filter` in one of the right places
- the right `group_by`
- the right `summarize`

(17) (3 points) (This question requires explanation and *not* code.) The R function `cor` calculates the correlations between the two columns given as its inputs. Some correlations are calculated in Figure 9. With reference to Figure 4, why do the correlation values in Figure 9 make sense? Explain briefly.

The first correlation is close to zero (or, positive but very small). This shows up on the plot by there being (almost) no relationship between the two joint measurements, when both species are taken together (ignoring the colours of the points).

The second pair of correlations are for each species separately. These are both clearly positive and much larger. On the plot, if you look at the points of each colour separately, both sets of points (the red ones and the blue ones) show an upward trend. That is, *within* each species, a larger joint 1 measurement tends to go with a larger joint 2 measurement.

A point each for linking each of the three correlations with something on the graph. Say something about the relationship between joint measurements (to show that you know what the correlations *are* in the context of these data).

Don't forget to talk about the first correlation (the one that is 0.096), and why it is so small (close to zero).5

In other words: work out what the three correlations calculated in Figure 9 are actually correlations *of*, and translate that into something that you also see on Figure 4. The code on Figure 9 gives you a clue as to what is going on each time: the first one is all the data (no group-by) and the second one is for each species separately. Then find a nice clear way of showing that you know what is going on.

We haven't talked about correlations in lecture, but this is prerequisite knowledge (it comes from probably your first class, and if not, the second class where you learned about

regression), and this is conceptually a group-by and summarize (that we *have* seen in lecture). So you have enough knowledge to be able to figure this out.

Extra: Analyzing correlations is kind of weird, because correlations that exist within groups can disappear when you combine groups, and you have to think about what's going on. I find regression (looking forward to later in the course) a clearer way to understand things. Here, imagine that you were predicting joint 2 from joint 1, first without regard to species:

```
beetle.1 <- lm(jnt2 ~ jnt1, data = beetle)
summary(beetle.1)
```

```
Call:
lm(formula = jnt2 ~ jnt1, data = beetle)

Residuals:
     Min       1Q   Median       3Q      Max
-17.4830  -4.2253   0.4881   5.8445  18.3397

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 116.24053   15.29149   7.602 7.85e-09 ***
jnt1          0.04431    0.07880   0.562    0.578
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.243 on 34 degrees of freedom
Multiple R-squared:  0.009217,  Adjusted R-squared:  -0.01992
F-statistic: 0.3163 on 1 and 34 DF,  p-value: 0.5775
```

Nothing is happening at all. Now add species to the regression and suddenly everything becomes significant:[2]

```
beetle.2 <- lm(jnt2 ~ jnt1 + species, data = beetle)
summary(beetle.2)
```

```
Call:
```

[2]Also, R-squared jumps from basically nothing to something substantial.

```
lm(formula = jnt2 ~ jnt1 + species, data = beetle)

Residuals:
   Min     1Q Median     3Q    Max
-7.047 -3.332  0.038  2.622 10.499

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  57.96152   11.13231   5.207 1.00e-05 ***
jnt1          0.39231    0.06103   6.428 2.75e-07 ***
speciesheik -17.96342    2.12810  -8.441 9.41e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.707 on 33 degrees of freedom
Multiple R-squared:  0.6864,    Adjusted R-squared:  0.6674
F-statistic: 36.11 on 2 and 33 DF,  p-value: 4.906e-09
```

that is to say, when you have species in the regression, knowing the joint 1 measurement tells you about the joint 2 measurement, but you need to know *both* joint 1 *and* species to have any success predicting joint 2. My take on regression, which you will see later in the course, is to start with a model containing everything (my `beetle.2`), and then see what I can remove (here, nothing) to simplify things. Starting with model `beetle.1` is more confusing, because it looks as if `jnt1` has nothing to do with `jnt2`, and I have to know to try adding species. This is part of the justification for preferring what regression people call "backward elimination" (starting from `beetle.2` and getting rid of stuff if you can), rather than trying to start from `beetle.1` and seeing if adding anything ends up helping.

Stress on turbine blades

Ten observations were taken on vibratory stress of turbine blades, as shown in Figure 10, in dataframe `turbine`. It is of interest to see whether the mean vibratory stress is 14.5 or whether it is less (in the units given).

(18) (2 points) What do you learn from Figure 11? Explain briefly.

This is a normal quantile plot. The points on it are either:

- close to the line, so the vibratory stress values are close to normal.

- in a very slight curve shape, opening upwards, so the vibratory stress values are very slightly right-skewed. (The two lowest points are too high (bunched up) and the two highest points are also too high (too spread out)).

For the two points, say what you see, and what that tells you about normality. I don't mind which of those two conclusions you come to, as long as your logic for getting there is sound. Alternatively, make a statement about normality first, and then support it (for example, "the vibratory stress values are close to normal because the points are close to the line").

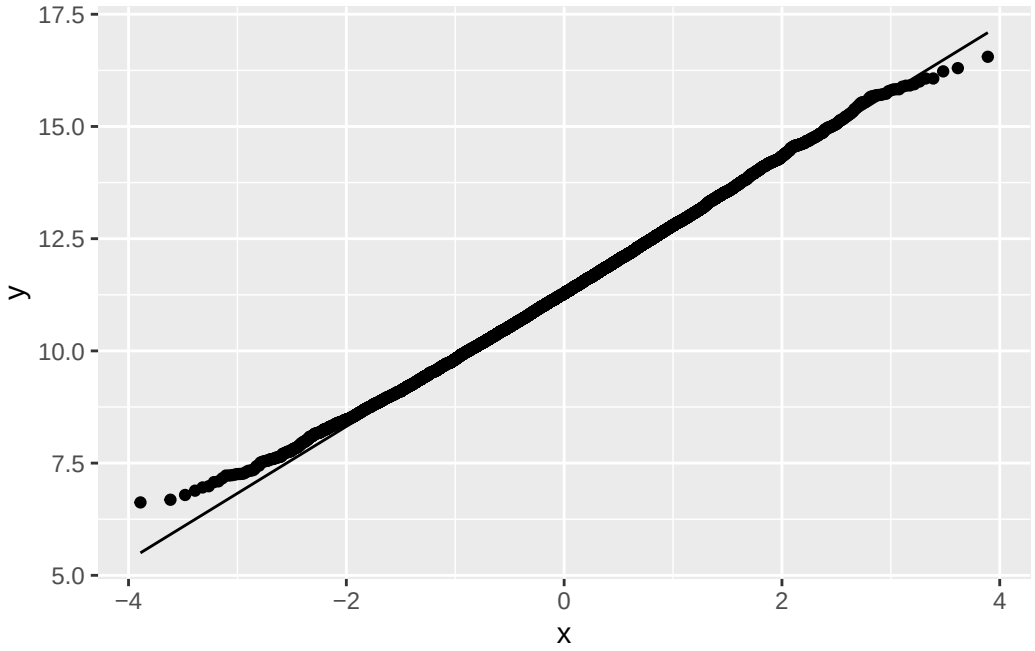
Extra: in the source where I got these data, the instructions were to fit a Rayleigh distribution, which is right-skewed, so if anything, as far as you can tell with ten observations, these data are right-skewed.

