University of Toronto Scarborough
Department of Computer and Mathematical Sciences
STAC33 (K. Butler), Midterm Exam
March 3, 2022

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 8 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. A study was made of three species of crab. In particular, interest was in the `Force` exerted by the crab's claw as it closed, and how that depended on the `Height` of the next joint of the claw. Also noted in the dataset was the `Species` of crab (text, possibly including a space). Some of the data file is shown in Figure 2. `Force` is measured in Newtons, and `Height` is measured in millimetres.

   (a) [4] What R code would read the data from the data file, called `crab-claws.txt`, into a dataframe called `crabs` and display the dataframe? The file is in the same folder that you are running R Studio in. (Here and elsewhere, if I ask for the code, I want the code and *not* the output. If I want the output, I will ask for it specifically.)

My answer:

Take a look at the data file in the Figure. The species names are two words with a space between, so the actual data values are separated by something else. In this case, it's a plus sign. Hence:

```
crabs <- read_delim("crab-claws.txt", delim = "+")
```

```
## Rows: 38 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: "+"
## chr (1): Species
## dbl (2): Force, Height
##
## 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.
```

```
crabs
```

```
## # A tibble: 38 x 3
##    Force Height Species
##    <dbl>  <dbl> <chr>
##  1   3.2    5   Hemigrapsus nudus
##  2   6.4    6   Hemigrapsus nudus
##  3   2      6.4 Hemigrapsus nudus
##  4   2      6.5 Hemigrapsus nudus
##  5   4.9    6.6 Hemigrapsus nudus
##  6   3      7   Hemigrapsus nudus
##  7   2.9    7.9 Hemigrapsus nudus
##  8   9.5    7.9 Hemigrapsus nudus
##  9   4      8   Hemigrapsus nudus
## 10   3.4    8.2 Hemigrapsus nudus
## # ... with 28 more rows
```

One point for displaying the dataframe after you have read it in, and three points for the code to read it in. Here, and elsewhere when you supply code, the grading is the grader's best estimate of what fraction of the way you came to getting it right. On this one, expect to lose one per error down to a minimum of one point (for the first line) if you have anything substantial correct.

This way *does not* work here:

```
crabs <- read_delim("crab-claws.txt")
```

```
## Error: Could not guess the delimiter.
##
## Use `vroom(delim =)` to specify one explicitly.
```

What happens is that `read_delim` tries to guess what the data values were separated by, and in this case failed because `+` is not commonly used to separate data values. Out of three: one if you do it this way without further explanation, and two if you explain that `read_delim` will try to guess the delimiter (which is what it does, but is not successful).

About the output thing: I add the output to my solutions (here and elsewhere) to convince you that the code I give actually does work (and to check that my code actually *does* work, because I do make mistakes).

The base R `read.table` and `read.delim` are *wrong* in this course, because they are not what I taught you in lecture. No points for those, even if your code would otherwise work. If you try to use these, your maximum for this part is the one point for displaying the dataframe. For these, your second input would need to be `header` and the delimiter is called `sep`, so either (if you include them) you betray that your code is not as learned in this course, or (if you omit them) you are making an additional error.

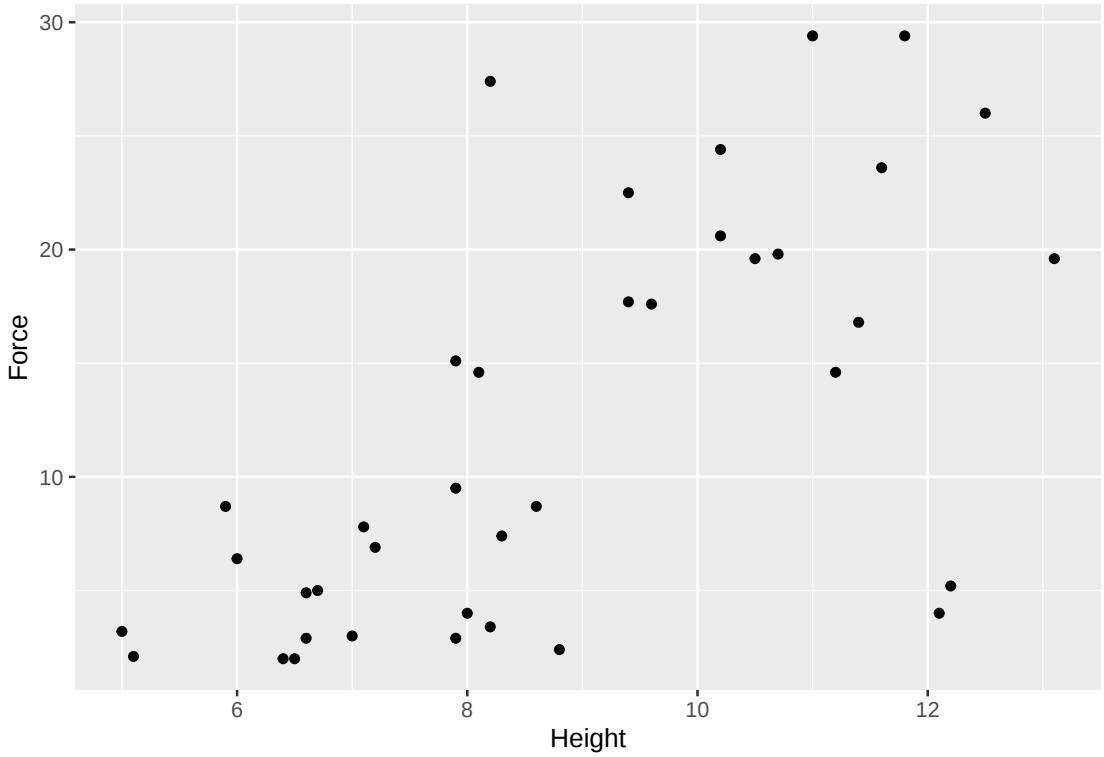
Extra: `https://thomaselove.github.io/2018-431-book/studying-crab-claws-crabs.html` tells you about this data set, and precisely what `Height` is. I have probably oversimplified for you. The link has a nice picture.

(b) [3] What code would make a suitable graph of the two variables `Force` and `Height`? (It is a good idea to name the graph you are trying to draw in case you go wrong.)

My answer:

This is two quantitative variables, so a scatterplot is the thing. The `Force` is the outcome ("how the force depended on the height"), so that goes on the y-axis:

```
ggplot(crabs, aes(x = Height, y = Force)) + geom_point()
```



This will plot the `Height` and `Force` of all the crabs together, regardless of species. We get to including the species in the next part.

Add a smooth trend (better than a regression line) if you wish. Expect to lose a point if you add a regression line, because you don't know until you look at the graph (which of course you cannot do in an exam) whether the relationship is linear: if it is not, a regression line doesn't tell you very much.

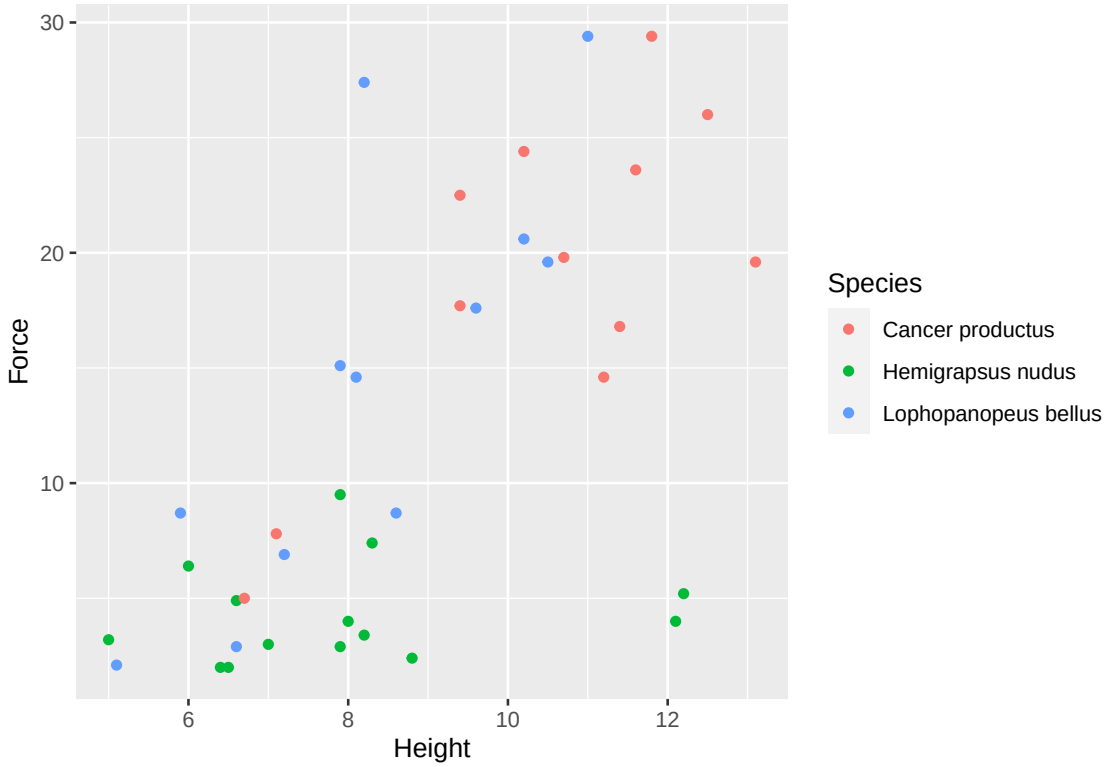
(c) [2] Describe in words how you would add `Species` to your graph of the previous part, and describe how you would *change* your code of the previous part to make this happen. (If you prefer, you can write out the complete code to draw this graph.)

My answer:

`Species` is categorical, so add this to your scatterplot by colouring the points according to species. (One point.)

