University of Toronto Scarborough
Department of Computer and Mathematical Sciences
STAC32 (K. Butler), Midterm Exam
October 4, 2023

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 xx 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.

1. When you are found guilty of wrongdoing, does it help to smile at the person who decides what your punishment is? An experiment was conducted to investigate this. Participants in the experiment pretended to be members of a college disciplinary panel judging students accused of cheating. For each suspect, along with a description of the offence, a picture was provided with either a smile or neutral facial expression. Each participant said what they thought was a suitable punishment based on the evidence they had seen and a leniency score was calculated based on the disciplinary decisions made by the participants. (A higher leniency score means a *smaller* punishment.)

   The data file is shown in Figure 2, and is in the file `smiles.txt` in the same folder as your current R Studio project.

   (a) [3] What R code would read the data from the file into a dataframe called `smiles` and display (at least some of) that dataframe?

My answer:

These are aligned columns with variable numbers of spaces in between, and `read_table` is what we used to read in this kind of thing:

```
smiles <- read_table("smiles.txt")
```

```
-- Column specification --------------------------------------------------------
cols(
  Group = col_character(),
  Leniency = col_double()
)
```

```
smiles
```

```
# A tibble: 20 x 2
   Group   Leniency
   <chr>      <dbl>
 1 neutral      6
 2 smile        3.5
 3 smile        4.5
 4 smile        6
 5 smile        4
 6 neutral      2.5
 7 smile        7.5
 8 smile        2.5
 9 smile        3.5
10 neutral      4
11 neutral      2.5
12 neutral      4.5
13 smile        3.5
14 smile        9
```

```
15 neutral      3
16 smile        3
17 smile        5
18 neutral      4.5
19 smile        5.5
20 smile        5
```

Don't forget to add the name of the dataframe to display it! (In this course we make a point of displaying the dataframe, or of otherwise looking at it, after reading it in, on the basis that in the real world we will want to convince ourselves that we have read in the right thing.)

Any other way of displaying at least some of the dataframe, such as `glimpse` (that I used with the Blue Jays data in lecture) or `slice`-ing off the first few rows, is acceptable, but just adding the name of the dataframe is definitely enough.

Because the number of spaces between the group name and the leniency score is not constant, any kind of `read_delim` is not going to work.

Two points for correct code to read the file, and one point for code to display the dataframe in some reasonable way. `read_delim` is only 0.5 out of two even if you get the filename correct.

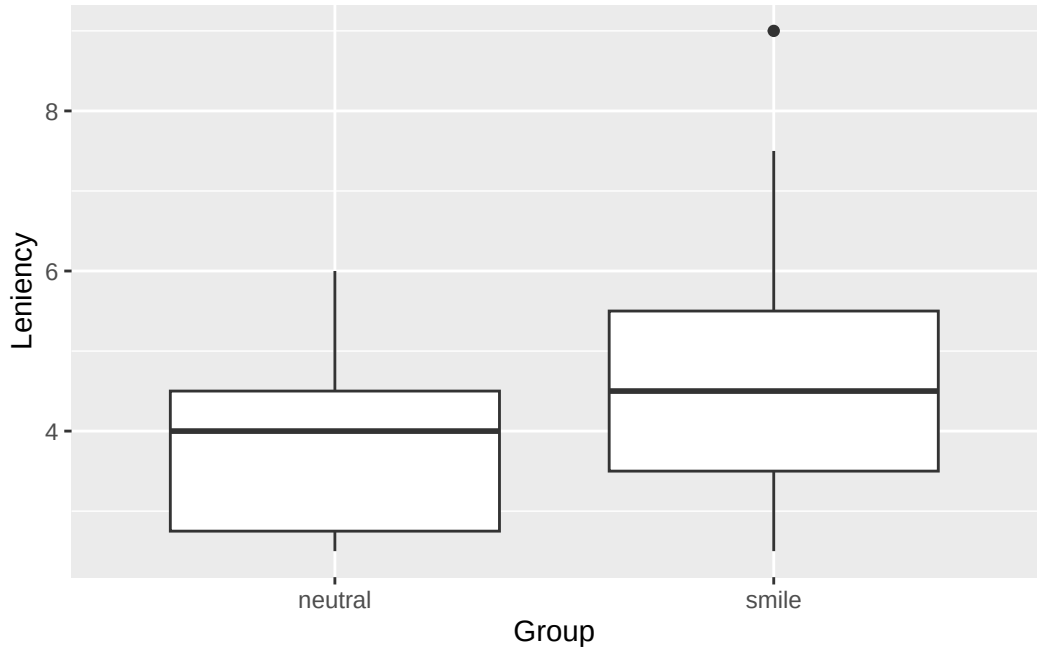
(b) [3] What code will make a suitable graph of the two variables in your dataframe? Justify your choice of graph briefly.

My answer:

One categorical variable `Group` and one quantitative one `Leniency`, so the obvious thing is a boxplot. One point for something like that, including "boxplot" and saying which variable is categorical and which is quantitative (to show that you know). The other two points are for the graph code; see below for the scale.

The plot is this:

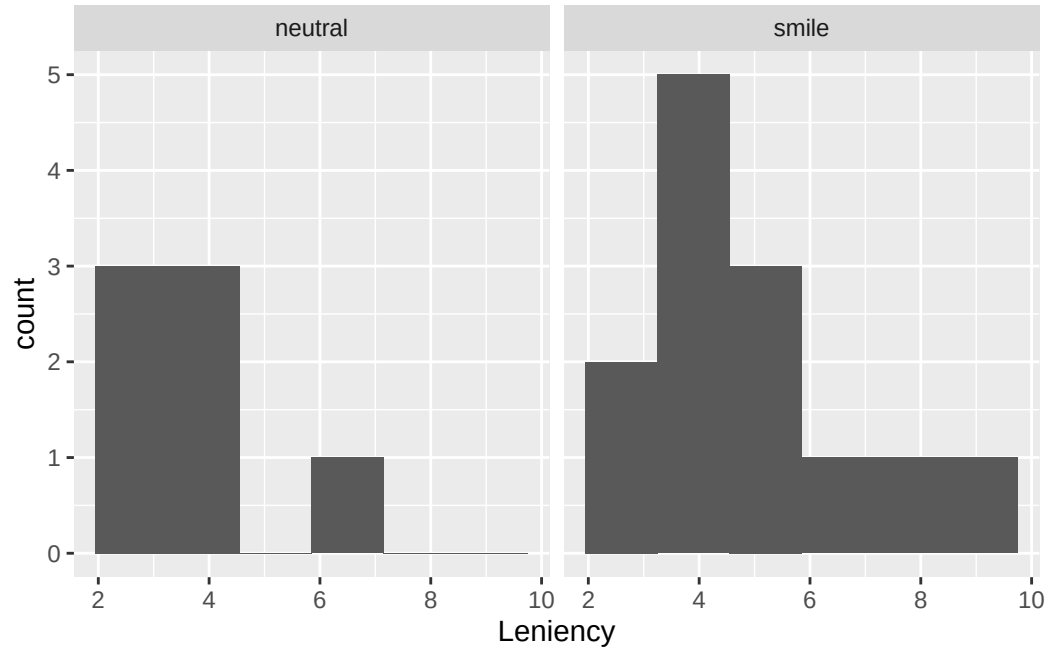
```
ggplot(smiles, aes(x = Group, y = Leniency)) + geom_boxplot()
```



The aim of the study was to compare leniency scores for the two groups of supposed student offenders, and that is what the boxplot does, so I think this is the best graph.

As a (slightly inferior) graph, you might take the attitude that the key variable was `Leniency` and that `Group` is an "extra categorical variable", so that you could do histograms of `Leniency` facetted by `Group`:

```
ggplot(smiles, aes(x = Leniency)) + geom_histogram(bins = 6) +
  facet_wrap(~ Group)
```

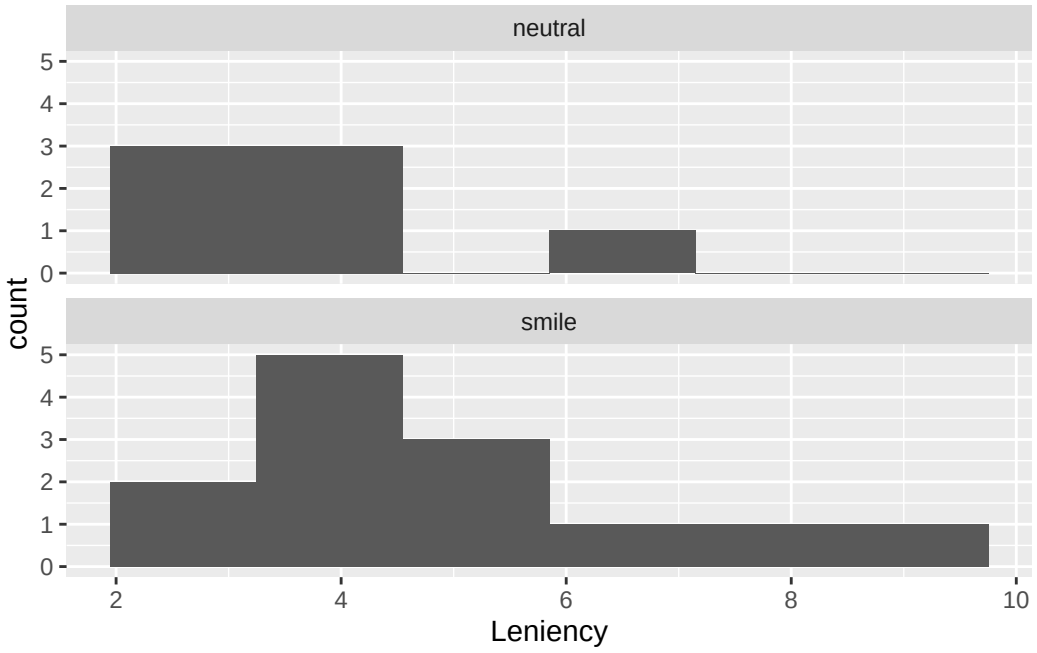


Remember in these ones, you don't mention the variable you're going to facet by until right at the end.

I think it's better to use a few more bins than you normally would, on the basis that the bins are used for *both* histograms and (as it turns out) the `neutral` values don't go up as far and you don't get much of a picture of their shape with, say, 6 bins.

On the basis that we really want to *compare* these two histograms, it is better to put them above and below with a common `Leniency` scale, thus, using one "column of plots":

```
ggplot(smiles, aes(x = Leniency)) + geom_histogram(bins = 6) +
  facet_wrap(~ Group, ncol = 1)
```



Even with this, though, it's not very easy to compare "typical" leniency scores between the two groups. If you were assessing the *shapes* of these two distributions, you could make a case for this being the best graph (boxplots don't show shape very well), but the aim of the study was to see whether punishments are less (leniency scores are higher) on average for the `smile` group, and for that a boxplot is a better graph.

For the graphing part, two points for a boxplot, one point for an above-and-below histogram, 0.5 for a side-by-side histogram, less any deductions for errors.