(19) (2 points) What *additional* piece of information would support using a t-test for the mean here? How does this additional piece of information offer support?

The additional piece of information is the sample size. This is $n = 10$, which will provide some help in making the sampling distribution of the sample mean close to normal. (Not much, but it doesn't need to be much, even if you thought the distribution was right-skewed.)

Extra: the inevitable bootstrap sampling distribution of the sample mean:

```
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(turbine$vib_stress, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(sample = my_mean)) + stat_qq() + stat_qq_line()
```



which is *very* close to the line, with a very slightly short lower tail.

(20) (3 points) A test is shown in Figure 12. What code produced this output?

Specifically, this:

```
with(turbine, t.test(vib_stress, mu = 14.5, alternative = "less"))
```

```
    One Sample t-test

data:  vib_stress
t = -2.0791, df = 9, p-value = 0.03368
alternative hypothesis: true mean is less than 14.5
95 percent confidence interval:
     -Inf 14.12142
sample estimates:
mean of x
     11.3
```

This is also acceptable:

```
t.test(turbine$vib_stress, mu = 14.5, alternative = "less")
```

```
    One Sample t-test

data:  turbine$vib_stress
t = -2.0791, df = 9, p-value = 0.03368
alternative hypothesis: true mean is less than 14.5
95 percent confidence interval:
     -Inf 14.12142
sample estimates:
mean of x
     11.3
```

but in any case, you have to have some way of saying which dataframe the vibratory stress values came from.

(The output in the second case is very slightly different, but I'm not expecting you to be able to determine which code produces *precisely* that output.)

(21) (3 points) What do you conclude from Figure 12, in the context of the data?

The null hypothesis is that the mean vibratory stress is 14.5, and the alternative is that it is less than 14.5 (make sure you say that, and do not imply "different from"). One point for stating or strongly implying knowledge of both hypotheses.

The P-value is 0.034, which is less than 0.05, so we reject the null hypothesis in favour of the alternative (the second point) and conclude that the mean vibratory stress (of all turbine blades of which these are a sample) is in fact less than 14.5. (The third point.)

(22) (2 points) If you can, give a 95% confidence interval for the population mean vibratory stress. If you cannot, on the basis of the information you have, explain why not and describe what you would do to obtain it.

The first thing to observe is that confidence intervals are two-sided, so doing a one-sided test is not going to get one. (The other clue to that is that the interval goes down to negative infinity.)

To get the interval, you would re-run the test two-sided: that is, take the `alternative = "less"` out. Of course, you won't get the output below:

```
with(turbine, t.test(vib_stress, mu = 14.5))
```

```
    One Sample t-test

data:  vib_stress
t = -2.0791, df = 9, p-value = 0.06737
alternative hypothesis: true mean is not equal to 14.5
95 percent confidence interval:
  7.818222 14.781778
sample estimates:
mean of x
     11.3
```

You can optionally also remove the `mu`, because you're going to ignore the P-value anyway.

Extra: this is one of those in-between ones where the one-sided test is significant but the two-sided test is not, and hence the null mean of 14.5 is actually just *inside* the confidence interval. I wouldn't ask you about that on an exam, but it is certainly something to think about afterwards.

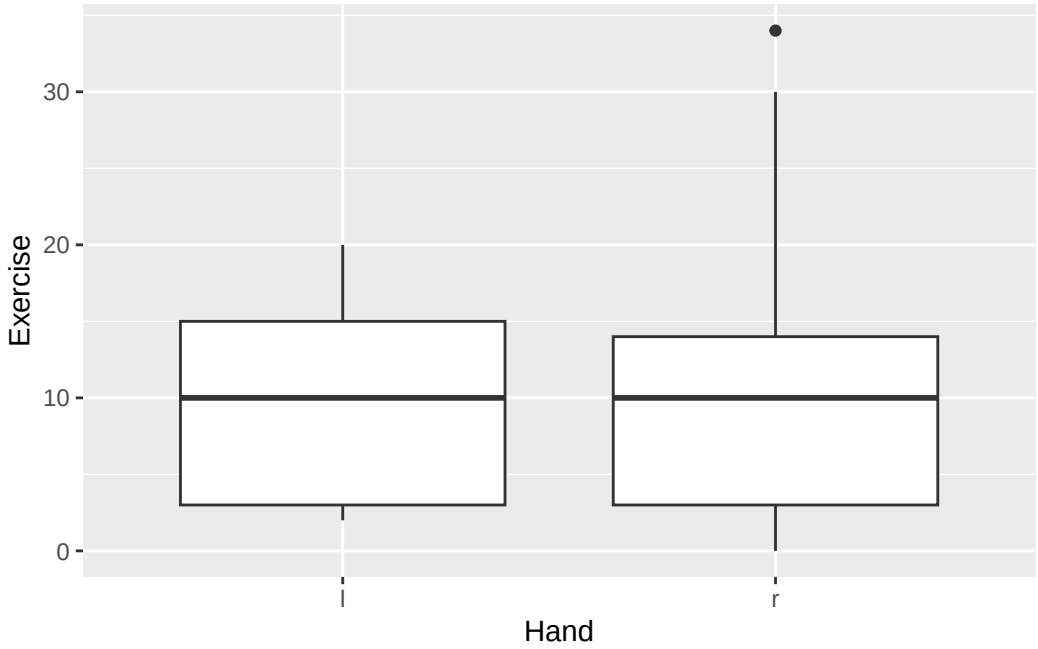
Exercise hours

Do left-handed or right-handed people exercise more? To find out, a survey was carried out on some randomly-chosen students. Some of the data is shown in Figure 13. The variables are described in Figure 14. The dataframe is called `ExerciseHours`.