My entire code is this:

```
ggplot(crabs, aes(x = Height, y = Force, colour = Species)) + geom_point()
```

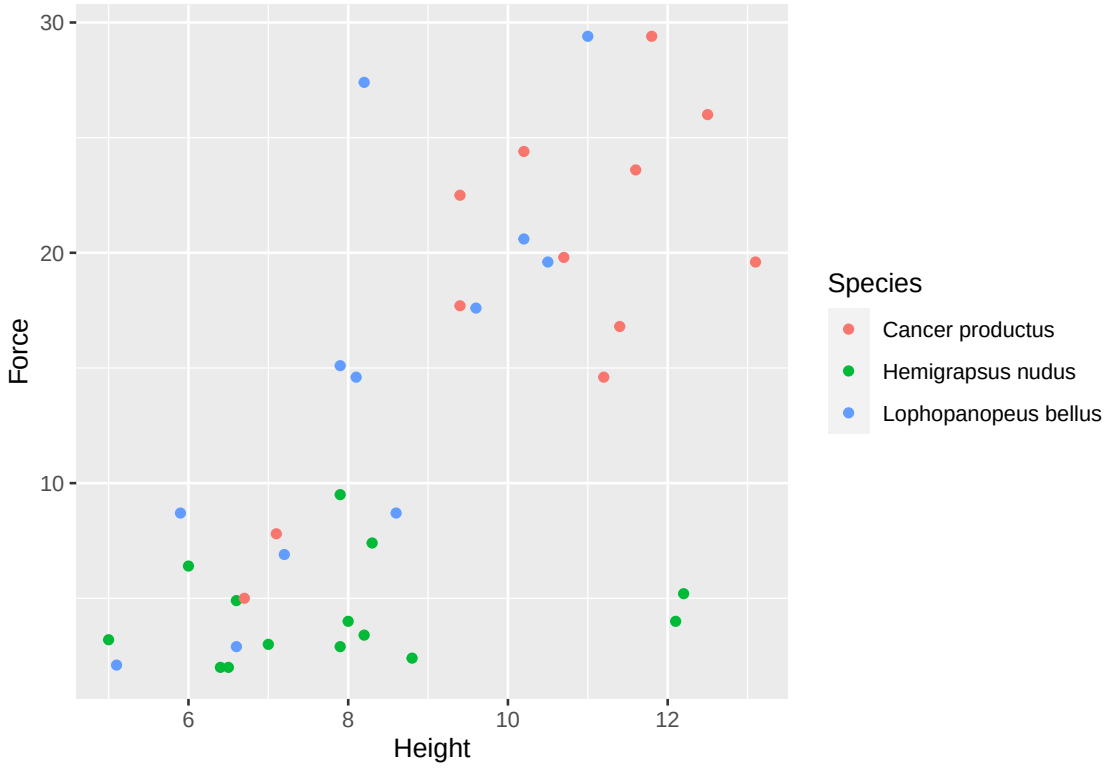


One point for this code, or for saying "add `colour = Species` to the `aes`". Only one point because it's a small addition to what you had before.

Grading note: the second point here is for the `colour = Species` part. As long as this appears, either with instructions to add it to the `aes`, or the entire code from the previous part with this added in the right place, you are good. See also Extra 2 below.

The above is the way we did it in lecture. The following also works if you do it right:

```
ggplot(crabs, aes(x = Height, y = Force)) + geom_point(aes(colour = Species))
```

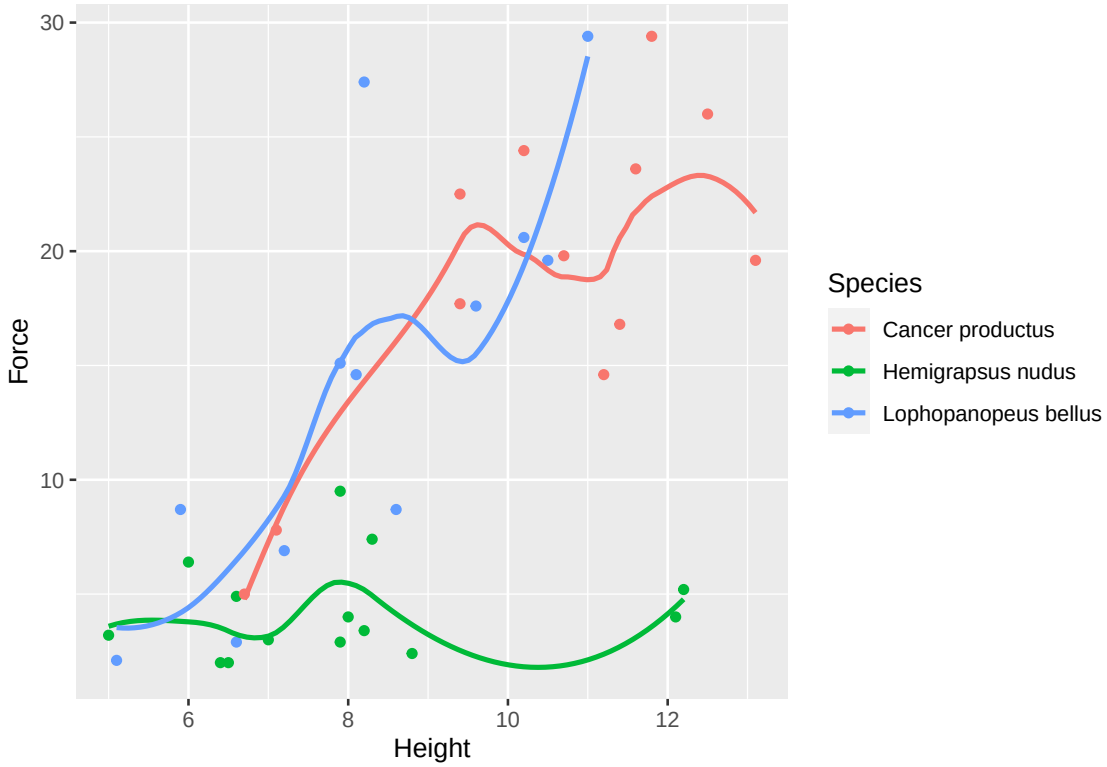


If you do it this way, you need another `aes` so that `ggplot` goes looking for `Species` in the dataframe you gave at the beginning, and not as a separate object in your workspace.

Extra 1: on the first graph we made, it looked sort of like a linear trend with two outliers at the bottom right. This graph reveals that they are both Hemigrapsus nudus, and if you add separate trends for species:

```
ggplot(crabs, aes(x = Height, y = Force, colour = Species)) +
  geom_point() + geom_smooth(se = FALSE)
```

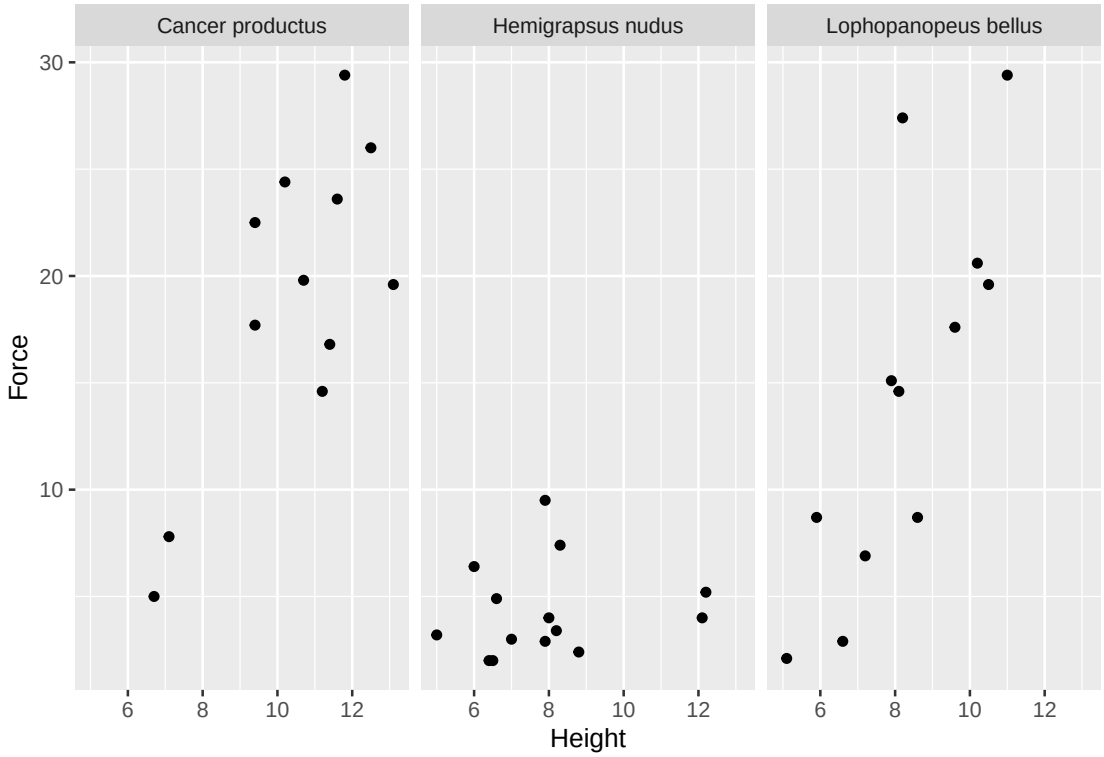
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



you see that as height increases, force increases as well for Cancer productus (red) and Lophopanopeus bellus (blue), but for Hemigrapsus nudus (green), there is no trend at all.

Extra 2: the graph in Figure 3 is also a reasonable way to add `Species` to your scatterplot. In words, "add facets, one for each `Species`". The code for that graph is here:

```
ggplot(crabs, aes(x = Height, y = Force)) +
  geom_point() +
  facet_wrap(~Species)
```



so, "use facets to distinguish species" is a good answer to the "in words" part (one point) and "add `facet_wrap(~Species)`" is a good answer for the code part (the other point). With the `Species` in facets, there is no need to distinguish them by colour, though you can do so if you want.

Now that you see the graphs, you might even say that it is clearer to see the upward trend on the left and right facets and no trend in the middle one, compared to a scatterplot where all the points are on the same axes.

If you were stuck earlier on adding Species to your graph, it would also be possible to reverse-engineer Figure 3, since this is also a valid graph. Think `Force` by `Height`, facetted by `Species`, and then write the words and additional code (beyond the scatterplot) to make that happen. Expect to get full credit if you get that right.

(d) [3] What code would show the number of crabs and the mean and standard deviation of `Force`, for each species separately?

My answer:

`group_by` and `summarize`. This one uses the `n()` to get the number of crabs in each group:

```
crabs %>%
  group_by(Species) %>%
  summarize(n = n(), mean_force = mean(Force), sd_force = sd(Force))
```

```
## # A tibble: 3 x 4
##   Species                 n mean_force sd_force
##   <chr>               <int>      <dbl>    <dbl>
## 1 Cancer productus       12      18.9      7.19
## 2 Hemigrapsus nudus      14       4.31     2.19
## 3 Lophopanopeus bellus   12      14.5      8.92
```

Give the summaries whatever names you like, but it is better if the names have something to do with what they are.

(e) [2] A graph is shown in Figure 3. This may or may not resemble the graph you gave code for earlier. From this graph, which species has the smallest standard deviation of `Force`? Explain briefly.