(c) [2] A graph is shown in Figure 3. This may or may not be the same as the graph you gave code for earlier. What does this plot tell you that would be of interest to the researchers who designed this experiment? Explain briefly.

My answer:

Two things (a point each):

- the students who were smiling in their photo got a slightly higher leniency score (a lesser punishment) on average than the students with a neutral facial expression (the median is larger)
- the difference is small, or there is a lot of variability (the boxes are tall), or the difference is small relative to the amount of variability, or the difference is likely not large enough to be statistically significant. Or some relevant discussion about variability.

I was also prepared to accept some comments about the shapes of the distributions. There were some answers that got a bit close to "write down everything you can think of"; I would prefer you to exercise some judgement about what to write. The reason why the researchers would be interested in shape is that this would tell them whether or not to do a t-test, so it is really better to say this in your discussion.

I need you to interpret *this* boxplot, not boxplots in general. (This is the meaning of the "describe" comment.)

(Boxplots don't show means. If you say they do, expect to lose a half point.)

(d) [3] What code would work out the number of observations in each group, along with the mean leniency score of each group?

My answer:

This is not the kind of case where `count` will work, because you are calculating something else (the mean) along with doing the counting. So you have to use the `n()` idea, like this:

```
smiles %>%
  group_by(Group) %>%
  summarize(n = n(), mean_len = mean(Leniency))
```

```
# A tibble: 2 x 3
  Group       n mean_len
  <chr>   <int>    <dbl>
1 neutral     7     3.86
2 smile      13     4.81
```

Give the summaries whatever names you like, but `n` is a good name for the number of observations in it, and something with `mean` in it is a good name for the mean.

One point for the right `group_by`, and two points for a `summarize` with the right two things in it.

Extra: As it turns out, there are only seven observations in the `neutral` group (where the students had a neutral facial expression in their photo), with 13 in the `smile` group. This was not actually all of the original data; I took a random sample from that dataset, so that I could show you all of the data we used here (in Figure 2) to help you decide how you were going to read the values into a dataframe.

2. Pew Research Center conducted a survey in 2018, asking a sample of U.S. adults to categorize five factual and five opinion statements. This dataset provides data from this survey, with information on the age group of the participant as well as the number of factual and opinion statements they classified correctly (out of 5). Some of the data are shown in Figure 4. A total of 5,035 adults were surveyed altogether. The dataframe is called `fact_opinion`.

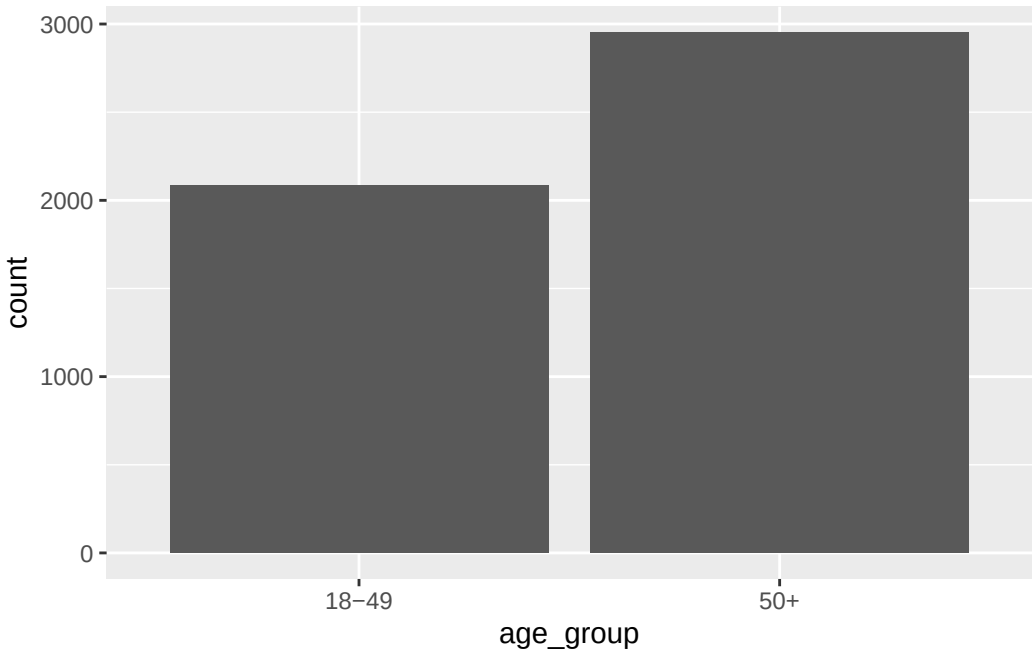
(a) [2] Suppose we want to make a graph that will enable us to see which age group has the most

respondents in the survey. What code will make such a graph?

My answer:

One categorical variable, `age_group`, so a bar chart:

```
ggplot(fact_opinion, aes(x = age_group)) + geom_bar()
```



Extra: there were only two age groups, and there were more respondents in the 50+ age group (taller bar).

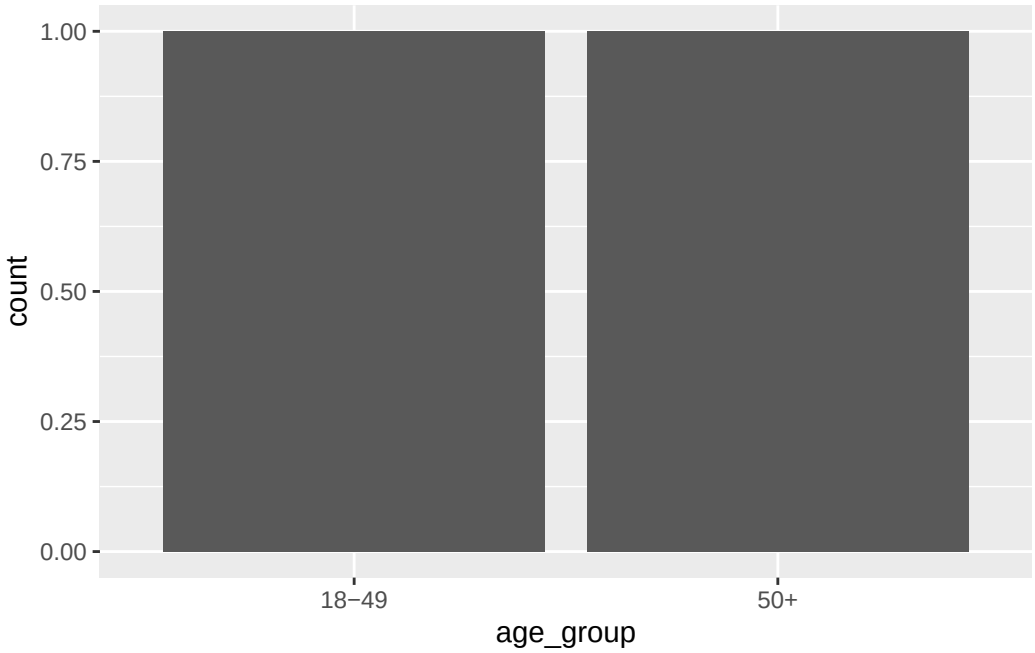
Points: minus a half point per small error (in the grader's opinion). If you try to draw some other kind of plot, half a point total is your maximum (and that's if your graph would work and has some relevance towards answering the question).

(b) [2] Suppose you run the code shown in Figure 5. What will happen? Explain briefly.

My answer:

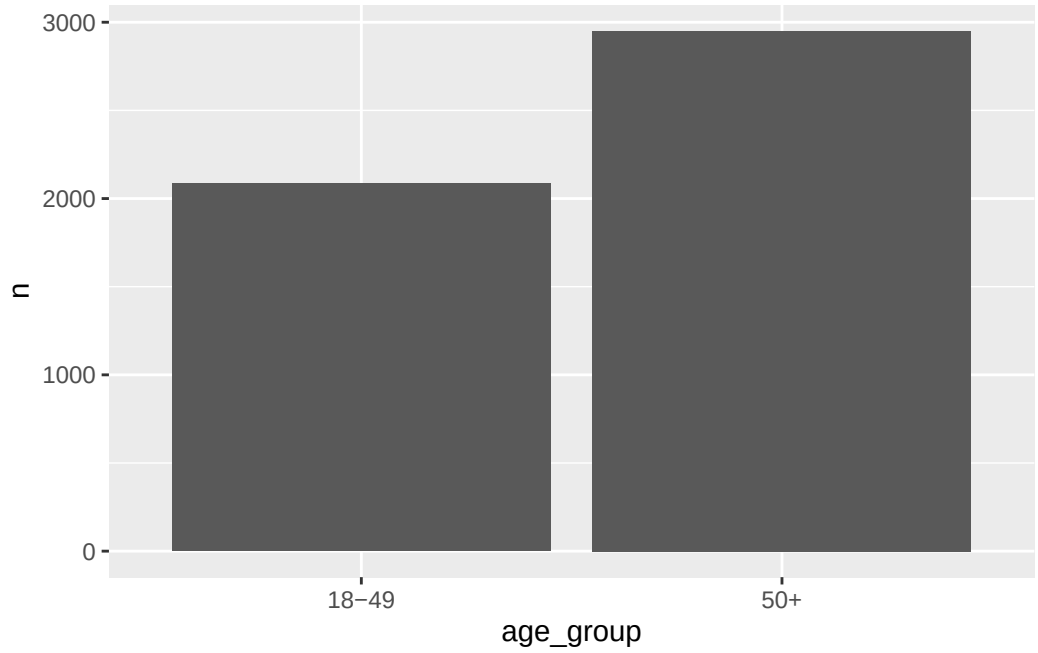
The dataframe `counted` has only two rows, one for each age group. What `geom_bar` does is to count the number of *rows* (observations) for each age group and it will get 1 for each one, so the bar chart will contain two bars, both of height *1*:

```
fact_opinion %>% count(age_group) -> counted
ggplot(counted, aes(x = age_group)) + geom_bar()
```



Extra: if you want to make a bar chart, but you already have values to use for the heights of the bars, instead of using `geom_bar`, use `geom_col`, which has an additional input `y` that is the height of each bar. Hence, this code gives you the same bar chart as we got before:

```
ggplot(counted, aes(x = age_group, y = n)) + geom_col()
```



because it uses our counts in `n` as the heights of the bars. I haven't shown you this in lecture, so it's not an acceptable answer to the previous part, but now you know that you can also draw a bar chart in two steps: by counting, and then using those counts in `geom_col`.

(c) [3] A plot is shown in Figure 6. What does this plot tell you about how the age groups differ? Explain briefly. (The variable on the x-axis, though quantitative, is treated as ordered categorical for this plot.)

My answer:

This graph compares the older and younger respondents according to how many factual statements they classified correctly (see the code above the plot) in a grouped bar chart.