(23) (2 points) What code would draw a suitable boxplot to assess the question of interest?

According to Figure 14, the two variables of interest are `Hand` (categorical) and `Exercise` (quantitative), and so the boxplot code will look like this:

```
ggplot(ExerciseHours, aes(x = Hand, y = Exercise)) + geom_boxplot()
```



(24) (2 points) A numerical summary is shown in Figure 15. The names of the columns in the summary are meant to suggest what they are. What code was used to make this summary?

This was what I used:

```
ExerciseHours %>%
  group_by(Hand) %>%
  summarize(n = n(), mean_ex = mean(Exercise), sd_ex = sd(Exercise))
```

Hand	n	mean_ex	sd_ex
l	9	9.111111	6.827233
r	41	10.926829	8.325834

The two variables you used in your boxplot were `Hand` (categorical) and `Exercise` (quantitative), so there should be a group-by of the first one, and you need to figure out how to make the three summaries: `n()` for the number of rows (observations) in each group, and `mean` and `sd` will get the numerical summaries.

(25) (2 points) Two different tests are shown in Figure 16 and Figure 17. Which one of these tests is more appropriate here? Explain briefly.

These are t-tests, the first one being Welch and the second one pooled. These are both assuming that normality is good enough; they differ according to whether they assume the spreads are equal enough. Use the summary in Figure 15 to decide for yourself whether the spreads are equal enough or not. If you think they are (they look a bit different but are actually not far apart), choose the pooled test; if not, choose the Welch test.

The points are for the explanation; if the explanation is good enough, you can come to either conclusion.

Another way to get to an answer is to look at the P-values for the two tests: 0.50 for the Welch test, 0.55 for the pooled test. These are not far apart, and both clearly lead to the same decision (next question), so from that point of view it does not matter which test you do, because the answer is going to be the same either way.

You should *never* choose a test on the basis of it having a smaller P-value (ie., the Welch test here). If you go through your scientific career doing that, you will reject your null hypotheses more often than you should. Tests work by picking *one* (for a reason external to the P-value), and then going with whatever it says. If the P-values had been more different in this case (say, one of them said to reject), then it would matter very much which test you did, and you would need to have a good non-P-value reason for choosing one over the other.

(26) (2 points) What do you conclude from your chosen test, in the context of the data?

The null hypothesis is that left-handers and right-handers have the same mean number of hours of exercise, and the alternative hypothesis is that the means are different (two-sided because of the lack of an `alternative` in the code).

The P-value is 0.50 (Welch) or 0.55 (pooled), so you would not reject the null hypothesis, and therefore there is no evidence of a difference in mean number of exercise hours between left-handers and right-handers. (Or, "we conclude that there is no difference", likewise.)

Make sure you state your P-value, because your reader might have a different α value than you and they need to know whether *they* should be rejecting (which they would if their α was 0.10 and the P-value happened to be 0.08, but they would not if the P-value is 0.50 (or 0.55) and you say that).

This is probably not very surprising: there seems to be no practical reason why exercise amounts would differ between left-handed and right-handed people.

There was a lot of confusion about the null hypothesis for the two-sample test. It is either one of these two things:

- the mean exercise times for left-handed and right-handed people are *equal*
- the difference between mean exercise times for left-handed and right-handed people is *zero*

A lot of people tried to squeeze both of these into their null, along the lines of "the mean differences for the two groups are equal".

Things that will cost you a half point include:

- not saying what the P-value *is*
- not saying what the null hypothesis is (unless you imply by your answer that you know what it is)
- not saying *what* is equal in your null hypothesis
- the confusion above about saying "the mean differences are equal"
- not making a conclusion about *mean* exercise times (actual exercise times will vary)
- talking about differences (or lack thereof) between "groups L and R", without saying that these are left-handed and right-handed people

Extra: In fact, if you go back and look at my boxplot above, it looks as if the medians are actually *identical*, so it is not surprising that the means are very close together as well.

(27) (2 points) Normal quantile plots of the relevant variables are shown in Figure 18. Use these plots to argue *in favour* of using a t-test here.

You need to argue that *both* distributions are normal enough given their sample sizes (which you can see from Figure 15 or by eyeballing the number of dots on each plot).