My answer:

The `Force` is on the y-axis of the facets (and the `Force` scale is the same for each one). In the facets for Cancer productus and Lophopanopeus bellus (the left and right ones) `Force` ranges from about zero to about 30, all the way up the facet, so the standard deviations will be large. For Hemigraspus nudus (middle facet), all the `Force` values are less than 10: that is to say, the values are close together and the standard deviation will be smaller. Hence Hemigraspus nudus will have the smallest standard deviation of `Force`.

Your answer should name a species (which you should make a reasonable attempt to spell correctly), and should also show your knowledge that the standard deviation is a measure of spread and how you know that the `Force` values for Hemigraspus nudus have a small spread or are close together.

Grading note: I would say minus a half point for not giving the species by its name, and one out of two for not sufficiently making the connection between small SD = small spread and what that looks like on the graph.

2. In question 1, we looked at some data on crab claws. The data, in dataframe `crabs`, are shown in Figure 4. There are three variables, `Species` (categorical), `Height` (quantitative), and `Force` (also quantitative). In this question, you are asked to give R (Tidyverse) code to accomplish the task described.

   (a) [2] Display all the variables *except* `Species`, without naming any of the other variables.

My answer:

Where I am trying to get you to go is this:

```
crabs %>% select(-Species)
```

```
## # A tibble: 38 x 2
##    Force Height
##    <dbl>  <dbl>
##  1   3.2    5
##  2   6.4    6
##  3   2      6.4
##  4   2      6.5
##  5   4.9    6.6
##  6   3      7
##  7   2.9    7.9
##  8   9.5    7.9
##  9   4      8
## 10   3.4    8.2
## # ... with 28 more rows
```

but if you have another way, like this one ("does not start with S"):

```
crabs %>% select(!starts_with("S"))
```

```
## # A tibble: 38 x 2
##    Force Height
##    <dbl>  <dbl>
##  1   3.2    5
##  2   6.4    6
##  3   2      6.4
##  4   2      6.5
##  5   4.9    6.6
##  6   3      7
##  7   2.9    7.9
##  8   9.5    7.9
##  9   4      8
## 10   3.4    8.2
## # ... with 28 more rows
```

that is also good. You can name `Species`, but you cannot name either of the other two.

You can also `select` columns by number. For this part I would (rather unwillingly) accept this:

```
crabs %>% select(1, 2)
```

```
## # A tibble: 38 x 2
##    Force Height
##    <dbl>  <dbl>
##  1   3.2    5
##  2   6.4    6
##  3   2      6.4
##  4   2      6.5
##  5   4.9    6.6
##  6   3      7
##  7   2.9    7.9
##  8   9.5    7.9
##  9   4      8
## 10   3.4    8.2
## # ... with 28 more rows
```

but not any base R version of selecting columns (or rows, later). No credit, for example, for this:

```
crabs[, 1:2]
```

```
## # A tibble: 38 x 2
##    Force Height
##    <dbl>  <dbl>
##  1   3.2    5
##  2   6.4    6
##  3   2      6.4
##  4   2      6.5
##  5   4.9    6.6
##  6   3      7
##  7   2.9    7.9
##  8   9.5    7.9
##  9   4      8
## 10   3.4    8.2
## # ... with 28 more rows
```

because you did not learn that in this course.

(b) [2] Display all the categorical (text) columns, without using or even having to know their names.

My answer:

Select the columns that are (have the property of being) text (there is in fact only one):

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

```
## # A tibble: 38 x 1
```

```
##    Species
##    <chr>
##  1 Hemigrapsus nudus
##  2 Hemigrapsus nudus
##  3 Hemigrapsus nudus
##  4 Hemigrapsus nudus
##  5 Hemigrapsus nudus
##  6 Hemigrapsus nudus
##  7 Hemigrapsus nudus
##  8 Hemigrapsus nudus
##  9 Hemigrapsus nudus
## 10 Hemigrapsus nudus
## # ... with 28 more rows
```

You need to display the ability to select columns by property, not by name or number, because either of those do not answer the question.

(c) [2] Display all the crabs for which `Force` is greater than 20.

My answer:

The crabs are in rows, so this one is `filter`:

```
crabs %>% filter(Force > 20)
```

```
## # A tibble: 8 x 3
##   Force Height Species
##   <dbl>  <dbl> <chr>
## 1  20.6   10.2 Lophopanopeus bellus
## 2  27.4    8.2 Lophopanopeus bellus
## 3  29.4   11   Lophopanopeus bellus
## 4  22.5    9.4 Cancer productus
## 5  23.6   11.6 Cancer productus
## 6  24.4   10.2 Cancer productus
## 7  26     12.5 Cancer productus
## 8  29.4   11.8 Cancer productus
```

(d) [3] Display only the `Species` of the crabs that have `Force` less than 5.

My answer:

Filter first (to get the crabs you want) and then select (to display the variables you want):

```
crabs %>% filter(Force < 5) %>%
  select(Species)
```

```
## # A tibble: 12 x 1
```

```
##    Species
##    <chr>
##  1 Hemigrapsus nudus
##  2 Hemigrapsus nudus
##  3 Hemigrapsus nudus
##  4 Hemigrapsus nudus
##  5 Hemigrapsus nudus
##  6 Hemigrapsus nudus
##  7 Hemigrapsus nudus
##  8 Hemigrapsus nudus
##  9 Hemigrapsus nudus
## 10 Hemigrapsus nudus
## 11 Lophopanopeus bellus
## 12 Lophopanopeus bellus
```

If you do the `select` first, you won't have a `Force` column to `select` by, so you need to do it this way around.

(e) [3] Find how many crabs of each species have `Force` less than 5.

My answer:

Probably several ways, but the one that comes to mind is this:

```
crabs %>% filter(Force < 5) %>%
  count(Species)
```

```
## # A tibble: 2 x 2
##   Species                 n
##   <chr>               <int>
## 1 Hemigrapsus nudus      10
## 2 Lophopanopeus bellus    2
```

This one actually works (after a fashion):

```
crabs %>% count(Force < 5, Species)
```

```
## # A tibble: 5 x 3
##   `Force < 5` Species                n
##   <lgl>       <chr>              <int>
## 1 FALSE       Cancer productus      12
## 2 FALSE       Hemigrapsus nudus      4
## 3 FALSE       Lophopanopeus bellus  10
## 4 TRUE        Hemigrapsus nudus     10
## 5 TRUE        Lophopanopeus bellus   2
```

except that it also counts the crabs where `Force` is greater than 5, which we don't care about. Two points for that, but you can rescue yourself like so if you go this way:

```
crabs %>% count(Force < 5, Species) %>%
  filter(`Force < 5`)
```

```
## # A tibble: 2 x 3
##   `Force < 5` Species                 n
##   <lgl>       <chr>                <int>
## 1 TRUE        Hemigrapsus nudus       10
## 2 TRUE        Lophopanopeus bellus     2
```

which displays only the rows in the above for which the column `Force < 5` is `TRUE`. Note that you will need to put backticks around the column name here because it is not a legal column name by itself. 2.5 if you forget those and get everything else.

If you didn't remember `count`, you can also use `summarize` with `n()`, as long as you remember to `group_by` the right thing first:

```
crabs %>%
  filter(Force < 5) %>%
  group_by(Species) %>%
  summarize(count = n())
```

```
## # A tibble: 2 x 2
##   Species              count
##   <chr>                <int>
## 1 Hemigrapsus nudus       10
## 2 Lophopanopeus bellus     2
```

Full credit for this, done correctly, or for other Tidyverse solutions that will work.

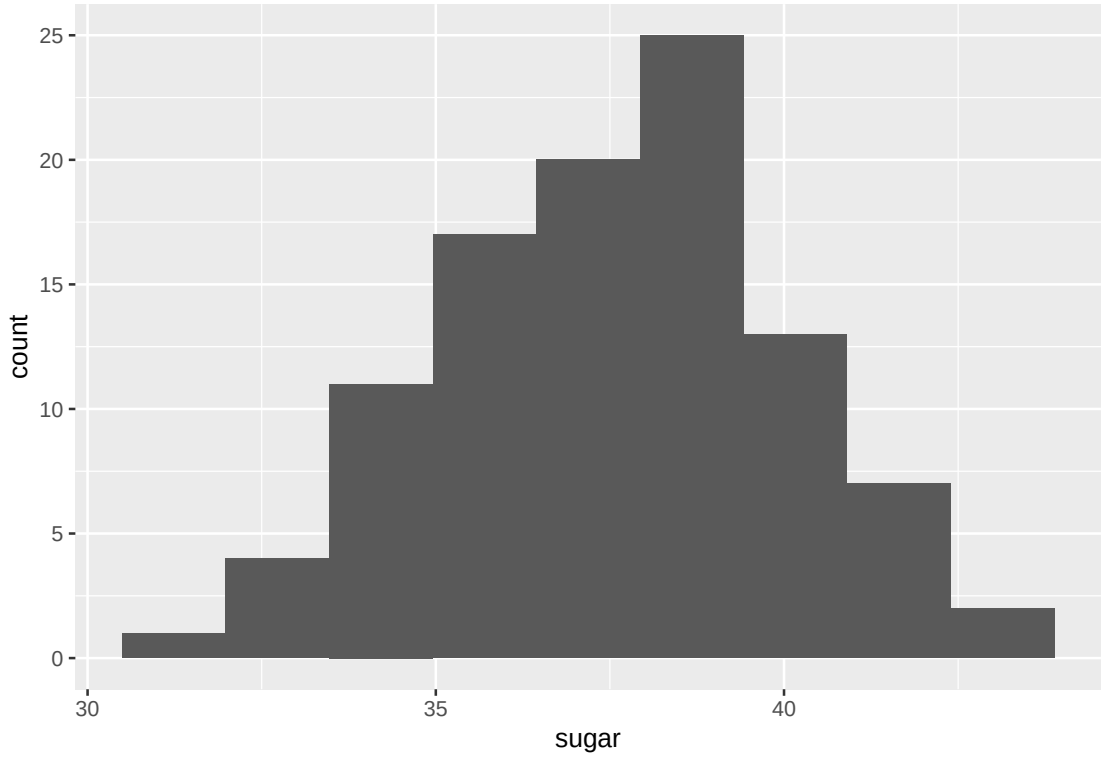
3. A well-known cereal manufacturer has been adjusting the recipe for its frosted flakes so that a serving of the cereal should contain less sugar. One hundred servings of the latest batch of frosted flakes have been obtained, and some of the data, in grams of sugar per serving, is shown in Figure 5. The dataframe is called `cereal_sugar`.

   (a) [3] A graph is shown in Figure 6. What code was used to draw the graph?