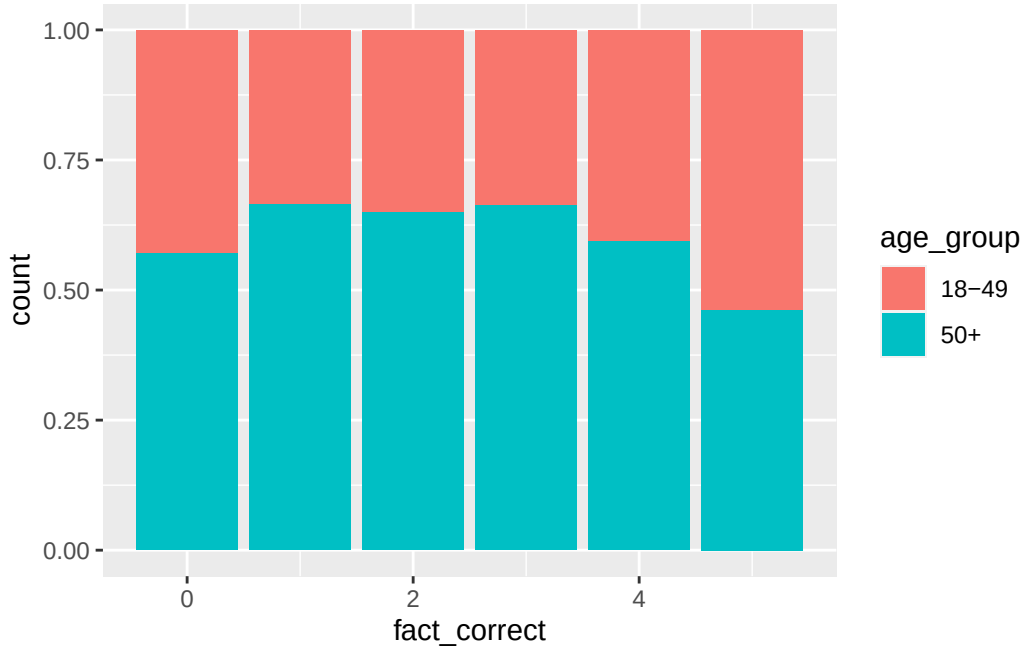
Usually, the blue bars are taller than the corresponding red ones (because there are more older respondents altogether), but look to see where this is not the case: at 5 correct, meaning that the younger people tended to get all 5 of the factual statements correct more than the older people, who were more likely to get somewhere between 1 and 4 correct.

Three points for something sufficiently close to this.

Noting that the blue bars are in general taller than the red ones says only that there are more older respondents than younger ones overall, which is less insightful than the above (we could have seen *this* by looking at a simple bar chart). Only one point if this is as far as you get. What you are looking for is where the *relative* heights of the bars differ.

Extra: we said above that there were more older respondents than younger ones; to account for this in our comparison, we might use `position = "fill"` instead:

```
ggplot(fact_opinion, aes(x = fact_correct, fill = age_group)) +
  geom_bar(position = "fill")
```



The fact that there are more older respondents than younger ones is why the graph is more blue than red overall. But to compare the age groups, look at where the red bars are relatively bigger: mainly at 4 and (especially) 5 correct, so relatively more younger people got 4 and 5 correct, and relatively more older people got only 3, 2, or 1 correct.

Extra 1: you will note that the younger people were also more likely to get *none* of the factual statements correct. I don't know what that means.

Extra 2: I am not quite sure from the information given with the dataset exactly *what* the respondents had to do. The most likely thing I can think of is that they were given *ten* statements, and were told that some of them were facts and some were opinions, and they had to say whether each of the ten statements was fact or opinion. If all that is true, each respondent would produce a contingency table like this:

```
          Choice
Truth     Fact   Opinion  Total
Fact        4       1       5
Opinion     2       3       5
```

```
Total       6       4      10
```

This is a frequency table, but one with additional structure: there are known (by the researcher) to be 5 facts and 5 opinions, so the rows must add up to 5. Normally, if you are planning to do something like a chi-squared test, the row and column totals could be anything. The numbers 4 and 3 down the diagonal of the table are `fact_correct` and `opinion_correct` in our data; a significant chi-squared test here would indicate "discrimination", but in the positive sense: the person or people you are looking at has (have) a better-than-chance ability to distinguish facts from opinions.

But I don't know, without further research, whether it was actually like that or not.

3. The US is divided into ten "health regions" that each contain several states. For each region, for males and females separately and for urban and rural residents separately, the regional mortality (death) rates from various causes are recorded. Some of the data are shown in Figure 7. The dataframe is called `mortality`.

   In the question parts below, give (only) code to display what is requested, unless otherwise stated.

   (a) [2] Display (only) the columns for health region, cause of death and death rate.

My answer:

As with all of these, you won't have the output. I include it to show that my code does indeed work.

```
mortality %>% select(Region, Cause, Rate)
```

```
# A tibble: 400 x 3
   Region        Cause          Rate
   <chr>         <chr>         <dbl>
 1 HHS Region 01 Heart disease  188.
 2 HHS Region 01 Heart disease  199.
 3 HHS Region 01 Heart disease  115.
 4 HHS Region 01 Heart disease  124.
 5 HHS Region 02 Heart disease  227.
 6 HHS Region 02 Heart disease  249.
 7 HHS Region 02 Heart disease  149.
 8 HHS Region 02 Heart disease  166.
 9 HHS Region 03 Heart disease  218.
10 HHS Region 03 Heart disease  246
# i 390 more rows
```

These columns are not consecutive, so you need to name them one by one. `Region:Rate` would also select the columns `Status` and `Sex` that you don't want. `select(Region, Cause:Rate)` is kind of weird, but it works, so full points. The column names have initial Capital Letters that need to appear in your answer; *this* part of `select` is case-sensitive because R is usually

case-sensitive. The place where case does not matter is with the select-helpers like `starts_with` that appear below.

Points for all of these: expect to lose a half point for a small error and a full point for a bigger one. The grader will endeavour to award you a score that reflects your progress towards a full solution. For example, 1 out of 2 means that you got about halfway towards a working answer.

(b) [2] Display the columns whose names begin with S (either uppercase or lowercase), without naming or numbering the columns in your code.

My answer:

Use the select-helper `starts_with`:

```
mortality %>% select(starts_with("S"))
```

```
# A tibble: 400 x 3
   Status Sex       SE
   <chr>  <chr>  <dbl>
 1 Urban  Male     1
 2 Rural  Male     2.6
 3 Urban  Female   0.6
 4 Rural  Female   1.7
 5 Urban  Male     0.8
 6 Rural  Male     3.3
 7 Urban  Female   0.5
 8 Rural  Female   2.3
 9 Urban  Male     0.8
10 Rural  Male     2
# i 390 more rows
```

Having the S as uppercase is optional; it will work the same with a lowercase S because `starts_with` is not case-sensitive. An `ignore.case` is an error, because you do want to ignore case (match uppercase or lowercase). Including `ignore.case = TRUE` is "correct" but unnecessary; including it will cost you a half point, because the case *is already* by default ignored.

(c) [2] Select the columns whose names have the letter A in them somewhere (uppercase or lowercase), without naming or numbering them.

My answer:

```
mortality %>% select(contains("A"))
```

```
# A tibble: 400 x 3
```

```
   Status Cause          Rate
   <chr>  <chr>         <dbl>
 1 Urban  Heart disease  188.
 2 Rural  Heart disease  199.
 3 Urban  Heart disease  115.
 4 Rural  Heart disease  124.
 5 Urban  Heart disease  227.
 6 Rural  Heart disease  249.
 7 Urban  Heart disease  149.
 8 Rural  Heart disease  166.
 9 Urban  Heart disease  218.
10 Rural  Heart disease  246
# i 390 more rows
```

Again, the A can be uppercase or lowercase. For the grader: if a student uses `ignore.case` again here, penalize only the first use and ignore others.

(d) [2] Select the categorical variables (that are text), again without naming or numbering them.

My answer:

Use `where(is.character)` to select the columns that have the property of being text:

```
mortality %>% select(where(is.character))
```

```
# A tibble: 400 x 4
   Region        Status Sex    Cause
   <chr>         <chr>  <chr>  <chr>
 1 HHS Region 01 Urban  Male   Heart disease
 2 HHS Region 01 Rural  Male   Heart disease
 3 HHS Region 01 Urban  Female Heart disease
 4 HHS Region 01 Rural  Female Heart disease
 5 HHS Region 02 Urban  Male   Heart disease
 6 HHS Region 02 Rural  Male   Heart disease
 7 HHS Region 02 Urban  Female Heart disease
 8 HHS Region 02 Rural  Female Heart disease
 9 HHS Region 03 Urban  Male   Heart disease
10 HHS Region 03 Rural  Male   Heart disease
# i 390 more rows
```

(e) [3] Display the rows that are for health region 04.

My answer:

`filter` to display rows rather than columns. Get the designation for health region right (it should match what is in `Region` in Figure 7):

```
mortality %>%
  filter(Region == "HHS Region 04")
```

```
# A tibble: 40 x 6
   Region        Status Sex    Cause              Rate    SE
   <chr>         <chr>  <chr>  <chr>             <dbl> <dbl>
 1 HHS Region 04 Urban  Male   Heart disease     213.    0.5
 2 HHS Region 04 Rural  Male   Heart disease     276.    1.3
 3 HHS Region 04 Urban  Female Heart disease     132.    0.4
 4 HHS Region 04 Rural  Female Heart disease     177.    0.9
 5 HHS Region 04 Urban  Male   Cancer            205.    0.5
 6 HHS Region 04 Rural  Male   Cancer            245.    1.2
 7 HHS Region 04 Urban  Female Cancer            140.    0.4
 8 HHS Region 04 Rural  Female Cancer            156.    0.8
 9 HHS Region 04 Urban  Male   Lower respiratory  48.5   0.3
10 HHS Region 04 Rural  Male   Lower respiratory  70.7   0.6
# i 30 more rows
```

(f) [3] For each of the health regions, display the median death rates from heart disease (but not the death rates from any other cause).

My answer:

Grab the heart disease rows first, throwing away the others, then do a group-by over regions:

```
mortality %>%
  filter(Cause == "Heart disease") %>%
  group_by(Region) %>%
  summarize(median_death_rate = median(Rate))
```

```
# A tibble: 10 x 2
   Region        median_death_rate
   <chr>                     <dbl>
 1 HHS Region 01              156.
 2 HHS Region 02              196.
 3 HHS Region 03              190.
 4 HHS Region 04              195.
 5 HHS Region 05              182.
 6 HHS Region 06              198.
```

```
 7 HHS Region 07              181.
 8 HHS Region 08              143.
 9 HHS Region 09              160.
10 HHS Region 10              145.
```

It also works to do a regular group-by and summarize and do the filter at the end, *as long as you include the cause of death in the grouping*, or else you won't be able to filter by that:

```
mortality %>%
  group_by(Region, Cause) %>%
  summarize(median_death_rate = median(Rate)) %>%
  filter(Cause == "Heart disease")
```

```
`summarise()` has grouped output by 'Region'. You can override using the
`.groups` argument.
```

```
# A tibble: 10 x 3
   Region        Cause         median_death_rate
   <chr>         <chr>                     <dbl>
 1 HHS Region 01 Heart disease              156.
 2 HHS Region 02 Heart disease              196.
 3 HHS Region 03 Heart disease              190.
 4 HHS Region 04 Heart disease              195.
 5 HHS Region 05 Heart disease              182.
 6 HHS Region 06 Heart disease              198.
 7 HHS Region 07 Heart disease              181.
 8 HHS Region 08 Heart disease              143.
 9 HHS Region 09 Heart disease              160.
10 HHS Region 10 Heart disease              145.
```

(g) [3] Display any death rates that are either over 230 or are for Cancer (or both), along with the cause of death.