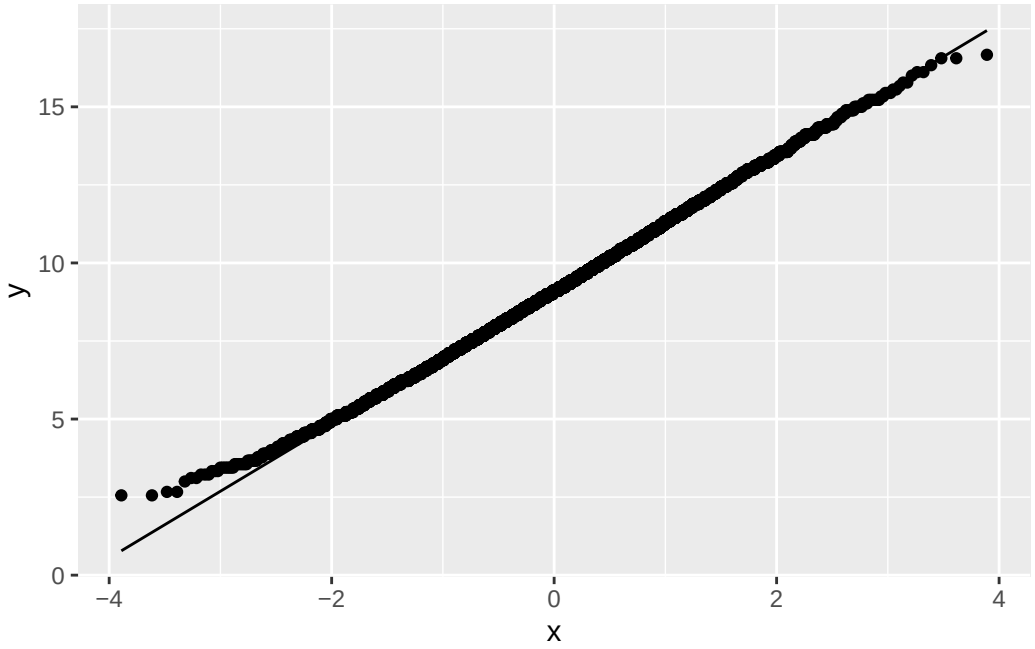
- The distribution of exercise hours for left-handed people (on the left) is pretty close to normal already, even with the small sample size ($n = 9$). So the small sample size is not a problem.
- The distribution of exercise hours for right-handed people (on the right) is noticeably right-skewed (curved pattern). (The horizontal patches are from several students reporting the same number of exercise hours.) That said, the sample size is large ($n = 41$), so the Central Limit Theorem can be expected to help a lot, and the skewed distribution may not be a problem. (See the Extra for more discussion.)

Where you need to get to, in brief, is something like this:

- the distribution for left-handers is close to normal, so we don't need to worry about the sample size.
- the distribution for right-handers is not normal (skewed to right), but the sample size is big, which may be enough to overcome the non-normal shape.

Extra: The way to get the "right" answer about whether the samples are normal and/or big enough is to look at a bootstrap sampling distribution of the sample mean. With a two-sample test, you look first at the sample you are most worried about. For me, the more problematic sample is the small one of left-handers; the right-hander distribution is less normal, but I anticipate that the much larger sample size will take care of that. Hence:

```
ExerciseHours %>% filter(Hand == "l") -> lefts
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(lefts$Exercise, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(sample = my_mean)) + stat_qq() + stat_qq_line()
```

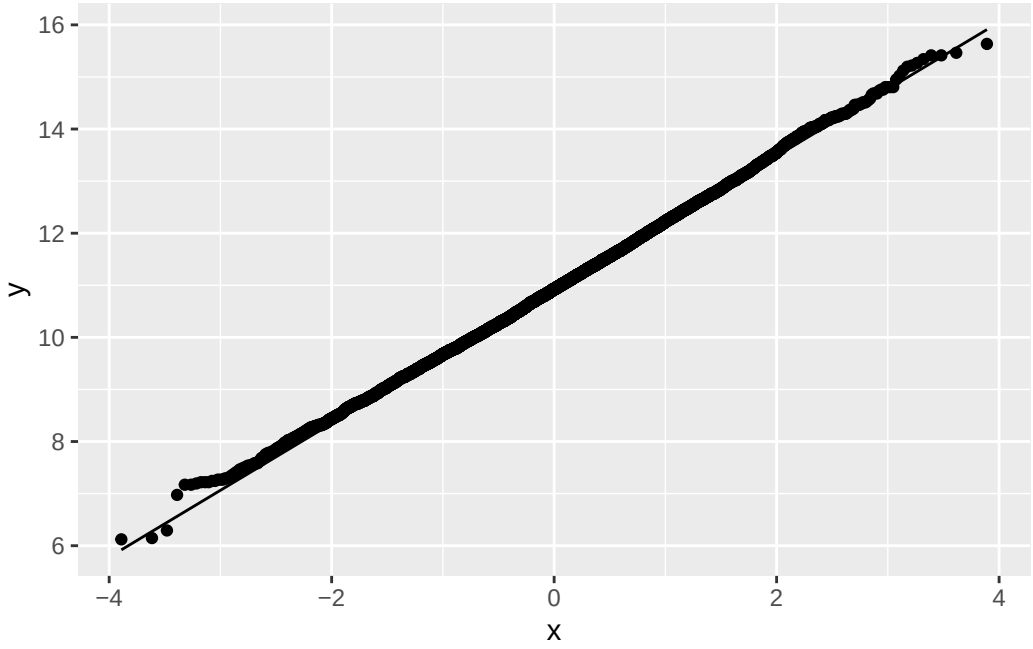


This, even with a sample size of 9, is really very good.

If you are still concerned about the right-handers, well, you have all the code, so you can copy, paste, and judiciously edit:

```
ExerciseHours %>% filter(Hand == "r") -> rights
tibble(sim = 1:10000) %>%
  rowwise() %>%
```

```
  mutate(my_sample = list(sample(rights$Exercise, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(sample = my_mean)) + stat_qq() + stat_qq_line()
```



and you see that the large sample size (in Central Limit Theorem terms) has taken care of business.

Nerve pulse waiting times

Two hundred waiting times (in milliseconds) were recorded for successive pulses along a nerve fibre. Some of the data are shown in Figure 19. The dataframe is called `nerve` and has one column called `wait`. The researchers who collected the data are interested in making inferences about the mean waiting time.

(28) (2 points) What do you conclude from Figure 20, in the context of the data? Explain briefly.

This is a normal quantile plot of the waiting times. It has a curved shape, opening upwards. Hence the lower values are bunched up, and the higher ones are spread out, and the distribution of waiting times is skewed to the right.

(29) (2 points) Why does it make practical sense that the distribution of these data would have the shape it does? Explain briefly.

These are waiting times, which cannot be less than zero (and there are in fact a lot of data values close to zero), but there is no upper limit. So, even without seeing the data, we would have guessed that the distribution would be skewed to the right (as it is).

(30) (3 points) What do you conclude from Figure 21? Your answer should clearly indicate that you know what this graph is, what that tells the researchers, and why the conclusion might be a surprise to someone not well trained in Statistics.

The code above the graph indicates that this is a bootstrap sampling distribution of the sample mean. This distribution is very close to normal (one point for all of that), and hence that obtaining a t-test (or t confidence interval) for the mean waiting time is entirely appropriate (the second point).