My answer:

It's a histogram; the dataframe is called `cereal_sugar` and the one column in it is called `sugar`. There are nine bins (count them!):

```
ggplot(cereal_sugar, aes(x = sugar)) + geom_histogram(bins = 9)
```



Minus one per error, down to a minimum of one point if you have something substantial correct. Minus a half point if you are off by one on the number of bins, on the basis that you knew what to count.

(b) [3] Last year's recipe for frosted flakes contained a mean of 38.5 grams of sugar per serving. What code would carry out a suitable hypothesis test, bearing in mind what the cereal manufacturer would like to have evidence for?

My answer:

The cereal manufacturer wants to have evidence that the amount of sugar per serving has *decreased*: that is, it is now less than 38.5 grams. Hence, the null hypothesis is that the mean is (still) 38.5, and the alternative is that the mean is less than 38.5. Hence the code you need is either this:

```
with(cereal_sugar, t.test(sugar, mu = 38.5, alternative = "less"))
```

```
##
##  One Sample t-test
##
## data:  sugar
## t = -3.6638, df = 99, p-value = 0.0002007
## alternative hypothesis: true mean is less than 38.5
## 95 percent confidence interval:
##      -Inf 38.00568
## sample estimates:
## mean of x
##    37.596
```

or this:

```
t.test(cereal_sugar$sugar, mu = 38.5, alternative = "less")
```

```
##
##  One Sample t-test
##
## data:  cereal_sugar$sugar
## t = -3.6638, df = 99, p-value = 0.0002007
## alternative hypothesis: true mean is less than 38.5
## 95 percent confidence interval:
##      -Inf 38.00568
## sample estimates:
## mean of x
##    37.596
```

(c) [3] The test you gave code for in the previous part has P-value 0.0002. Bearing in mind what the cereal manufacturer would like to know, what do you conclude from the test and why, in the context of the data?

My answer:

The null hypothesis (from the previous part) was that the mean sugar content was 38.5 grams

per serving, and the alternative was that it was less than that. With a P-value of 0.0002, we reject the null hypothesis in favour of the alternative, and conclude that the mean sugar content is less than it was last year. (Or, "we have evidence that the mean sugar content is less than it was last year", or similar wording.)

Three points for leading your reader through the process: stating (or otherwise implying that you know) what the hypotheses are, saying what the P-value is, what that implies for the hypotheses (*without* using the word "accept" anywhere), and what that means in the context of the data. Two points if a step of the logic is missing, and only one if you say "reject the null hypothesis" and stop there. The point of a hypothesis test, as I have said elsewhere, is to make a decision or to take action; in this case, the manager who hired you wants to know whether the mean sugar content really has gone down (or not), so that is what your answer needs to finish by saying.

I put the phrase "bearing in mind what the cereal manufacturer would like to know" in the question again to remind you to check what the cereal company wants evidence for: that the sugar content has gone *down*. 2.5 points if you conclude that the sugar content has *changed*, even if you did a two-sided test in the previous part (because it means that you did not pay attention to this phrase). If you did a one-sided test in the previous part, and then make a two-sided conclusion here, 2 is your maximum, because your logic is flawed. Expect the grader to check back if you make a two-sided conclusion here.

(d) [4] The cereal manufacturer's statistician is not happy with Figure 6, so decides to obtain a bootstrap sampling distribution of the sample mean. Some code is shown at the top of Figure 7. Unfortunately, the statistician spilled his coffee on the printout of his code, and only the four lines of code shown in Figure 7 could be read. What code was missing, and which line of the code shown should it come after?

My answer:

The correct (five) lines of code are these:

```
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(cereal_sugar$sugar, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(x = my_mean)) + geom_histogram(bins = 12)
```

so there is one missing line, this one:

```
mutate(my_sample = list(sample(cereal_sugar$sugar, replace = TRUE)))
```

which goes after the `rowwise` line. Strictly, it should have a pipe symbol `%>%` on the end of it, but I am most interested in the contents of the line.

The hint is that, on reading my code above the graph, you ought to get to the third line and think "where did `my_sample` come from?", and hence the missing code should obtain it.

Three points for the code, and one for saying where it goes.

(e) [2] Why did you need a `list` around something in your code in the previous part? Explain briefly.

My answer:

A hint here is that your code in the previous part should have a `list` in it somewhere!

The `list` is around the `sample`. This is because the bootstrap sample contains (here) 100 values, which is too much to go into one cell of the dataframe, so we need `my_sample` (or whatever you called it) to be a list-column to contain all the sample values for each sample. (The column I called `my_mean` does *not* need a `list` in its definition because a sample mean is a single number and that will fit into one cell.)

(f) [3] What do you conclude from the graph at the bottom of Figure 7? Explain briefly. (You may assume that the graph came from the code at the top of the Figure, but including the missing code that you supplied earlier.)

My answer:

This is a histogram of the bootstrap sampling distribution of the sample mean (one point). This distribution is very close to normal in shape (one point), so the sample size is big enough to overcome the non-normality in the distribution of sugar content per serving, and the t-test we did is entirely appropriate (one point).

One of the points here is for saying what the histogram is a distribution *of*. It is *not* a histogram of the original data (the sugar content values). The other points are for saying that the shape is normal and that this means that the t-test is OK.

Extra: The histogram of the sugar content values is the graph in Figure 6. I chose the number of bins in that Figure to make it look as non-normal as I could, although the actual distribution is already close to normal, and even for a much smaller sample size, the t-test would be entirely fine. With the large sample size we have, there is no doubt about this t-test at all.

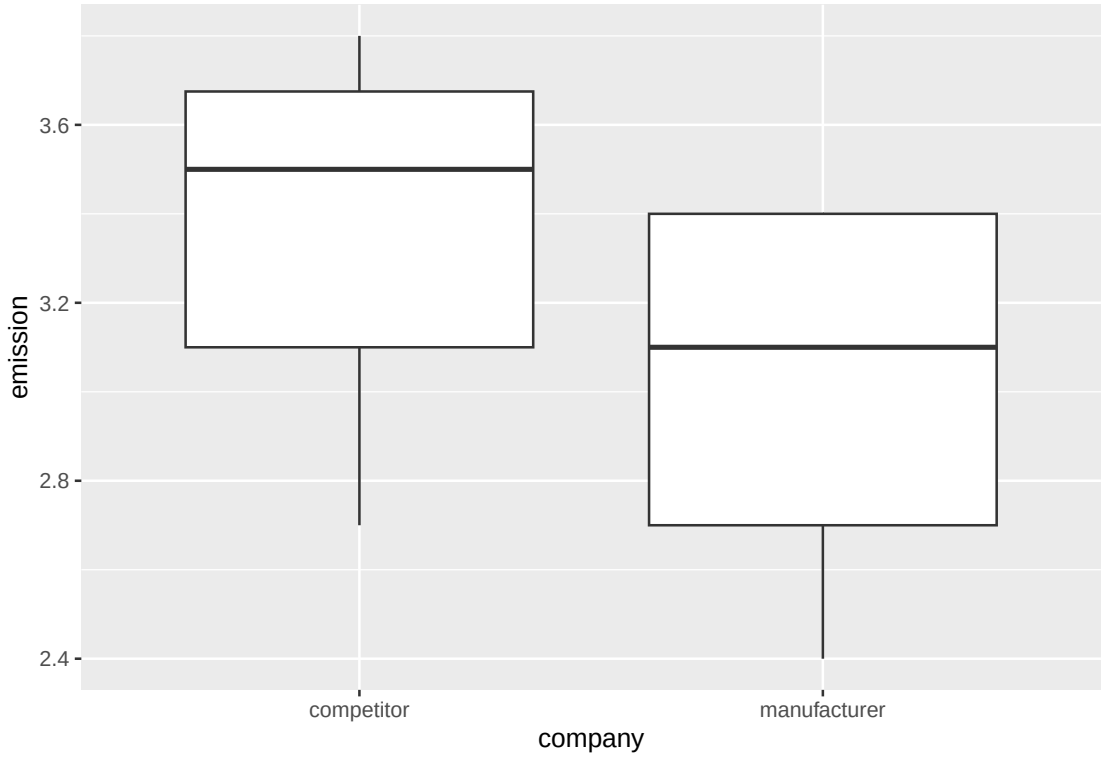
4. A company believes that it has a "cleaner" manufacturing process than other companies making the same product. To assess this belief, the company measures the carbon monoxide emissions from smoke stacks at its factory, and from smoke stacks at a competitor's factory. The data are shown in Figure 8. The dataframe is called `Monoxide`. A lower carbon monoxide emission indicates a cleaner manufacturing process.

(a) [3] What code would make an appropriate graph of this dataset?

My answer:

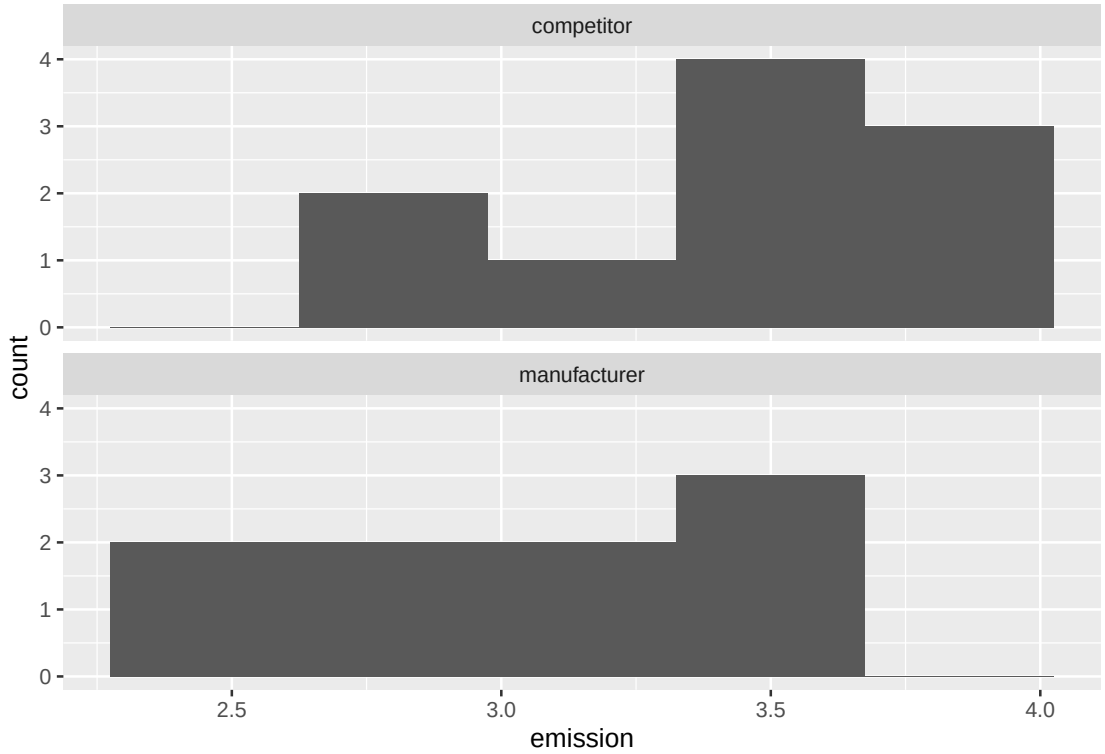
This needs to be a boxplot with the categorical variable `company` as the `x` and the quantitative variable `emission` as the `y`:

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



At this point, we are interested in a *general* picture of the distributions, not in normality specifically, so a normal quantile plot is premature. The only other suitable plot at this stage is facetted histograms (above and below):

```
ggplot(Monoxide, aes(x = emission)) + geom_histogram(bins = 5) +
  facet_wrap(~company, ncol = 1)
```



We can use either of these two plots to compare centre (`competitor` is higher on average), spread (similar), shape (both skewed to left). Hence, either of the plots is good at this point.