My answer:

```
mortality %>%
  filter(Rate > 230 | Cause == "Cancer") %>%
  select(Cause, Rate)
```

```
# A tibble: 45 x 2
   Cause          Rate
   <chr>         <dbl>
 1 Heart disease  249.
 2 Heart disease  246
 3 Heart disease  276.
```

```
 4 Heart disease  266.
 5 Heart disease  238.
 6 Cancer         194.
 7 Cancer         203.
 8 Cancer         140.
 9 Cancer         145.
10 Cancer         188.
# i 35 more rows
```

You could do the `select` first, since the things you're selecting are also part of the `filter`.

(h) [4] Display the two lowest mortality rates and their accompanying causes of death for each health region (which you should also display), but only for females.

My answer:

Make sure you read the whole question, because the thing you do first is mentioned at the end: look only at the females. After that, group by region, find the two lowest mortality rates for each one (which will happen automatically with the `group_by`), then display what you want. You can either use `slice_min` (easiest) or sort and then slice. So either this:

```
mortality %>%
  filter(Sex == "Female") %>%
  group_by(Region) %>%
  slice_min(Rate, n = 2) %>%
  select(Region, Cause, Rate)
```

```
# A tibble: 20 x 3
   Region        Cause      Rate
   <chr>         <chr>     <dbl>
 1 HHS Region 01 Suicide     4.5
 2 HHS Region 01 Suicide     5.8
 3 HHS Region 02 Suicide     3.4
 4 HHS Region 02 Suicide     3.9
 5 HHS Region 03 Suicide     4.8
 6 HHS Region 03 Suicide     6.6
 7 HHS Region 04 Suicide     5.9
 8 HHS Region 04 Suicide     6
 9 HHS Region 05 Suicide     4.9
10 HHS Region 05 Suicide     5.1
11 HHS Region 06 Suicide     5.3
12 HHS Region 06 Suicide     6.4
13 HHS Region 07 Suicide     5.8
14 HHS Region 07 Suicide     5.8
15 HHS Region 08 Nephritis   8.3
```

```
16 HHS Region 08 Suicide     8.4
17 HHS Region 09 Suicide     5.2
18 HHS Region 09 Nephritis   6.1
19 HHS Region 10 Nephritis   5.9
20 HHS Region 10 Nephritis   6.7
```

or this:

```
mortality %>%
  filter(Sex == "Female") %>%
  group_by(Region) %>%
  arrange(Rate) %>%
  slice(1:2) %>%
  select(Region, Cause, Rate)
```

```
# A tibble: 20 x 3
   Region        Cause      Rate
   <chr>         <chr>     <dbl>
 1 HHS Region 01 Suicide     4.5
 2 HHS Region 01 Suicide     5.8
 3 HHS Region 02 Suicide     3.4
 4 HHS Region 02 Suicide     3.9
 5 HHS Region 03 Suicide     4.8
 6 HHS Region 03 Suicide     6.6
 7 HHS Region 04 Suicide     5.9
 8 HHS Region 04 Suicide     6
 9 HHS Region 05 Suicide     4.9
10 HHS Region 05 Suicide     5.1
11 HHS Region 06 Suicide     5.3
12 HHS Region 06 Suicide     6.4
13 HHS Region 07 Suicide     5.8
14 HHS Region 07 Suicide     5.8
15 HHS Region 08 Nephritis   8.3
16 HHS Region 08 Suicide     8.4
17 HHS Region 09 Suicide     5.2
18 HHS Region 09 Nephritis   6.1
19 HHS Region 10 Nephritis   5.9
20 HHS Region 10 Nephritis   6.7
```

In each of these, the `group_by` remains active, so the `slice` or `slice_min` will work *within* each health region, and give you the two smallest *from each one*.

Roughly speaking, one point for each of: the filter, the group-by, the slice-min (or sort and slice), and the final select. Expect to lose half a point if you have the appropriate element but somehow mess up coding it. The `filter` needs to be first, because you don't want to be including males and then trying to get rid of them later. (One or both of the two smallest

mortality rates in each region might be for males, and the ones you actually want might be the third or fourth smallest altogether if you have males and females both, and you don't know ahead of time whether this is the case or not.)

Extra: as I have it, the least likely cause of death is usually suicide, but sometimes nephritis. Suicide, of course, can have an outsized impact on friends and family even though it is not that common as a cause of death. Nephritis is inflammation of the kidneys; the kidneys are bean-shaped organs that filter the blood circulating the body to remove excess water and waste products from it. If the kidneys stop working properly, other bad things can happen.

4. Shrimp cocktail is a seafood dish consisting of shelled, cooked shrimp in a sauce, and is served in a glass. It used to be a popular starter at restaurants. Shrimp cocktail is required to contain a certain percentage (by weight) of shrimp. Samples of a certain brand of shrimp cocktail were sent to 18 different labs for analysis, with the results shown in Figure 8.

   (a) [2] What code would obtain a 90% confidence interval for the mean percentage of shrimp (by weight)?

My answer:

There is (as you see in the Figure) one column called `percent`; the dataframe (as you see from the code above the Figure) is called `shrimp`.

Hence, two ways (either of which is good). Don't forget to specify the confidence level, since it is not the default 95%:

```
with(shrimp, t.test(percent, conf.level = 0.90))
```

```
    One Sample t-test

data:  percent
t = 73.175, df = 17, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
90 percent confidence interval:
 31.03859 32.55030
sample estimates:
mean of x
 31.79444
```

or

```
t.test(shrimp$percent, conf.level = 0.90)
```

```
    One Sample t-test
```

```
data:  shrimp$percent
t = 73.175, df = 17, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
90 percent confidence interval:
 31.03859 32.55030
sample estimates:
mean of x
 31.79444
```

I'm guessing that most of you will go for the second one, which is fair enough as it is shorter (and, you might say, easier to understand).

Expect to lose a point for forgetting the `conf.level`, which I suspect will be the most common error. Minus a half point for smaller things that will stop the code working.

Extra: I can see (which you won't have known) that the confidence interval is from 31.0 to 32.6 (percent). The data are given in Figure 8 to one decimal, so you could justifiably go up to two decimals in giving the interval, had I asked you to do so.

(b) [3] Figure 9 shows the code and output for an analysis of these data. What specifically do you conclude? Explain briefly (by which I mean that your reader should end up convinced that you have drawn an appropriate conclusion).

My answer:

This is doing a hypothesis test (the clue being the presence of the `mu` and the `alternative` in the code at the top of the Figure).

I think the clearest way to show that you know what you are doing is bullet points:

- The null hypothesis is that the population mean is 34 (percent; the `mu = 34` in the code)
- The alternative hypothesis is that the population mean is less than 34 (the `alternative = "less"` in the code)
- The P-value is 0.000047
- This is (much) less than 0.05, so we reject the null hypothesis in favour of the alternative
- we therefore conclude that the population mean shrimp content (of this brand of shrimp cocktail) is in fact less than 34 percent.

The clearest answer here is one that goes through all the steps of the test, as I did: what are the hypotheses, on what basis are we rejecting the null hypothesis, what does this mean in terms of shrimp content of shrimp cocktail. That's what I meant by being convincing: by leading your reader through your process, you have demonstrated that you know what you are talking about.

An additional clue that the confidence interval is not what we care about here is that the interval given, which goes down to minus infinity, is a one-sided thing, and confidence intervals for us are two-sided.

If you try to do the test using this confidence interval, you could say that this interval only includes values less than 34, and therefore that the population mean is significantly less than 34, but the problem is that you don't know on the basis of the interval exactly how significant the result is. The P-value must be less than 0.05 (95% CI), but you don't know how much less, until you look at the actual P-value.

Points: three points for getting all the way to the end for a good reason (including a statement about percent of shrimp). Two for doing the right thing but missing a key step of logic (which includes trying to do the test with the confidence interval, because by not giving an accurate P-value, your reader doesn't know what to do if they want to carry out their test at 0.01, say, rather than 0.05). One for something relevant.

Extra: now that you have all the results, including the confidence interval you gave code for above, the smallness of the P-value is not so surprising: the null mean of 34 is apparently close to the confidence interval, but the confidence interval is short (because the labs gave pretty consistent measurements) and hence 34 is, relatively speaking, in fact quite a long way outside the confidence interval.

(c) [2] A graph of the shrimp percentages is shown in Figure 10. On the basis of this graph and the information given in the question, what are *two* reason why the analysis above is trustworthy? Explain briefly.

My answer:

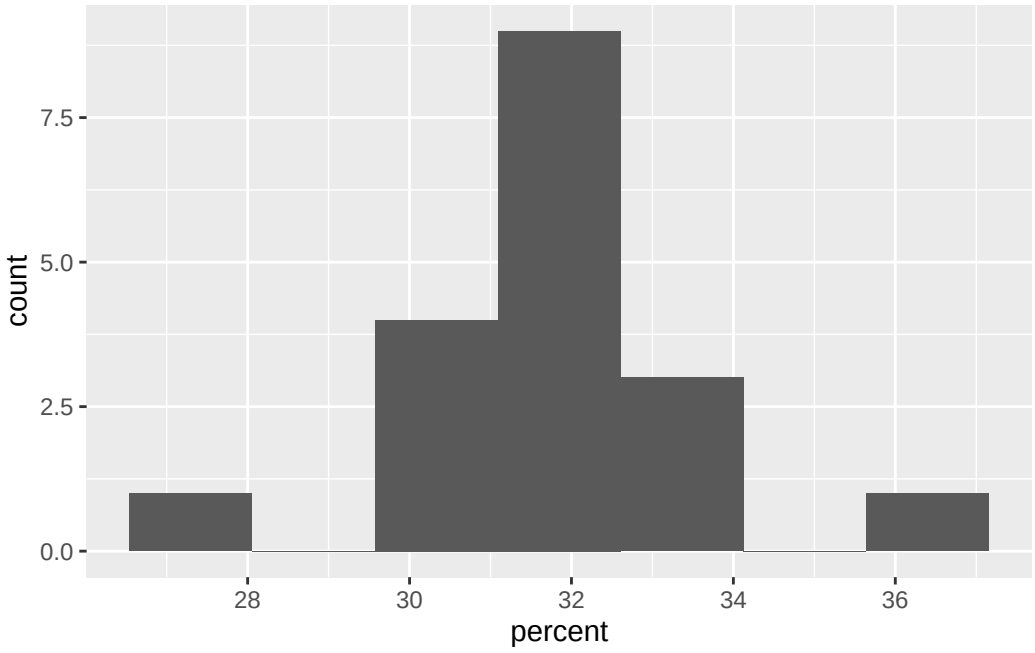
- The distribution looks approximately bell-shaped (close to a normal distribution).
- The sample size of 18 (there are 18 rows of data in Figure 8) is large enough that the sampling distribution of the sample mean will be even closer to normal than the data are. There are many ways to say this; you could also say that the data distribution is close enough to normal already that a "moderate" sample size (or whatever adjective you want to apply to $n = 18$) is large enough in this case.