This might be a surprising conclusion to someone who looks at Figure 20 and sees how non-normal the data distribution is. They might say "not normal, therefore cannot run a t-test", but they would be wrong because they have not considered the sample size. Figure 21 indicates that the sample size ($n = 200$, as you can tell from Figure 19) is very much large enough to overcome the right-skewness in the distribution of waiting times, or, equivalently, the Central Limit Theorem works very well with this big of a sample. The third point for something resembling this.

If you need any more space, use the remainder of this page, labelling each answer with the question number it belongs to.

Figures

```
library(tidyverse)
```

Figure 1: Packages loaded

```
year|state|acres|yield
1948|Alabama|51000|21.5
1955|New Jersey|34000|19
1958|Oklahoma|48000|21
1938|North Carolina|162000|11.5
1986|North Dakota|470000|35
1939|South Carolina|13000|6.7
1957|Michigan|246000|22
1998|Texas|270000|22
1949|Illinois|3287000|25.5
1990|Virginia|525000|32
1975|South Carolina|1380000|22
2006|Louisiana|840000|36
1934|Michigan|8000|10.5
1928|Delaware|13000|17
1983|North Dakota|530000|27
1947|North Dakota|6000|10
1925|Indiana|40000|10
1937|Indiana|341000|17
2011|Pennsylvania|490000|44
```

Figure 2: Soybean data file, 20 randomly chosen lines

jnt1	jnt2	species
195	123	heik
217	122	heik
211	129	heik
201	130	conc
199	124	heik
192	116	heik
201	114	heik
186	107	heik
174	131	conc
182	130	conc

Figure 3: Flea beetle data (10 randomly chosen rows out of 26 rows total)

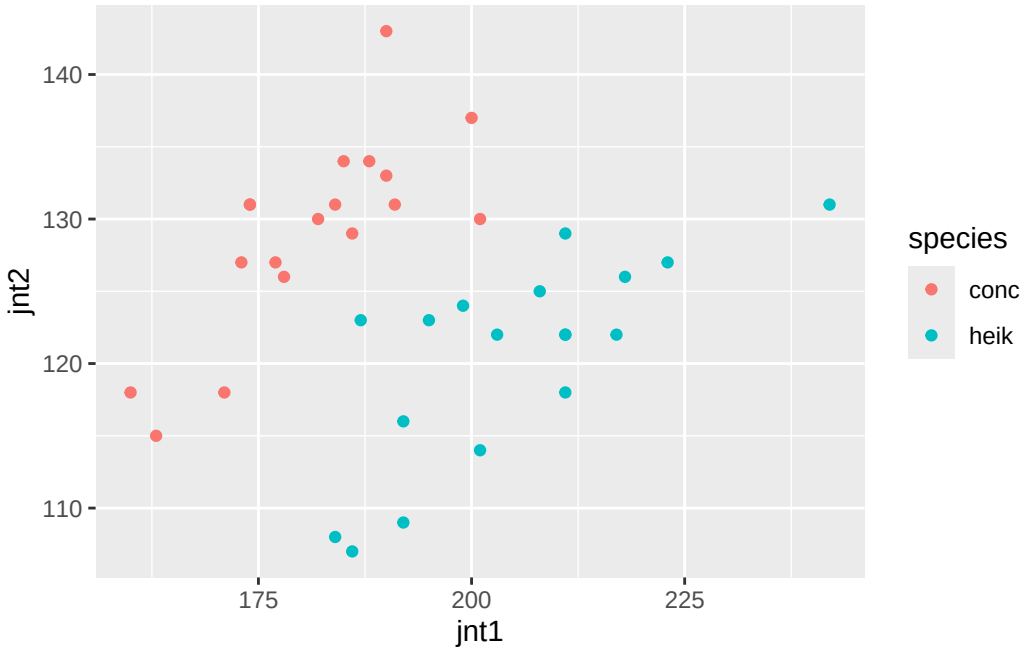


Figure 4: Flea beetle plot

cultivar	Alcohol	Colour
barolo	14.10	6.200000
barolo	14.39	5.250000
barbera	13.08	9.400000
grignolino	12.51	2.940000
grignolino	12.08	2.400000
barolo	13.73	5.700000
barbera	13.49	5.700000
barbera	13.52	4.350000
barbera	12.77	9.899999
barbera	13.36	5.600000
barbera	13.17	7.900000
grignolino	12.33	3.400000
barolo	13.05	7.200000
grignolino	12.33	3.270000
grignolino	12.37	4.500000
barolo	14.06	5.650000
grignolino	12.07	2.760000
barolo	13.20	4.380000
grignolino	11.64	2.800000
barolo	12.93	4.500000

Figure 5: Wine cultivar data (some)

```
ggplot(wine, aes(x = cultivar, y = Colour)) + geom_boxplot()
```