(b) [3] A test is shown in Figure 9. What do you conclude from the results shown in the Figure, and why, in the context of the data?

My answer:

The null hypothesis is that the mean emissions from the two factories are the same, and the alternative is that the mean emission from the competitor's factory is greater (or that the emission from the manufacturer's factory is less). The P-value is 0.02465, which is less than 0.05, so we have strong enough evidence to reject the null in favour of the alternative, and thus we conclude that the mean carbon monoxide emission at the manufacturer's factory is less than that at the competitor's factory.

Note that the two levels of the categorical variable `company` are `manufacturer` and `competitor`, and the second of these is the first one alphabetically, so the `alternative` is relative to `competitor`, and the competitor's carbon monoxide emissions will need to be greater in order to justify the manufacturer's belief.

As for the other hypothesis testing part, three points for leading the reader through the process

(that's the "and why") correctly, including a statement of the hypotheses or otherwise making it clear that you know what they are, saying what the P-value is, drawing a conclusion from the P-value about the hypotheses, and putting that conclusion in the context of the data. Two points for one piece of mistaken or missing logic, and only one if you "reject the null hypothesis" and say no more.

(c) [2] Why is there no `mu` in the code in Figure 9?

My answer:

The standard null hypothesis for a two-sample t-test is that the two groups have the *same* mean as each other, so there is no mean or difference in means to give (as there is with a one-sample t-test).

Say something about how the nature of the two-sample test is different from the one-sample one, where you have to give a value for the mean in the null hypothesis.

"Because the two-sample test does not use `mu`" is not insightful enough. This is in fact not even true; you can use a two-sample test to test that the means differ by, say, 10 (which you do by saying `mu = 10`) against an alternative that the means differ by something else. This is, as you might guess, rare, but it is possible. `mu = 0` is the default, so if you don't specify a `mu` in a two-sample t-test, you'll test a null that the two populations have the same mean.

(d) [3] The company wants to obtain a 90% confidence interval for the difference in mean emission between the two groups. What code would get this interval?

My answer:

Take the code at the top of Figure 9, and alter it to suit: take out the `alternative` (a CI is two-sided in this course), and include a `conf.level`:

```
t.test(emission ~ company, data = Monoxide, conf.level = 0.90)
```

```
##
##  Welch Two Sample t-test
##
## data:  emission by company
## t = 2.1187, df = 16.842, p-value = 0.04929
## alternative hypothesis: true difference in means between group competitor and group manufac
## 90 percent confidence interval:
##  0.06802198 0.69420024
## sample estimates:
##   mean in group competitor mean in group manufacturer
##                   3.370000                   2.988889
```

Of course, you won't know what the results are, but this one is the usual thing: we know that two groups have different mean emission, but the confidence interval says that we don't know much about how big the difference is. This is where we are hurt by the small samples; having bigger samples would make the confidence interval shorter.

(e) [2] A graph is shown in Figure 10. This may or may not be the same as the graph you gave code for in an earlier part. Does this graph give you any doubts about the appropriateness of your t-test? Explain briefly.

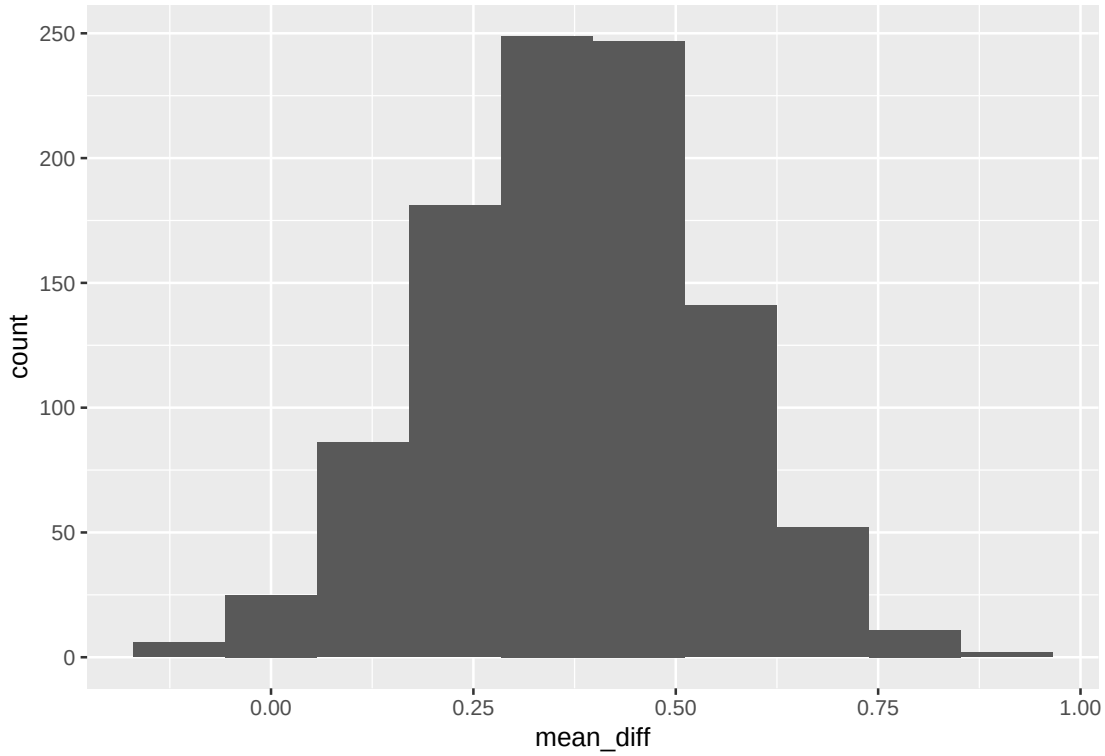
My answer:

The t-test requires both groups to have emission values that are sufficiently normal given the sample size. You can reasonably say that at least the `Manufacturer` group values, and possibly both groups, have a distribution of emission values that is skewed to the left. Also, you need to assess this in the light of the sample sizes: there are only nine emission values for `manufacturer` and ten for `competitor` (count them), and so you are justified in wanting better normality than this in order to trust the t-test.

One point each for discussion of normality, and discussion of sample size. You can end up trusting the t-test or not (it doesn't matter); I am interested in your thought process more than your final answer.

Extra: this is the only information you have available to you here, so you have to make a call on what you see. I also did a two-sample bootstrap, which I explain below:

```
Monoxide %>% filter(company == "manufacturer") -> manu
Monoxide %>% filter(company == "competitor") -> comp
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample_manu = list(sample(manu$emission, replace = TRUE)),
         my_sample_comp = list(sample(comp$emission, replace = TRUE))) %>%
  mutate(mean_manu = mean(my_sample_manu),
         mean_comp = mean(my_sample_comp)) %>%
  mutate(mean_diff = mean_comp - mean_manu) %>%
  ggplot(aes(x = mean_diff)) + geom_histogram(bins = 10)
```



First, I get the emission values for `manufacturer`, which I call `manu`, and then the ones for `competitor` (`comp`). The pipeline is like the ones for one-sample t, except that now I am drawing bootstrap samples from *each* sample, then working out the mean for *each*. Finally, I work out the difference between the means of each bootstrap sample and make a histogram of those.

I think you'll agree that this actually looks pretty normal, and so the two-sample t-test is actually all right. My take on this and my discussion above (and probably your discussion too) is that sample sizes of 9 and 10, while not big, actually *are* big enough to overcome the (fairly mild) skewness in the boxplots. I say "fairly mild" because the whiskers are not wildly different in length, and there are no outliers at the end of the long tail, as you might expect to see when the skewness is more extreme. The other thing to say is that both distributions are skewed *in the same direction*, so by looking at the difference in sample means (as the two-sample t statistics do), the skewness will tend to cancel out: if you get an unusually low value in one sample, you may well get an unusually low one in the other as well, and these will have the same effect on the sample means (pull them both downward, so that the *difference* in means may not be affected very much).

5. The beta distribution is a continuous distribution on $[0, 1]$. It has two parameters, called `a` and `b`. The R function `rbeta` obtains random samples from a beta distribution. It has three inputs. In order, these are the sample size, `a`, and `b`.

   (a) [2] What R code would draw a random sample from a beta distribution with `a` of 3, `b` of 6, of size 20?