Find a comment about the normality and a comment about the sample size that support each other, and you are good. One point for each of those.

You might also reasonably have said that the peak looks a bit too tall for the distribution to be bell-shaped. What happens in practice is that a tall peak like this one often goes with outliers (see the Extra below). An argument on those grounds for using the t-test nonetheless would have to rest on the Central Limit Theorem; you would need to say that the sample size is large enough to compensate for this kind of non-normality. This kind of argument, properly made, is fine.

Extra: I misled you a bit in actual fact. The above is how I wanted you to answer the question, but changing the number of bins on the histogram tells you a different story (I used 5 bins before):

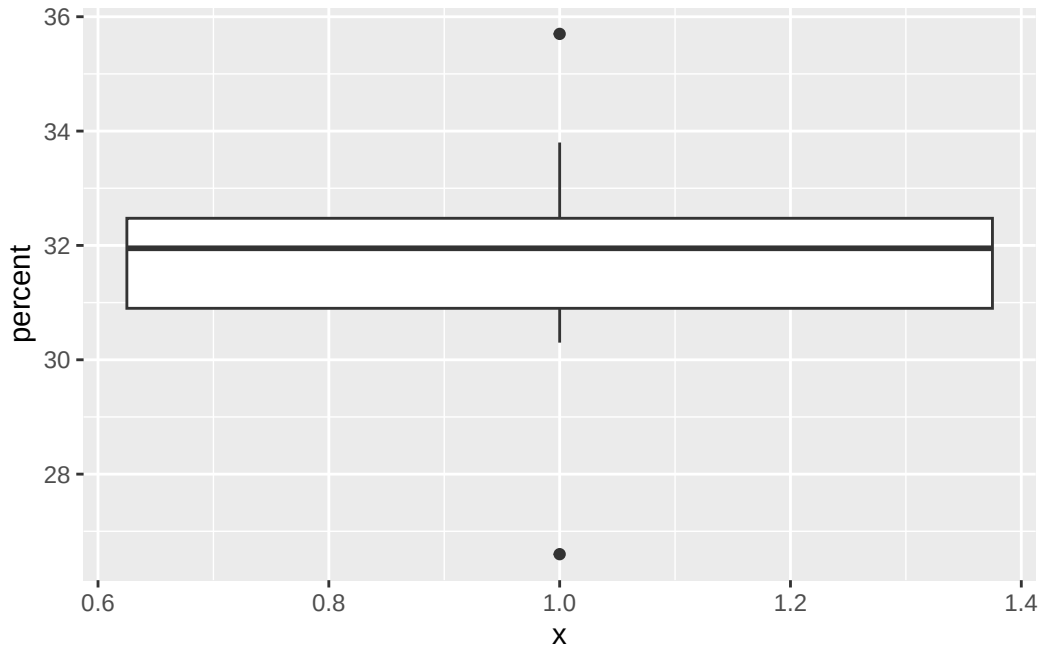
```
ggplot(shrimp, aes(x = percent)) + geom_histogram(bins = 7)
```



Now we see something that looks like outliers. (This is a danger of a histogram: it matters how many bins you choose.)

Now that we have seen that, you might suspect that in this case a one-sample boxplot might have been a better idea, and so it is:

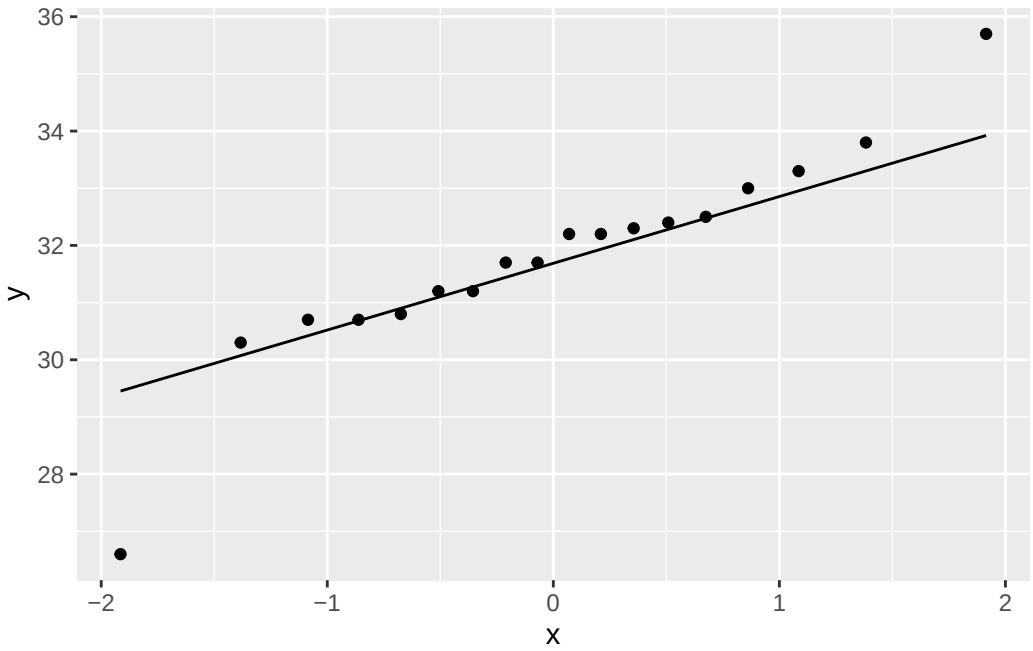
```
ggplot(shrimp, aes(x = 1, y = percent)) + geom_boxplot()
```



A boxplot is by its design good at showing outliers, and if I had shown you this instead of the histogram, your conclusion could have been very different.

A plot that is good for assessing normality specifically is the normal quantile plot. We have not reached this in the course yet, but I use the idea again below, so I introduce it now:

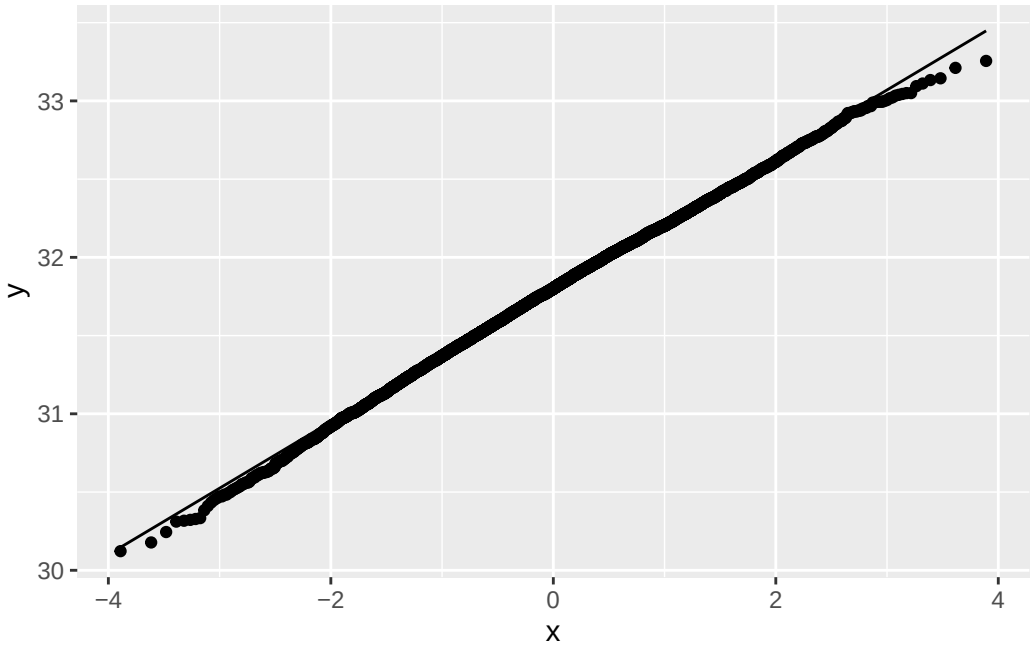
```
ggplot(shrimp, aes(sample = percent)) + stat_qq() + stat_qq_line()
```



All but the two extreme points are close to the line, but the highest value is too high and the lowest value is (much) too low, so that normality fails because of these two outliers.

The next question is "does this matter?" To assess this, find a bootstrap sampling distribution of the sample mean. When I'm going to draw a normal quantile plot at the end of one of these, I like to use 10,000 simulations, so that the tails of the distribution can be believed:

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



This is a tiny bit skewed to the left, to reflect that the lower outlier is more low than the high outlier is high. In fact, the high outlier is not even a problem (if it were, there would also be a noticeably long *upper* tail to the sampling distribution of the sample mean).

That is to say, when you look in enough detail, the distribution is in fact *not* quite normal enough to completely trust the t-interval and test, a different conclusion from the one I nudged you towards in the actual question. Outliers (or long tails at both ends) can be more of a problem than you might suspect at first glance; after this further consideration, the confidence interval for the mean is actually a bit longer than it ought to have been, because the outliers are inflating the sample standard deviation.

This dataset comes from the `MASS` package, and *they* got it from a textbook on "robust methods" (procedures that we see later like the sign test and its CI for the median) that are not affected (or are less affected) by outliers. In that book, there are about five different robust methods used, and their confidence intervals for the mean or median more or less agree in all cases, but the t interval is noticeably longer than all of those.

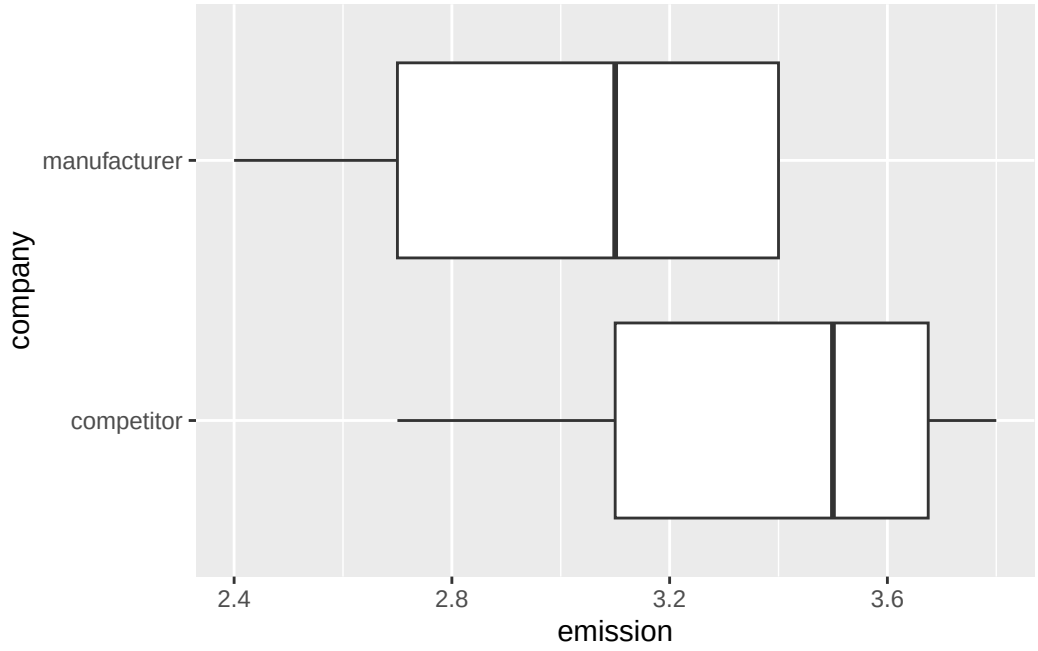
5. A manufacturer is concerned about the environmental impact of the smokestack emissions of its factory. In particular, the manufacturer measures the amount of carbon monoxide emitted from the smokestacks of its factory, and from a factory of a competitor, and wants to show that the manufacturer has less of an environmental impact than the competitor. A smaller carbon monoxide emission is better. The data are shown in Figure 11. There are nine observations from the manufacturer's smokestack and ten from the competitor's smokestack (each measured at different times). The dataframe is called `monoxide`.