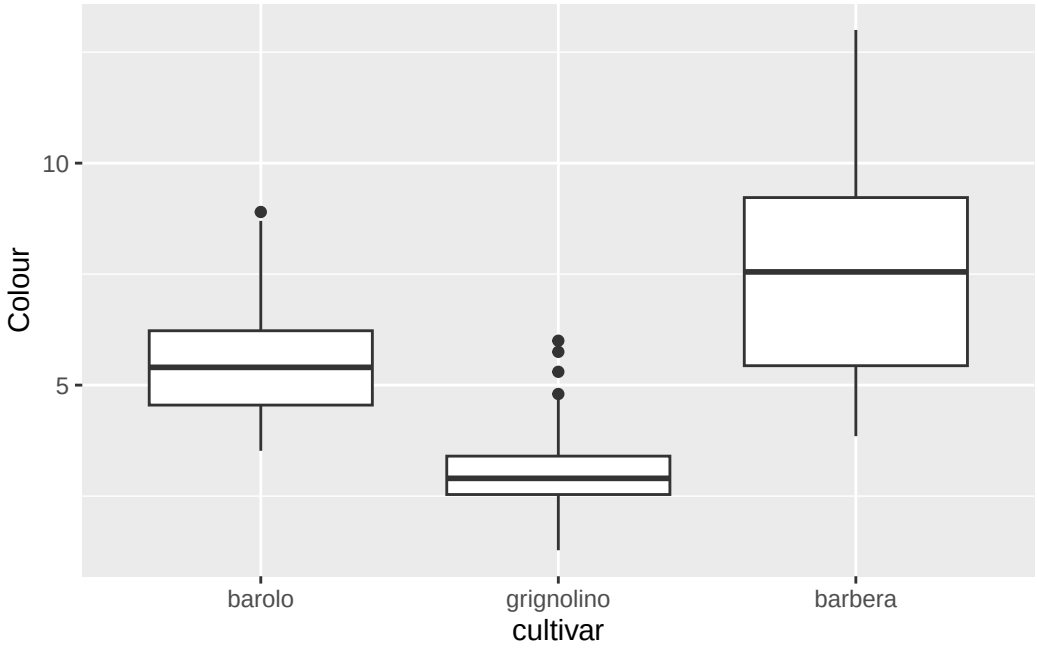


Figure 6: Wine cultivar plot 1

```
ggplot(wine, aes(x = Colour, y = Alcohol)) + geom_point()
```

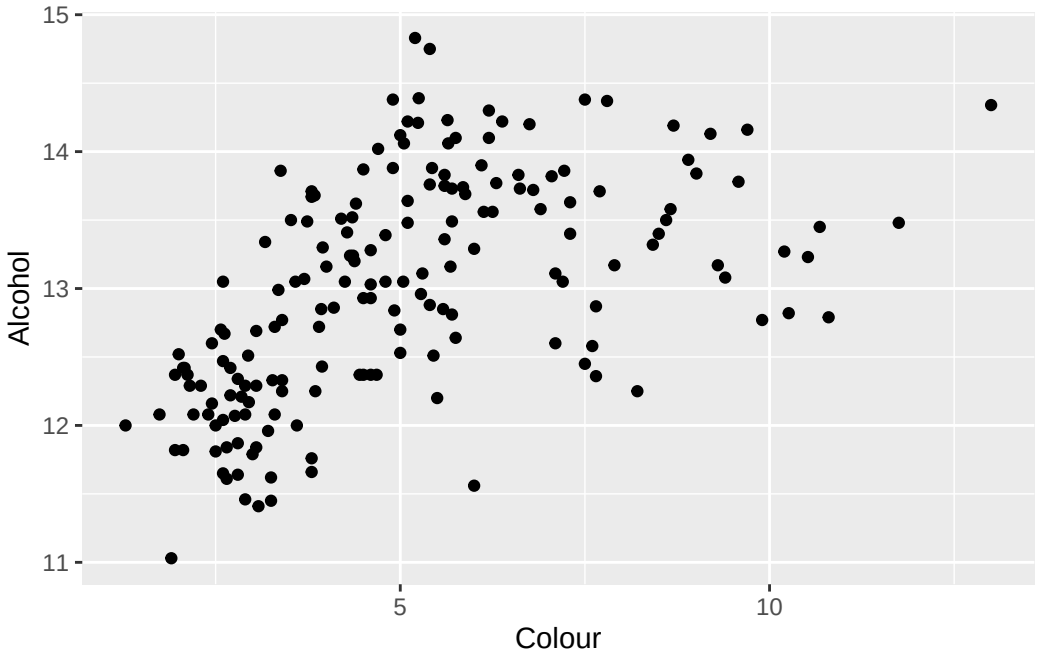


Figure 7: Wine cultivar plot 2

```
ggplot(wine, aes(x = Colour, y = Alcohol, colour = cultivar)) + geom_point()
```

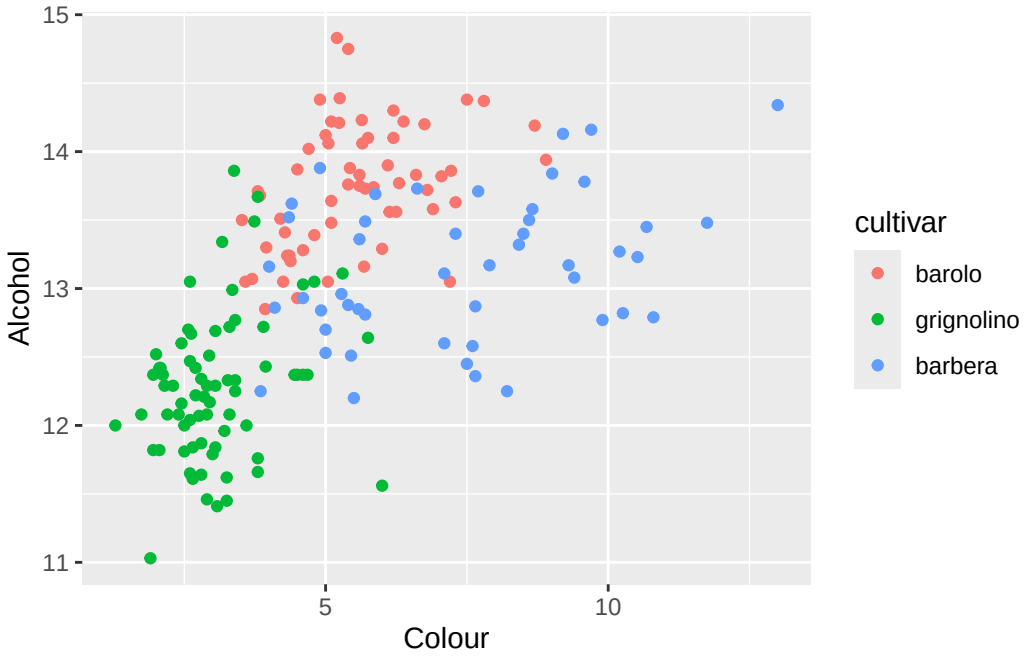


Figure 8: Wine cultivar plot 3

```
beetle %>% summarize(correl = cor(jnt1, jnt2))
```

correl
0.0960039

```
beetle %>% group_by(species) %>%
  summarize(correl = cor(jnt1, jnt2))
```

species	correl
conc	0.7690909
heik	0.7380546

Figure 9: Correlations for flea beetle data

```
turbine
```

vib_stress
16.88
10.23
4.59
6.66
13.68
14.23
19.87
9.40
6.51
10.95

Figure 10: Turbine blade stress data (all), in suitable units

```
ggplot(turbine, aes(sample = vib_stress)) + stat_qq() + stat_qq_line()
```