My answer:

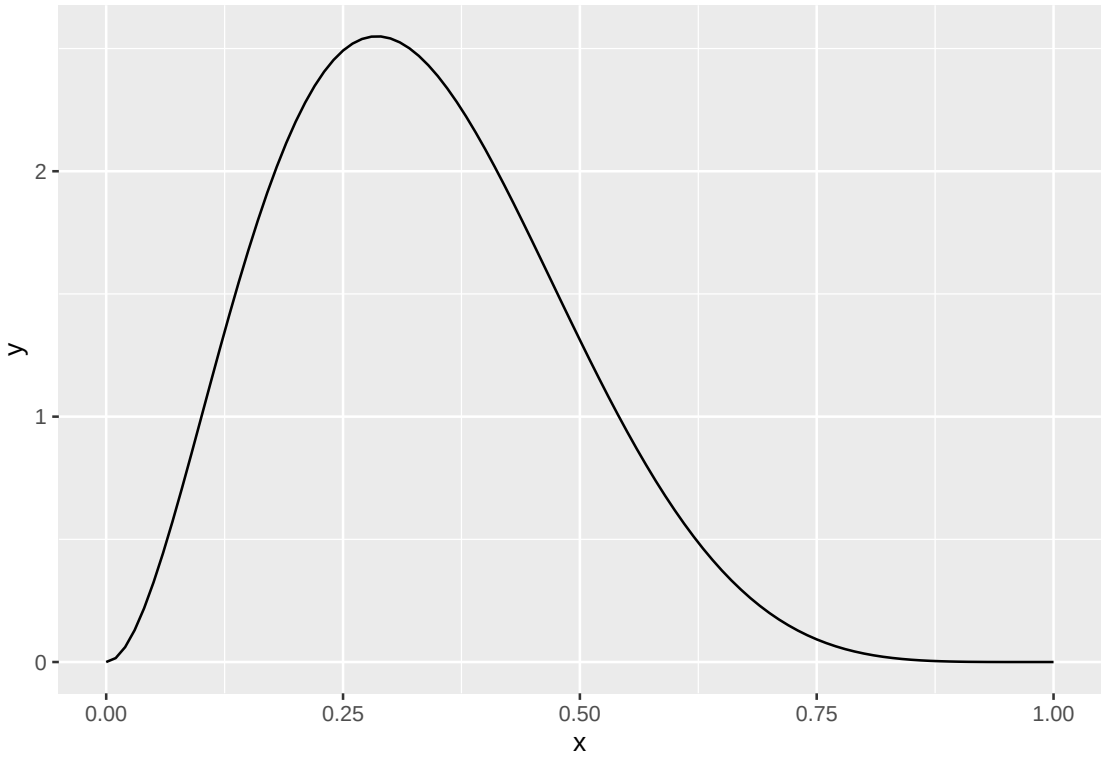
This is meant to be a warmup:

```
rbeta(20, 3, 6)
```

```
##  [1] 0.68584499 0.21166108 0.28651305 0.21854024 0.37296282 0.47404401
##  [7] 0.16581609 0.49203215 0.33406446 0.56114949 0.09937896 0.14949966
## [13] 0.60179169 0.50246108 0.21169624 0.34292688 0.66043122 0.31336089
## [19] 0.42412838 0.52634978
```

Extra: This beta distribution is actually skewed to the right, with a mean less than 0.5. Its density function looks like this:

```
tibble(x = seq(0, 1, 0.01)) %>%
  mutate(y = dbeta(x, 3,6)) %>%
  ggplot(aes(x = x, y = y)) + geom_line()
```



The graph looks like a curve, but it is actually a lot of short line segments joined together.

(b) [4] Suppose you are testing the null hypothesis that the population mean is 0.5, against the alternative hypothesis that it is less than 0.5, and the data is assumed to come from a beta distribution with `a` of 4 and `b` of 5. What code would enable you to estimate the power of a t-test under these assumptions, with a sample size of 30? (Note: the mean of this beta population is $4/(4+5) \simeq 0.44$.)

My answer:

This is as in the (power by simulation) lecture, with the right substitutions: `rbeta` for the sampling, with the right sample size and values for the parameters, and the right null mean:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(rbeta(30, 4, 5))) %>%
  mutate(t_test = list(t.test(my_sample,
                              mu = 0.5, alternative = "less"))) %>%
  mutate(p_value = t_test$p.value) %>%
  count(p_value <= 0.05)
```

```
## # A tibble: 2 x 2
## # Rowwise:
##   `p_value <= 0.05`     n
##   <lgl>             <int>
## 1 FALSE               420
## 2 TRUE                580
```

(c) [2] The output from your code in the previous part is shown in Figure 11. Your boss asks you what sample size you would need to obtain power 0.75. What **change** would you make to your code of the previous part in an attempt to answer this question? Explain briefly.

My answer:

The power as shown in the Figure is less than 0.75, so you need to try a larger sample size. Note that the question says you only need to "attempt" to answer your boss's question, so *any* larger sample size will do.

Below, I try a sample size of 40, so my third line becomes this, with the 30 changed to 40:

`mutate(my_sample = list(rbeta(40, 4, 5))) %>%`

This line (or anything indicating that the 30 that was in this line needs to become something bigger) is full credit, with any sample size larger than 30.

This is actually what happens with a sample size of 40:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(rbeta(40, 4, 5))) %>%
  mutate(t_test = list(t.test(my_sample,
                              mu = 0.5, alternative = "less"))) %>%
  mutate(p_value = t_test$p.value) %>%
  count(p_value <= 0.05)
```

```
## # A tibble: 2 x 2
## # Rowwise:
##   `p_value <= 0.05`     n
##   <lgl>             <int>
## 1 FALSE               264
## 2 TRUE                736
```

which turned out pretty well. (40 was just a guess, as your answer will have to be. To get to a power of 0.75, you would need to try again with a sample size a little bigger, something like 45.) A technical point is that with 1000 simulations, we are typically estimating the power to within about 3 percentage points (95% of the time), so we may not get much closer than this. It might be worth upping the number of sims to 10000 if we want to get the sample size more accurately.

Extra: we saw in an Extra above that a beta distribution with $a < b$ is skewed to the right, but only moderately so. This one, with parameters 4 and 5, is less right-skewed than one with parameters 3 and 6.

A sample size of 30 or so is likely to be big enough to make a t-test perfectly safe, and also to mean that it doesn't matter that the population distribution is actually skewed. Our beta distribution has mean

```
a <- 4
b <- 5
beta_mean <- a / (a + b)
beta_mean
```

```
## [1] 0.4444444
```

and variance (taken from the R help for `rbeta`, because I get this wrong if I try to remember it):

```
beta_var <- a * b / ((a + b)^2 * (a + b + 1))
beta_var
```

```
## [1] 0.02469136
```

so we can get a (possibly good) approximation to the desired sample size out of `power.t.test`:

```
power.t.test(delta = 0.5 - beta_mean, power = 0.75,
             sd = sqrt(beta_var), type = "one.sample", alternative = "one.sided")
```

```
##
##      One-sample t test power calculation
```

```
##
##                n = 44.41796
##            delta = 0.05555556
##               sd = 0.1571348
##        sig.level = 0.05
##            power = 0.75
##      alternative = one.sided
```

A sample size of 45 seems entirely consistent with what I found in my simulations.

6. Heights of 100 randomly-sampled adult males were measured, in whole numbers of inches. Some of the dataset is shown in Figure 12, and a graph is shown in Figure 13. Our aim is to test whether the population median height is 71.5 inches, or whether there is evidence that it is different from this.

   (a) [2] A sign test is run with the output shown in Figure 14. What code was used to obtain these results?

My answer:

This:

```
sign_test(male_heights, heights, 71.5)
```

```
## $above_below
## below above
##    68    32
##
## $p_values
##   alternative      p_value
## 1       lower 0.0002043886
## 2       upper 0.9999084284
## 3   two-sided 0.0004087772
```

Dataframe, column, and null median. No need for an `alternative` because the output gives you two-sided and both one-sided P-values.

(b) [2] Why might you guess that the P-value in Figure 14 will be small, even before you look at the P-value? (You can base your answer on any of the output, *except* for the P-values.)

My answer:

Look first at the counts of values above and below the null median at the top of the output. These are 32 and 68 respectively (the "below" one is given first). In 100 trials, this is a very unbalanced number of successes (above) and failures (below); if the null hypothesis is true, you would expect to see values much closer to 50–50. (The small P-value actually means that an above-below split this uneven is very unlikely if the median really is 71.5 inches.)

A fair number of people got at this by looking at the histogram, which is also fine: either see that there are more observations below 71.5 than above (substantially more), or make a call about where you think the sample median is (maybe 70 inches or a bit less). I'm happy with either of those, though an answer that there are (many) more observations below 71.5 inches than above is stronger, because you might have a sample median a bit less than 71.5, and because of large variability and/or small sample size, it's not significantly different from 71.5. Here, though, our sample size of 100 is large and there is not much variability, so our sample allows us to estimate the population median height accurately (a confidence interval for the population median would be short).

(c) [3] What do you conclude from the sign test, in the context of the data? Explain briefly.

My answer:

We are doing a two-sided test ("different from this" in the question), so we need the two-sided P-value, 0.0004. This is way smaller than 0.05, so we reject the null hypothesis in favour of the (two-sided) alternative, and conclude that the population median height is not equal to 71.5 inches.

(If you prefer, "we have evidence that the population median height is not 71.5 inches", or similar.)

Make sure you make it clear which P-value you are using, and that you know whether you are doing a one-sided or a two-sided test. A good habit to have is to say what the P-value *is* whenever you do a test. On an exam, that makes it clear that you have the correct one, and outside of an exam situation, it enables your reader to draw their own conclusion if they happen to disagree with your choice of α.

There were a lot of sloppy answers here. It is not hard to get this one right if you:

- write down the null hypothesis (population median height is 71.5 inches)
- write down the alternative hypothesis (population median is not equal to 71.5 inches; see question)
- write down the correct P-value, 0.0004
- make a conclusion about the hypotheses (reject the null in favour of the alternative)
- translate the alternative into a statement about the data (conclude that the median height is not 71.5 inches)

The fact that `sign_test` gives you three P-values is to enable you to pick the *one* you want, based on the null and alternative *you* are testing. Don't try to draw three conclusions. There is also no need to go beyond "median height is not equal to 71.5 inches"; that *is* the conclusion from the test.

(d) [3] Part of a binomial table for $n = 100, p = 0.5$ is shown in Figure 15. The column `x` denotes the number of successes, and the column `prob` is the probability of obtaining exactly that many successes. The probabilities have been rounded to six decimals. (If only five decimals are shown, the last one is zero.) You may assume that the probability of 20 successes or less is 0 to the accuracy shown. Show how you can use this table to obtain the P-value for your sign test. (If you do not have a calculator, show the calculation you would do.)

My answer:

The P-value is the probability of obtaining a test statistic (number of heights above 71.5 inches) as extreme or more extreme than the value observed. The observed value was 32, which is less than the mean number of successes (50), so we need to find the probability of 32 or less. Using the values in the table, this probability is

```
0.000113 + 0.000052 + 0.000023 + 0.000010 + 0.000004 + 0.000002 + 0.000001
```

```
## [1] 0.000205
```

Calculator tip: ignore the decimal places and the initial zeros, and calculate this as $113 + 52 + \ldots + 1$, then put the decimal places back.

Our test is two-sided, though, and we might have observed a value in the other tail, so we have to double this to get our P-value:

```
2 * 0.000205
```

```
## [1] 0.00041
```

and this is more or less the same as the value given in the `sign_test` output (which was calculated with more decimal places).

Grading note: if you for some reason thought that this was a lower-tail test, you will lose something above for not doing a two-sided test, but getting 0.000205 here is then correct, so that answer would be full marks here in that case.

(e) [2] Based on what you have seen in this question, and assuming that the mean height is also of interest, do you think it would have been better to do a t-test here? Justify your answer briefly.