(a) [3] A plot is shown in Figure 12. What code was used to make this plot, and why does it work?

My answer:

This is what I used. It's a boxplot, but with the axes flipped so that the boxes go across the page rather than up and down. You can break the line of code anywhere as long as you end with a `+`:

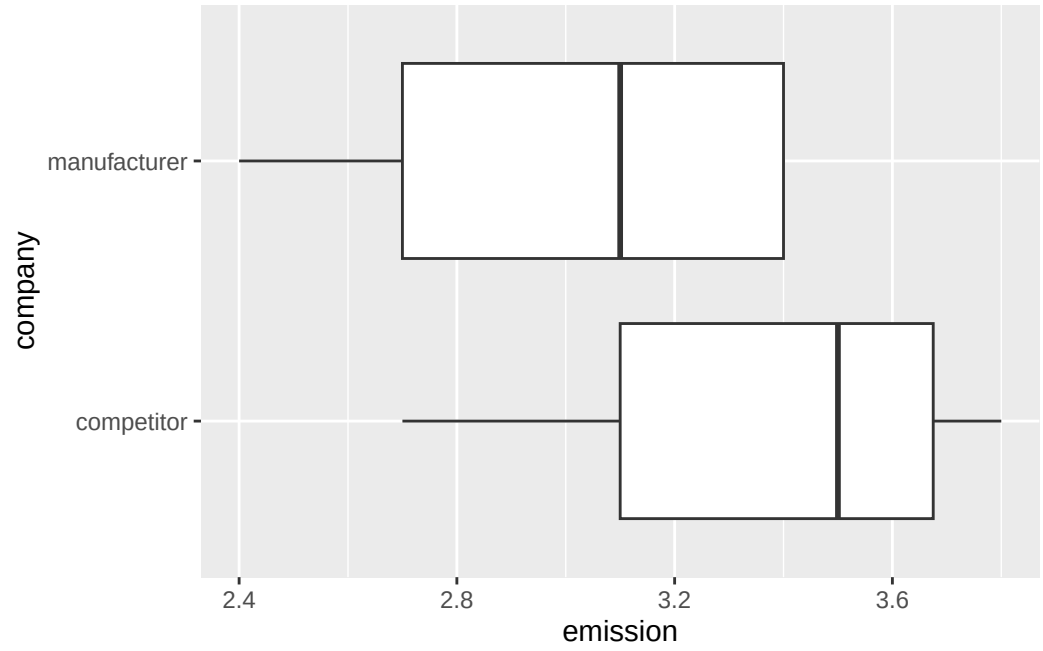
```
ggplot(monoxide, aes(x = company, y = emission)) +
  geom_boxplot() + coord_flip()
```



Remember that the `x` and `y` this way are for the *vertical* boxplot. Say something about the use of `coord_flip` in this context: for example, "we draw a regular boxplot and then interchange the roles of x and y".

I think somebody asked in lecture whether you could flip (interchange) the `x` and `y` directly:

```
ggplot(monoxide, aes(y = company, x = emission)) +
  geom_boxplot()
```



and in fact it *does* work. If you remember that, say something about having the x scale be the quantitative variable, and the y being the categorical one, which is the reverse of the usual way. This is actually an easier way to do it, and is perhaps too easy to guess if you don't know, but you'll need to assert something about why it works if you guess. In general, the `coord_flip` way is something you can rely on because it works for any plot; literally exchanging `x` and `y` happens to work for boxplots, but may not work at all for other plots.

Points: two for the boxplot, properly drawn (meaning that there should be something *in your code* that will get the boxes to go across the page), and one for saying something about *why* the boxplots will go left-to-right on your plot rather than up and down. (Thus a standard vertical boxplot with no other explanation is only one out of three.) Expect the grader to look carefully at your `x` and `y` to make sure that you have the categorical `company` as `x` *with* a `coord_flip`, or that you have it as `y` *without* a `coord_flip`.

(b) [3] What code would run a suitable t-test for these data? Justify your choice briefly.

My answer:

A two-sample t-test. You can choose to run either the Welch or the pooled one, but you *need a reason* to run the test you did. Also, you need to note that a smaller emission is better, and therefore that a one-sided test is needed: the manufacturer is trying to prove that its emissions are *better* (smaller) than its competitor.

The competitor is first (first alphabetically), so your choice of alternative is how the competitor

compares to the manufacturer *in that order*. (The flipped boxplot is confusing because the first category actually appears at the *bottom*.)

You might choose to run Welch's test on the grounds that the spread of the competitor's emissions is a little smaller:

```
t.test(emission ~ company, alternative = "greater", data = monoxide)
```

```
    Welch Two Sample t-test

data:  emission by company
t = 2.1187, df = 16.842, p-value = 0.02465
alternative hypothesis: true difference in means between group competitor and group manufactur
95 percent confidence interval:
 0.06802198        Inf
sample estimates:
  mean in group competitor mean in group manufacturer
                   3.370000                    2.988889
```

or you might say that the spreads are similar, and therefore we can run a pooled test:

```
t.test(emission ~ company, alternative = "greater", data = monoxide, var.equal = TRUE)
```

```
    Two Sample t-test

data:  emission by company
t = 2.1169, df = 17, p-value = 0.02466
alternative hypothesis: true difference in means between group competitor and group manufactur
95 percent confidence interval:
 0.06793172        Inf
sample estimates:
  mean in group competitor mean in group manufacturer
                   3.370000                    2.988889
```

I don't mind which one you choose, *as long as you state your reason for choosing the one you did*. As you see, the P-values don't differ until you get to the *fifth* decimal, so it actually doesn't matter at all in this case. But where I am trying to get you to is to make a good choice when it *does* matter. (Usually the big problem is using the pooled test when you should have used the Welch test.)

Two points for correct code for the two-sample t-test of your choice, and one for saying why you chose the one you did.

(c) [3] The output from an appropriate test is shown in Figure 13. (Note that this may or may not be the same test as you gave code for in the previous part.) What do you conclude from this output,

in the context of the data? (Note that some of the text in the Figure has run off the side of the page and is not visible. Use what you can see.) Explain briefly.

My answer:

The output is actually from a *one-sided* test, which you can tell from the one-sided confidence interval (that goes all the way up to infinity), and also from the very last bit of the alternative hypothesis line that is cut off at the right edge of the page: it must be something like "greater" rather than "not equal". The one-sided alternative must also be that emissions for the manufacturer are less (or that emissions for the competitor are greater), because of the way around the sample means came out, and that the P-value is small): we must be on the "correct side". Or piece this together with the statements in the question that a smaller emission is better and the manufacturer wants to show that it is doing better than the competitor. So:

- the null hypothesis is that the mean carbon monoxide emissions from the two factories are equal (over all times, not just the times when they happened to be observed)
- the alternative hypothesis is that the mean carbon monoxide emission is *less* for the manufacturer (or greater for the competitor)
- the P-value of 0.025 is smaller than 0.05, so we reject the null hypothesis in favour of the alternative
- therefore we conclude that the manufacturer's factory has *lower* carbon monoxide emissions than the competitor's factory (on average, over all times) and therefore that the manufacturer has less of an environmental impact (in terms of carbon monoxide) than the competitor does.

The best answer includes all of those, to convince the reader not only that the conclusion is what you say it is, but also that this conclusion is correct. Three points for that; two for something sound that is missing a piece of the story, and one for some relevant comment otherwise.

Your answer needs to make some things clear:

- whether your test is one-sided or two-sided (that is, what the alternative hypothesis actually is). This test should be one-sided, for the reasons given above.
- you need to draw the appropriate conclusion given what your alternative is. If your alternative is two-sided, your conclusion is that the mean carbon monoxide emissions are *different*, not that the manufacturer's mean is less than the competitor's. If you are going to do it differently from me, *be consistent*. Strictly, if you do a two-sided test here and make a one-sided conclusion, you have made *two* errors.
- It is cheating (statistically) to do a two-sided test and *then* look at the sample means; you need to make your decision about what we are trying to prove *before* you look at the data.
- you need to say what the P-value *is*. This is so that your reader can draw their own conclusion, if they happen to disagree with your choice of $\alpha = 0.05$. In this case, if they think α should be 0.01, they will *not* reject the null in favour of the one-sided alternative, but if you give the P-value, they can come to that decision for themselves.
- you need a conclusion about mean carbon monoxide emissions from factory smokestacks, because that is what your reader actually cares about. (Your reader might be someone like a manager who cares whether the manufacturer is producing less than the industry

standard level of emissions.)
- using a confidence interval to do a test is a mistake, because 0 being outside the (one-sided) CI tells you only that the P-value is less than 0.05, not how small it actually is. (Using words with "with 95% confidence" in your answer makes me wonder whether you really know what you are doing here.)

(d) [2] Why might you have some doubts about running a t-test here?

My answer:

Two things to consider:

- both distributions look left-skewed (saying that there is one distribution that is not normal enough is enough, because you only need one to fail for the whole t-test to fail)
- the sample sizes of 9 and 10 (stated in the question) are not large, so the Central Limit Theorem will not help much. (Consider the sample sizes separately, rather than the total sample size of 19.) Or say that the sample sizes are not large enough to overcome the skewness in the two distributions of carbon monoxide measurements.

One point each for a relevant comment about non-normality and for a relevant comment about sample sizes. The best comment about sample sizes says something about why the small sample sizes *matter*. Talking about the sample sizes "not being large enough to overcome the skewness" is a two-point answer because you have said enough about what the problem is and why a larger sample size would be better.

There is *nothing* magic about a sample size of 30. It depends on how non-normal your data are; if your data are only slightly skewed, a sample size of 10 (or maybe even less) might be big enough, but if you have severe skewness or outliers, you might need a sample size in the hundreds. The wording "a large enough sample to overcome the skewness" is powerful this way, because it reminds you that you only need the sample size to be large enough to overcome whatever non-normality you have (and that sample size might be quite small).