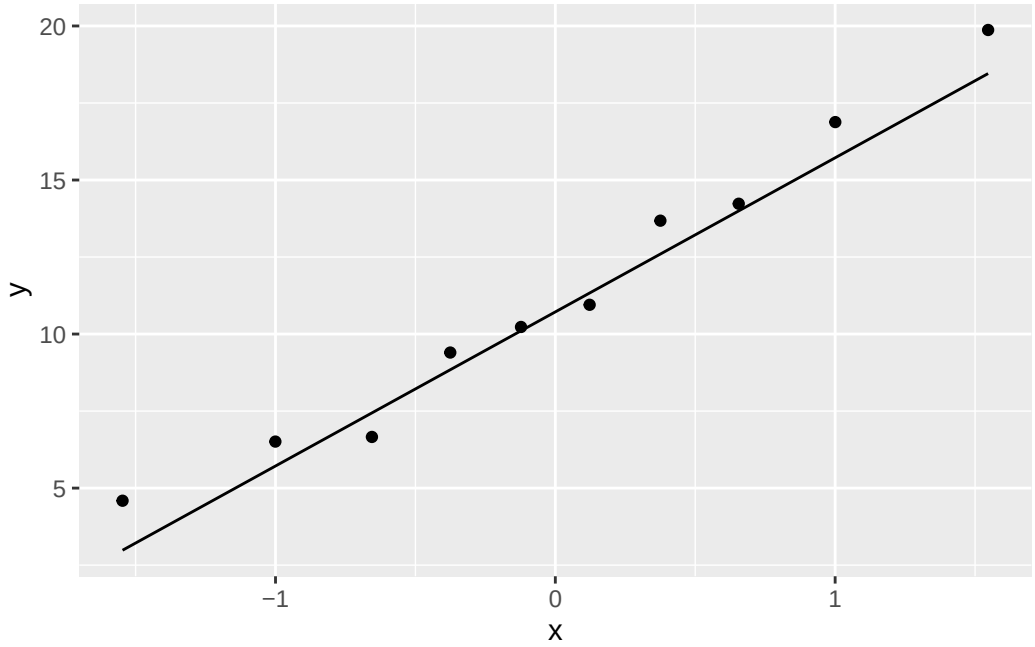


Figure 11: Turbine stress data, graph

```
    One Sample t-test

data:  vib_stress
t = -2.0791, df = 9, p-value = 0.03368
alternative hypothesis: true mean is less than 14.5
95 percent confidence interval:
     -Inf 14.12142
sample estimates:
mean of x
     11.3
```

Figure 12: Turbine stress, t-test

Year	Sex	Hand	Exercise	TV	Pulse	Pierces
3	F	r	8	0	50	6
2	M	r	14	4	62	0
1	F	r	20	6	58	6
1	F	l	10	5	66	3
1	F	r	34	4	71	2
1	M	r	8	2	52	0
4	M	r	2	2	59	0
1	M	r	20	12	75	0
1	M	l	2	0	74	0
2	M	l	20	14	70	0

Figure 13: Exercise hours data (10 randomly sampled rows out of 50 total)

The variables in the `ExerciseHours` data set are:

- `Year` in school: 1 (first year) through 4 (fourth year)
- `Sex` given here as female (`F`) or male (`M`)
- `Hand`: left-handed (`l`) or right-handed (`r`)
- `Exercise`: hours of exercise per week
- `TV`: hours of TV-watching per week (or equivalent such as Netflix)
- `Pulse`: resting pulse rate (beats per minute)
- `Pierces`: number of body piercings such as earrings

Figure 14: Exercise hours data description

Hand	n	mean_ex	sd_ex
l	9	9.111111	6.827233
r	41	10.926829	8.325834

Figure 15: Exercise hours data summary

```
t.test(Exercise ~ Hand, data = ExerciseHours)
```

```
    Welch Two Sample t-test

data:  Exercise by Hand
t = -0.69275, df = 13.782, p-value = 0.5
alternative hypothesis: true difference in means between group l and group r is not equal t
95 percent confidence interval:
 -7.445590  3.814154
sample estimates:
mean in group l mean in group r
       9.111111       10.926829
```

Figure 16: Exercise hours, t-test 1. The cut-off text here and in Figure 17 says "is not equal to 0".

```
t.test(Exercise ~ Hand, data = ExerciseHours, var.equal = TRUE)
```

```
    Two Sample t-test

data:  Exercise by Hand
t = -0.60931, df = 48, p-value = 0.5452
alternative hypothesis: true difference in means between group l and group r is not equal t
95 percent confidence interval:
 -7.807292  4.175856
sample estimates:
mean in group l mean in group r
       9.111111       10.926829
```