My answer:

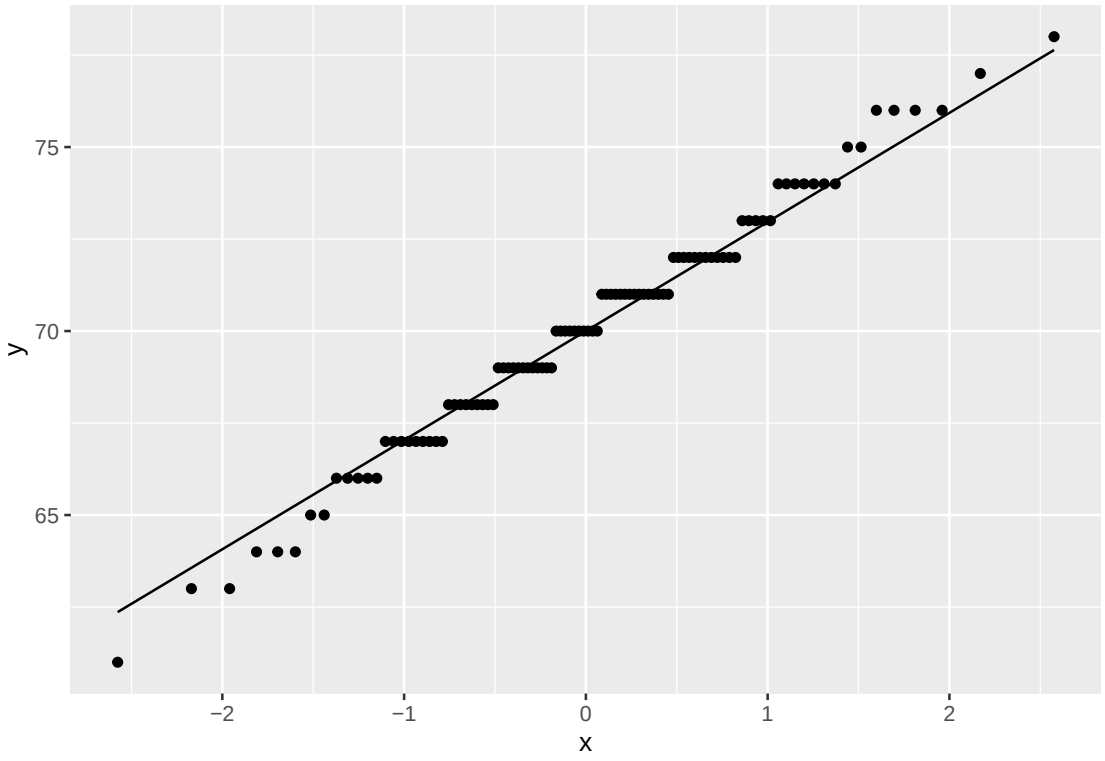
The histogram in Figure 13 indicates that the height values already have close to a normal distribution, and we have a large sample size ($n = 100$), so there are no problems with the assumptions for a t-test. (One point for an appreciable fraction of that.)

The second point is for saying that when either test is good, you should do a t-test, because it makes better use of the data: it uses the actual values, as opposed to counting only whether each one is above or below something (and throwing away how *far* above or below it is).

For example, if you have a distribution with outliers, you should probably not be using the mean (because it will be distorted by the outliers), and in addition, how far above or below the null median an outlier is may not be very reliable (an outlier could have been even more extreme than it was).

Extra 1: a normal quantile plot of the heights might give a better picture of the normality than a histogram does:

```
ggplot(male_heights, aes(sample = heights)) + stat_qq() + stat_qq_line()
```



The horizontal lines are because of the discreteness of the data (each height was only recorded to the nearest inch), but apart from that, the data follow the line very well. One of the reasons the normal distribution has that name is that it can be used to model "normal variation" in something like heights. (This goes back to people like Galton.) So I actually suspected, even before I began work on this question, that a t-test would be fine, but I wanted you to work through a sign test question and then think about whether a sign test was really necessary for the data I gave you.

Extra 2: here is how a t-test would come out:

```r
t.test(male_heights$heights, mu = 71.5)
```

```
##
##  One Sample t-test
##
## data:  male_heights$heights
## t = -4.4653, df = 99, p-value = 2.123e-05
## alternative hypothesis: true mean is not equal to 71.5
## 95 percent confidence interval:
##  69.37679 70.68321
## sample estimates:
## mean of x
##     70.03
```

The P-value is even smaller than the one from the sign test, about 20 times smaller. This indicates that the evidence against the null hypothesis is even stronger when you use a t-test, because the t-test uses the data more efficiently, and were are here in a situation where the distance from the null mean *can* be relied upon.

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

Figures

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

Figure 1: Packages

```
Force+Height+Species
3.2+5+Hemigrapsus nudus
6.4+6+Hemigrapsus nudus
2+6.4+Hemigrapsus nudus
2+6.5+Hemigrapsus nudus
4.9+6.6+Hemigrapsus nudus
3+7+Hemigrapsus nudus
2.9+7.9+Hemigrapsus nudus
9.5+7.9+Hemigrapsus nudus
4+8+Hemigrapsus nudus
7.4+8.3+Hemigrapsus nudus
2.4+8.8+Hemigrapsus nudus
4+12.1+Hemigrapsus nudus
5.2+12.2+Hemigrapsus nudus
2.1+5.1+Lophopanopeus bellus
8.7+5.9+Lophopanopeus bellus
2.9+6.6+Lophopanopeus bellus
6.9+7.2+Lophopanopeus bellus
8.7+8.6+Lophopanopeus bellus
```

Figure 2: Crab claws data (some)

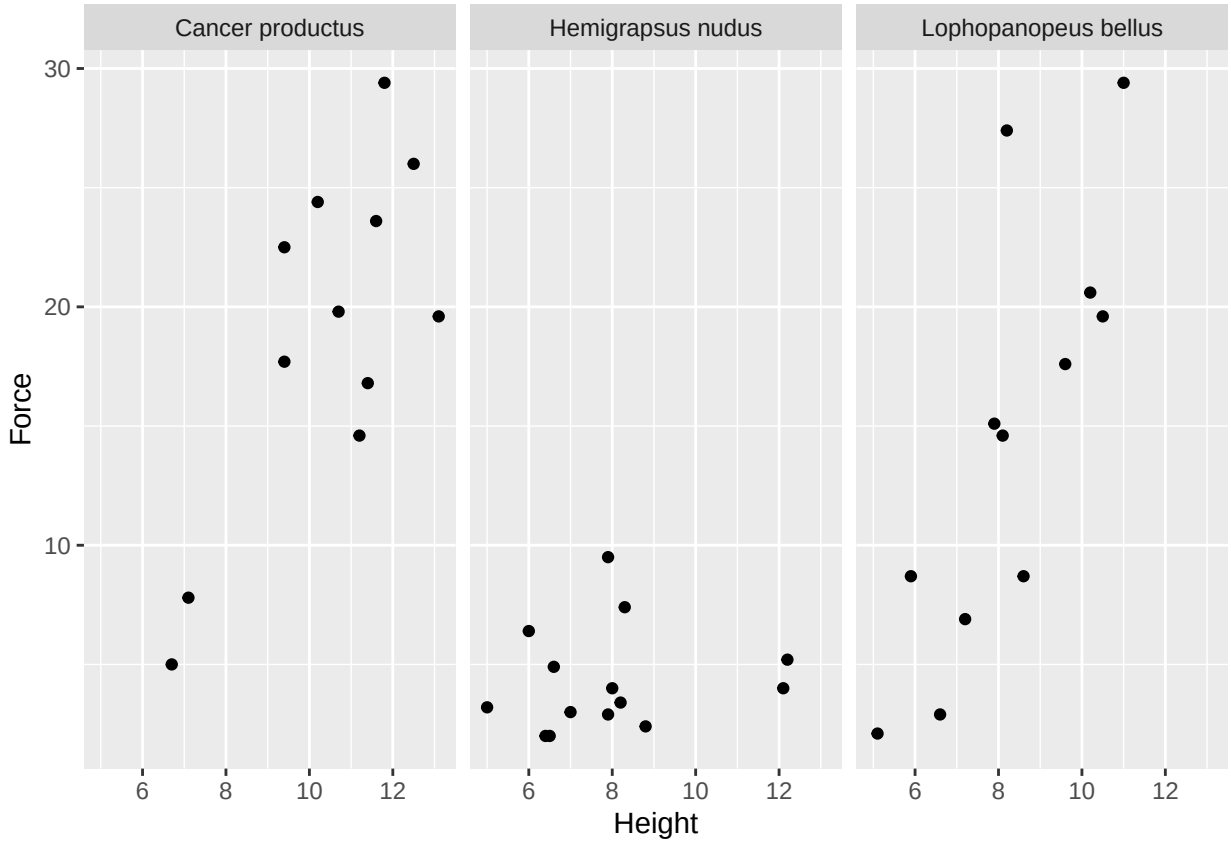


Figure 3: Crab claws graph

```
## # A tibble: 38 x 3
##    Force Height Species
##    <dbl>  <dbl> <chr>
##  1   3.2    5   Hemigrapsus nudus
##  2   6.4    6   Hemigrapsus nudus
##  3   2      6.4 Hemigrapsus nudus
##  4   2      6.5 Hemigrapsus nudus
##  5   4.9    6.6 Hemigrapsus nudus
##  6   3      7   Hemigrapsus nudus
##  7   2.9    7.9 Hemigrapsus nudus
##  8   9.5    7.9 Hemigrapsus nudus
##  9   4      8   Hemigrapsus nudus
## 10   3.4    8.2 Hemigrapsus nudus
## # ... with 28 more rows
```

Figure 4: Crab claws data (some)

```
cereal_sugar
```

```
## # A tibble: 100 x 1
##    sugar
##    <dbl>
##  1  36.3
##  2  33.2
##  3  39
##  4  37.3
##  5  40.7
##  6  38.4
##  7  35.8
##  8  36
##  9  37.9
## 10  42.6
## # ... with 90 more rows
```

Figure 5: Cereal sugar data (some)