Extra: with the actual data, we can bootstrap to find whether the t-tests are ok. I did this one by working out the bootstrap distribution of the difference between the two bootstrap means. You could also look at the bootstrap distributions of the two sample means separately:

```
monoxide %>% filter(company == "manufacturer") -> manu
manu
```

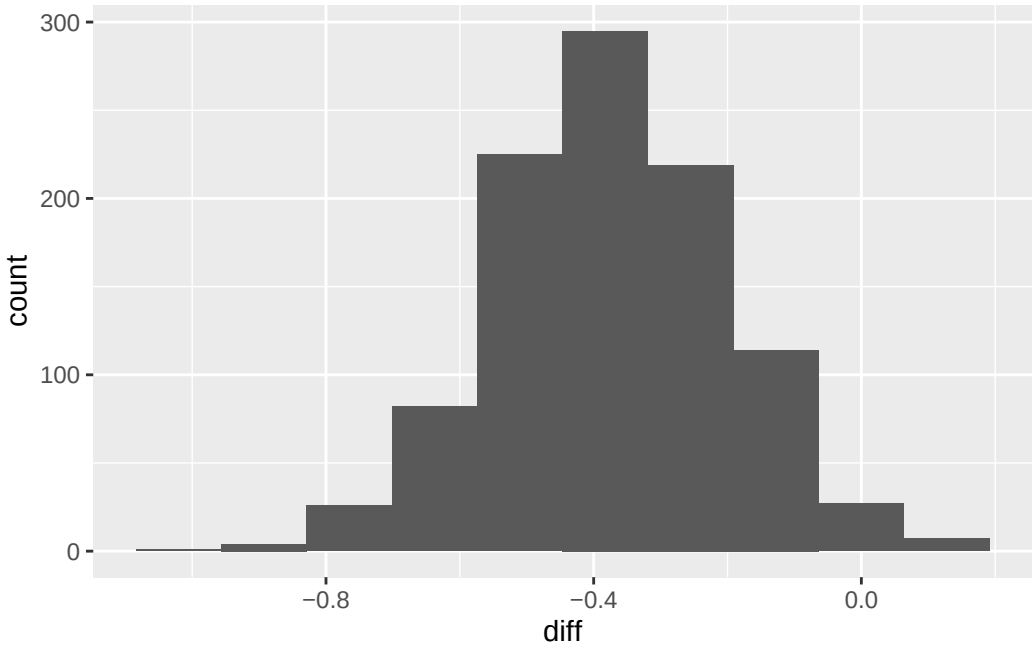
```
# A tibble: 9 x 2
  company      emission
  <chr>           <dbl>
1 manufacturer      2.7
2 manufacturer      3.1
3 manufacturer      3.1
4 manufacturer      2.9
5 manufacturer      2.5
```

```
6 manufacturer      3.4
7 manufacturer      3.4
8 manufacturer      3.4
9 manufacturer      2.4
```

```
monoxide %>% filter(company == "competitor") -> comp
comp
```

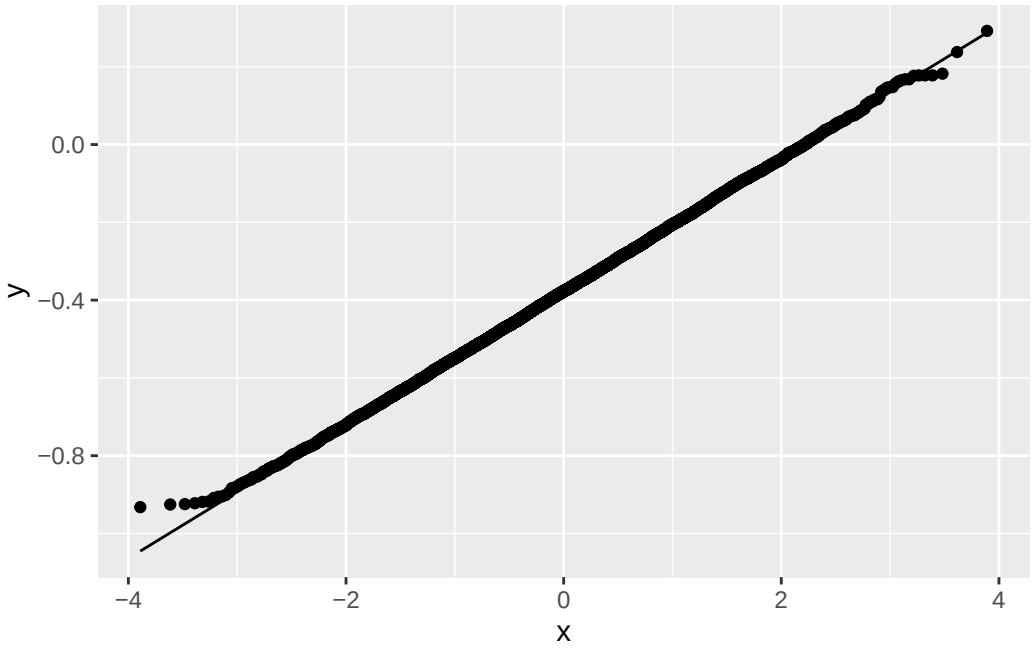
```
# A tibble: 10 x 2
   company    emission
   <chr>         <dbl>
 1 competitor      3.7
 2 competitor      3
 3 competitor      3.5
 4 competitor      3.8
 5 competitor      2.8
 6 competitor      3.5
 7 competitor      3.4
 8 competitor      3.6
 9 competitor      2.7
10 competitor      3.7
```

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(manu_sample = list(sample(manu$emission, replace = TRUE))) %>%
  mutate(comp_sample = list(sample(comp$emission, replace = TRUE))) %>%
  mutate(manu_mean = mean(manu_sample), comp_mean = mean(comp_sample)) %>%
  mutate(diff = manu_mean - comp_mean) %>%
  ggplot(aes(x = diff)) + geom_histogram(bins = 10)
```



That is actually really close to normal. Do again but get normal quantile plot instead:

```
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(manu_sample = list(sample(manu$emission, replace = TRUE))) %>%
  mutate(comp_sample = list(sample(comp$emission, replace = TRUE))) %>%
  mutate(manu_mean = mean(manu_sample), comp_mean = mean(comp_sample)) %>%
  mutate(diff = manu_mean - comp_mean) %>%
  ggplot(aes(sample = diff)) + stat_qq() + stat_qq_line()
```



it is in fact extraordinarily good: no problem with the t-test at all.

You might be curious about why that is; there was nothing in the data distributions or the sample sizes to suggest that the two-sample test would in fact work. There are two reasons I can think of:

- the central limit theorem is more powerful than you might expect, and even a sample size around 10 is enough to overcome some skewness.
- the two-sample t-test is based on the *difference* of the two sample means, and this can be enough to help the normality along more. In this case, both distributions are skewed *the same way*, and by taking the difference of the two sample means, the skewness will to some degree cancel out.

Those together, I think, are why the two-sample t-test here is better than you might think. (So again, in the question I actually misled you a bit.)

Use this page if you need more space. Be sure to label any answers here with the question and part they belong to.

Numbered Figures begin here:

```
library(tidyverse)
library(readxl)
library(smmr)
```

Figure 1: Packages

Group	Leniency
neutral	6
smile	3.5
smile	4.5
smile	6
smile	4
neutral	2.5
smile	7.5
smile	2.5
smile	3.5
neutral	4
neutral	2.5
neutral	4.5
smile	3.5
smile	9
neutral	3
smile	3
smile	5
neutral	4.5
smile	5.5
smile	5

Figure 2: Smiles leniency data

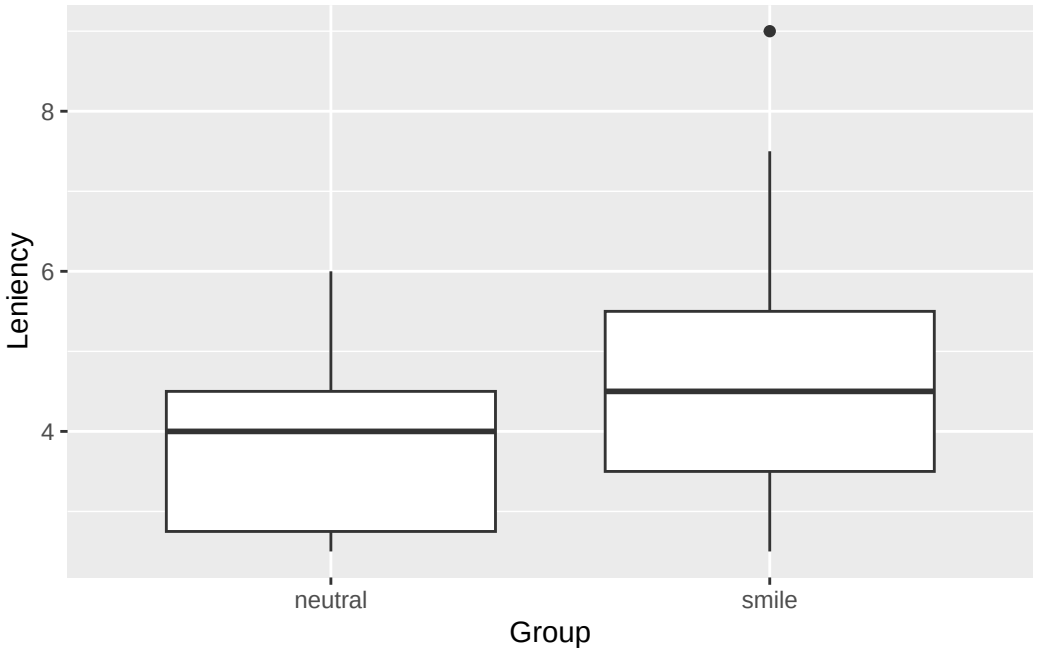


Figure 3: Smiles leniency plot

```
# A tibble: 30 x 3
   age_group fact_correct opinion_correct
   <chr>            <dbl>           <dbl>
 1 18-49                3               5
 2 18-49                5               5
 3 18-49                5               5
 4 50+                  4               1
 5 18-49                2               4
 6 50+                  5               5
 7 18-49                5               5
 8 50+                  4               2
 9 18-49                2               5
10 50+                  4               3
11 50+                  2               5
12 18-49                3               5
13 50+                  1               4
14 18-49                3               3
15 50+                  3               3
16 50+                  3               2
17 18-49                5               5
18 50+                  3               3
19 50+                  2               5
20 18-49                5               5
21 50+                  5               1
22 18-49                2               5
23 50+                  4               3
24 18-49                3               1
25 50+                  5               5
26 18-49                1               5
27 50+                  3               5
28 50+                  4               3
29 50+                  1               4
30 18-49                5               5
```

Figure 4: Fact and opinion survey data (30 randomly chosen rows)

```
fact_opinion %>% count(age_group) -> counted
ggplot(counted, aes(x = age_group)) + geom_bar()
```

Figure 5: Some code