Figure 17: Exercise hours, t-test 2

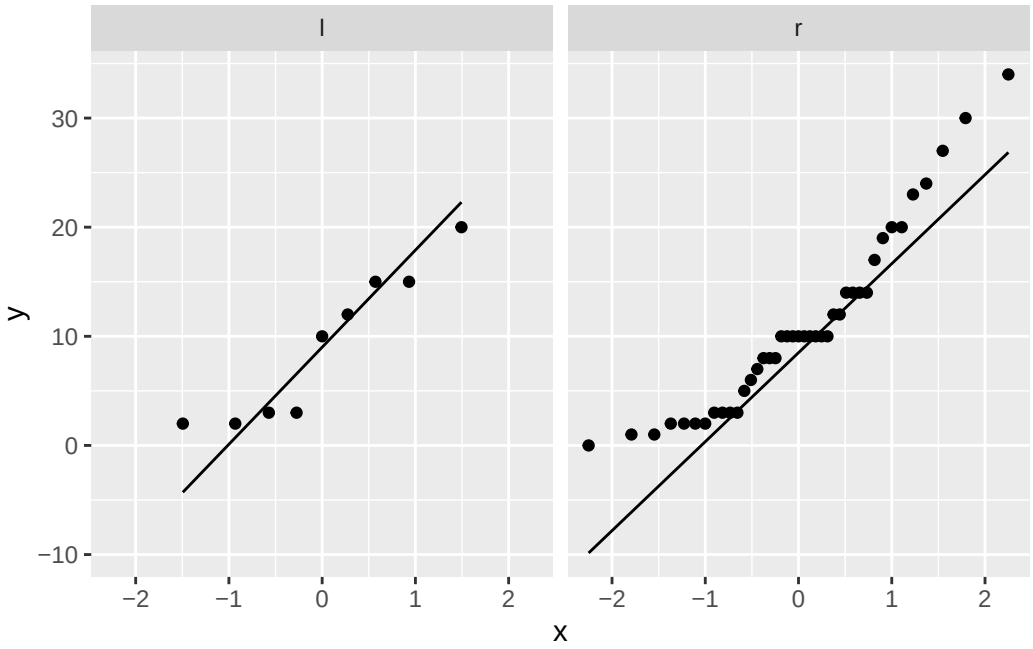


Figure 18: Exercise hours, normal quantile plots

wait
0.10
0.27
0.17
0.04
0.24
0.10
0.17
0.38
0.01
0.06

Figure 19: Nerve data (10 randomly chosen rows)

```
ggplot(nerve, aes(sample = wait)) + stat_qq() + stat_qq_line()
```

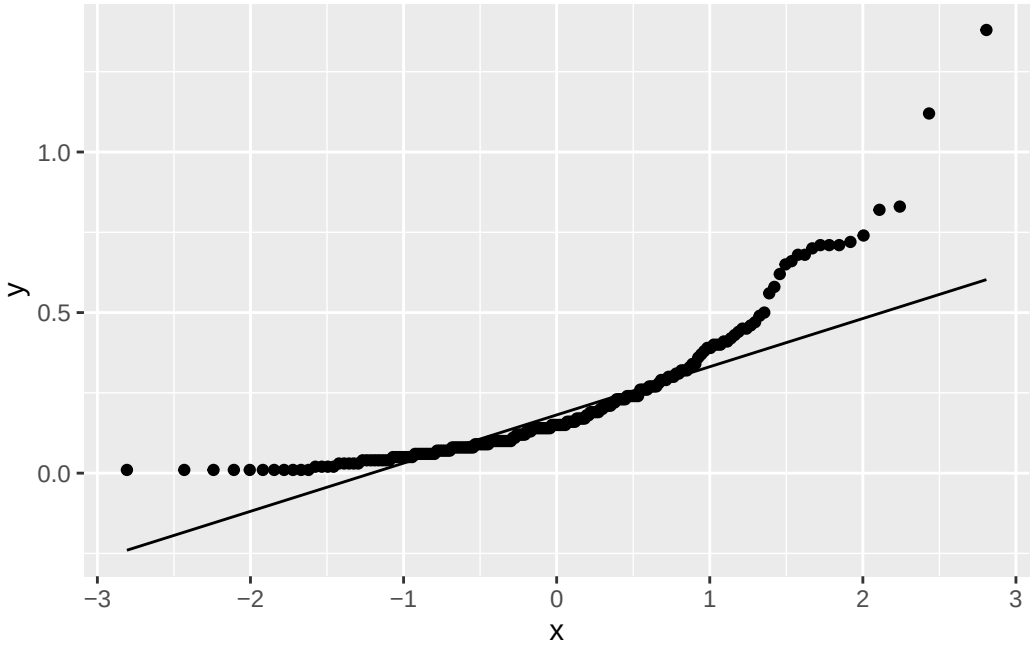


Figure 20: Nerve data plot

```
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(nerve$wait, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(sample = my_mean)) + stat_qq() + stat_qq_line()
```

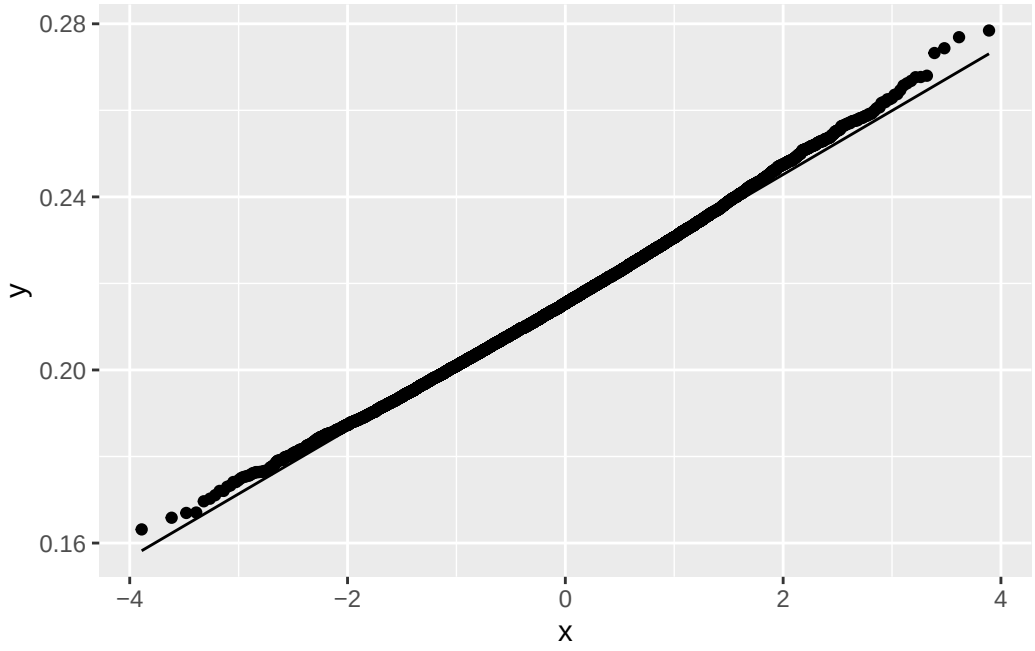


Figure 21: Nerve data, another plot (and the code that made it)