```
ggplot(fact_opinion, aes(x = fact_correct, fill = age_group)) +
  geom_bar(position = "dodge")
```

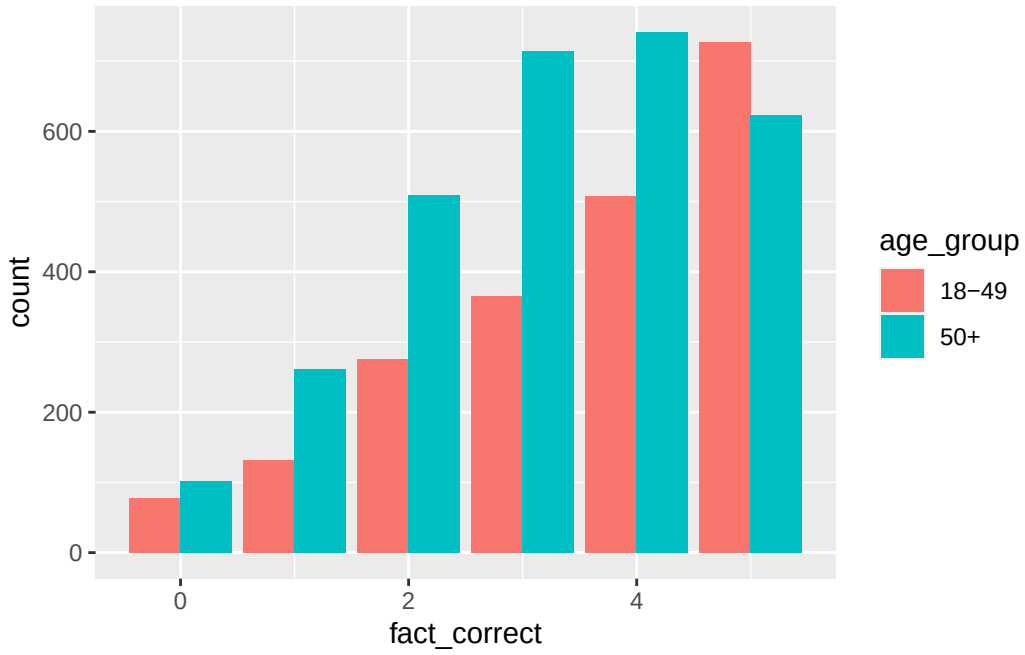


Figure 6: Fact and opinion survey plot

```
# A tibble: 30 x 6
   Region        Status Sex    Cause                     Rate    SE
   <chr>         <chr>  <chr>  <chr>                    <dbl> <dbl>
 1 HHS Region 08 Urban  Male   Unintentional injuries    55.3   0.7
 2 HHS Region 10 Urban  Male   Cancer                   191.    1.1
 3 HHS Region 10 Urban  Female Cerebrovascular diseases  35.2   0.4
 4 HHS Region 06 Urban  Male   Alzheimers                20.7   0.3
 5 HHS Region 10 Urban  Male   Unintentional injuries    49.8   0.6
 6 HHS Region 03 Rural  Female Cancer                   157.    1.4
 7 HHS Region 10 Rural  Male   Cerebrovascular diseases  37.1   1
 8 HHS Region 02 Rural  Male   Flu and pneumonia         19.6   0.9
 9 HHS Region 09 Rural  Male   Heart disease            206.    2.7
10 HHS Region 02 Urban  Female Flu and pneumonia         14.5   0.2
11 HHS Region 06 Urban  Male   Cancer                   202.    0.7
12 HHS Region 07 Rural  Male   Lower respiratory         65.9   0.9
13 HHS Region 01 Rural  Female Diabetes                  15     0.6
14 HHS Region 07 Urban  Female Diabetes                  16     0.3
15 HHS Region 09 Rural  Male   Unintentional injuries    79.1   1.8
16 HHS Region 04 Rural  Male   Unintentional injuries    79.1   0.7
17 HHS Region 01 Urban  Female Cancer                   140.    0.8
18 HHS Region 08 Urban  Female Nephritis                  8.3   0.3
19 HHS Region 07 Rural  Female Cancer                   150.    1.3
20 HHS Region 07 Rural  Male   Unintentional injuries    68.1   1
21 HHS Region 01 Rural  Male   Lower respiratory         51.7   1.3
22 HHS Region 09 Urban  Female Suicide                    5.2   0.1
23 HHS Region 05 Urban  Female Nephritis                 12.9   0.1
24 HHS Region 06 Rural  Male   Unintentional injuries    77.2   0.9
25 HHS Region 08 Rural  Male   Unintentional injuries    71     1.3
26 HHS Region 08 Urban  Female Alzheimers                30.9   0.5
27 HHS Region 10 Rural  Male   Alzheimers                22.9   0.8
28 HHS Region 06 Urban  Male   Heart disease            220.    0.8
29 HHS Region 05 Rural  Female Unintentional injuries    32.3   0.4
30 HHS Region 03 Rural  Male   Lower respiratory         62.1   1
```

Figure 7: US regional mortality rates data (randomly chosen rows)

```
my_url <- "http://ritsokiguess.site/datafiles/shrimp.csv"
shrimp <- read_csv(my_url)
```

```
Rows: 18 Columns: 1
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (1): percent

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.
```

```
shrimp
```

```
# A tibble: 18 x 1
   percent
     <dbl>
 1    32.2
 2    33
 3    30.8
 4    33.8
 5    32.2
 6    33.3
 7    31.7
 8    35.7
 9    32.4
10    31.2
11    26.6
12    30.7
13    32.5
14    30.7
15    31.2
16    30.3
17    32.3
18    31.7
```

Figure 8: Shrimp cocktail data

```r
with(shrimp, t.test(percent, mu = 34, alternative = "less"))
```

```
    One Sample t-test

data:  percent
t = -5.0761, df = 17, p-value = 4.674e-05
alternative hypothesis: true mean is less than 34
95 percent confidence interval:
    -Inf 32.5503
sample estimates:
mean of x
 31.79444
```

Figure 9: Code and output for an analysis on the shrimp data

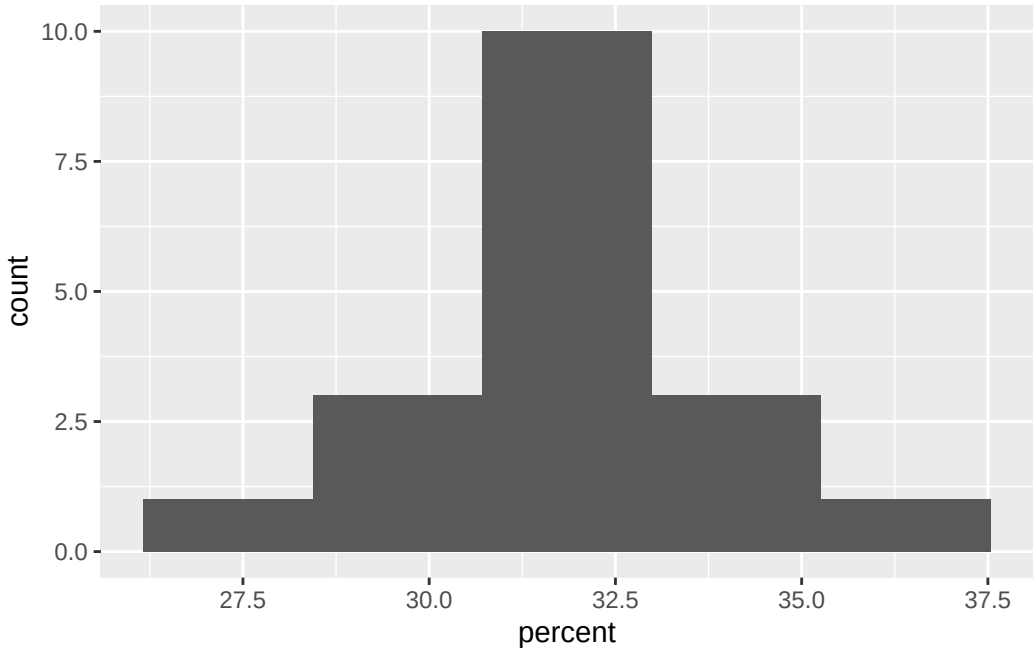


Figure 10: Histogram of shrimp data

```
# A tibble: 19 x 2
   company      emission
   <chr>           <dbl>
 1 manufacturer      2.7
 2 manufacturer      3.1
 3 manufacturer      3.1
 4 manufacturer      2.9
 5 manufacturer      2.5
 6 manufacturer      3.4
 7 manufacturer      3.4
 8 manufacturer      3.4
 9 manufacturer      2.4
10 competitor        3.7
11 competitor        3
12 competitor        3.5
13 competitor        3.8
14 competitor        2.8
15 competitor        3.5
16 competitor        3.4
17 competitor        3.6
18 competitor        2.7
19 competitor        3.7
```

Figure 11: Carbon monoxide data

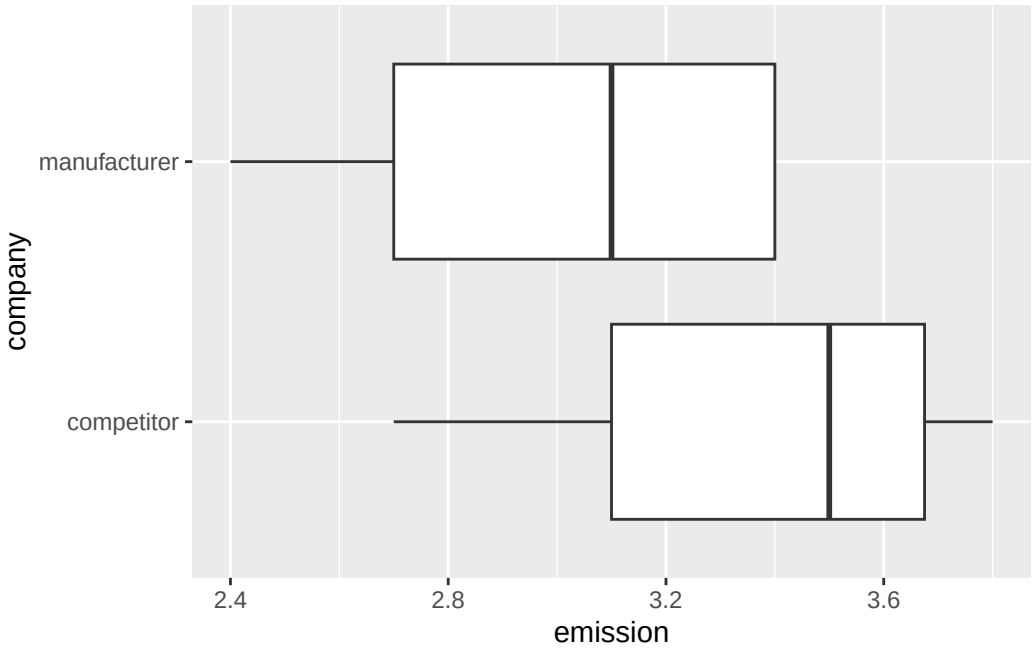


Figure 12: Plot for carbon monoxide data. Note that one of the whiskers for "manufacturer" is very short.

```
    Welch Two Sample t-test

data:  emission by company
t = 2.1187, df = 16.842, p-value = 0.02465
alternative hypothesis: true difference in means between group competitor and group manufacturer is grea
95 percent confidence interval:
 0.06802198        Inf
sample estimates:
  mean in group competitor mean in group manufacturer
                  3.370000                   2.988889
```

Figure 13: Test output for carbon monoxide data