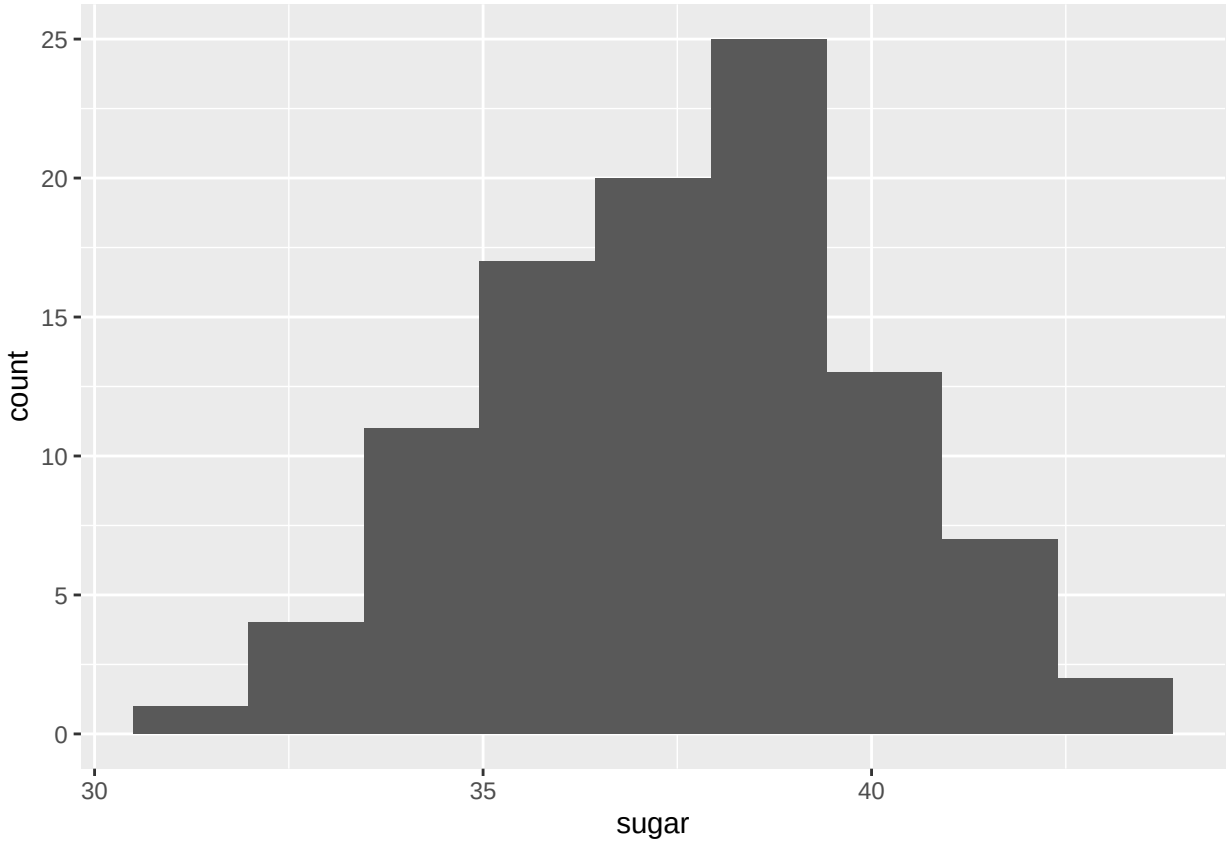


Figure 6: Cereal sugar graph

```
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(x = my_mean)) + geom_histogram(bins = 12)
```

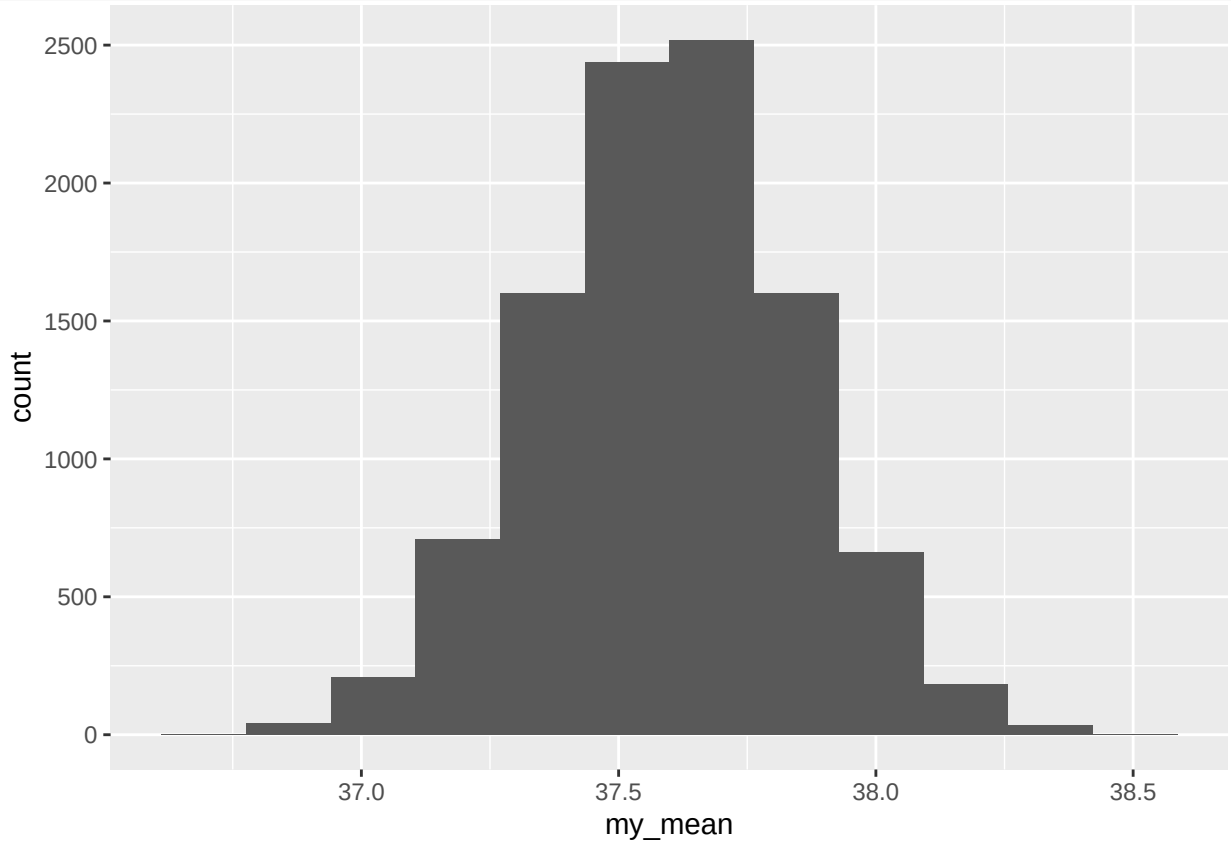


Figure 7: Cereal sugar summary bootstrap sampling distribution of sample mean (one line of code missing)

```
Monoxide
```

```
## # 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 8: Carbon monoxide emissions data

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

```
##
##  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 g
## 95 percent confidence interval:
##  0.06802198        Inf
## sample estimates:
##   mean in group competitor mean in group manufacturer
##                   3.370000                   2.988889
```

Figure 9: Carbon monoxide emissions t-test

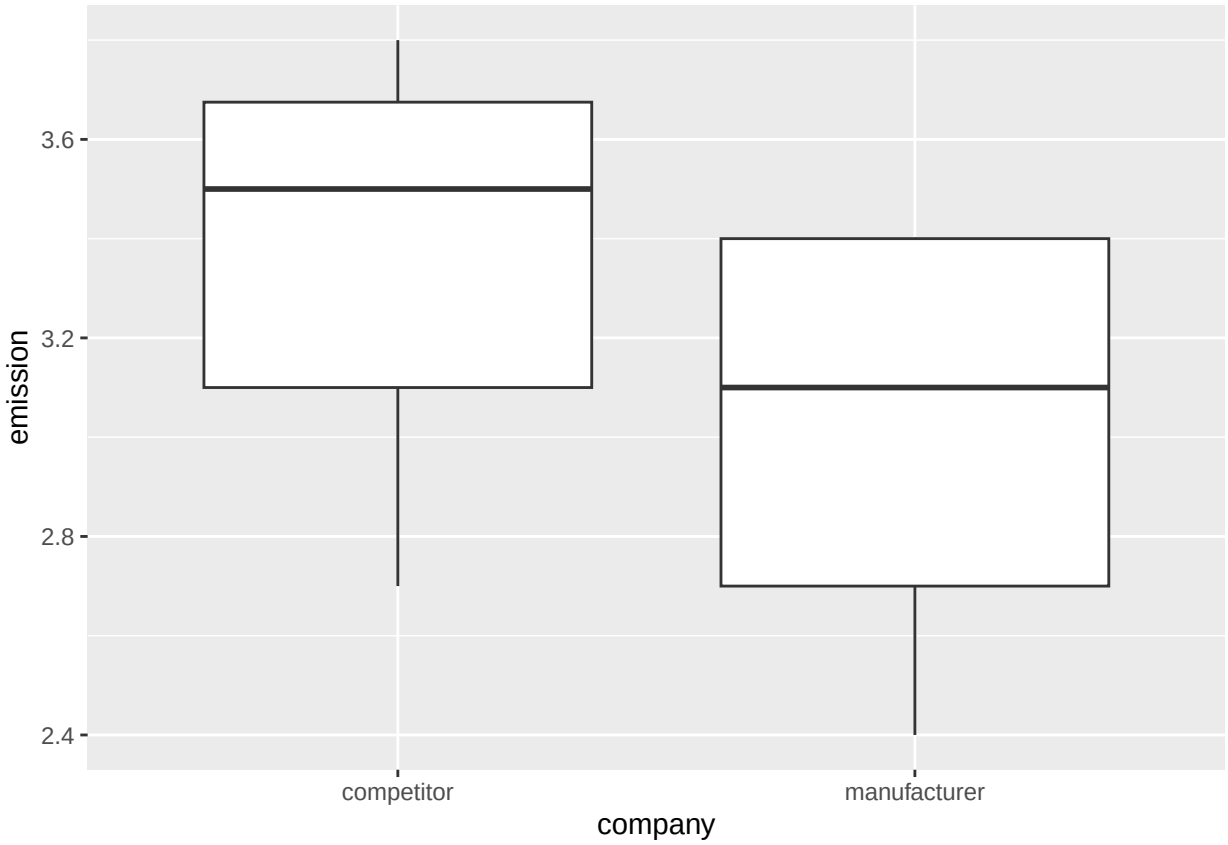


Figure 10: Carbon monoxide emissions graph

```
## # A tibble: 2 x 2
## # Rowwise:
##   `p_value <= 0.05`     n
##   <lgl>             <int>
## 1 FALSE               412
## 2 TRUE                588
```

Figure 11: Beta power simulation results

```
male_heights
```

```
## # A tibble: 100 x 1
##    heights
##      <int>
##  1      71
##  2      67
##  3      69
##  4      70
##  5      68
##  6      63
##  7      68
##  8      72
##  9      70
## 10      70
## # ... with 90 more rows
```

Figure 12: Heights data (some)

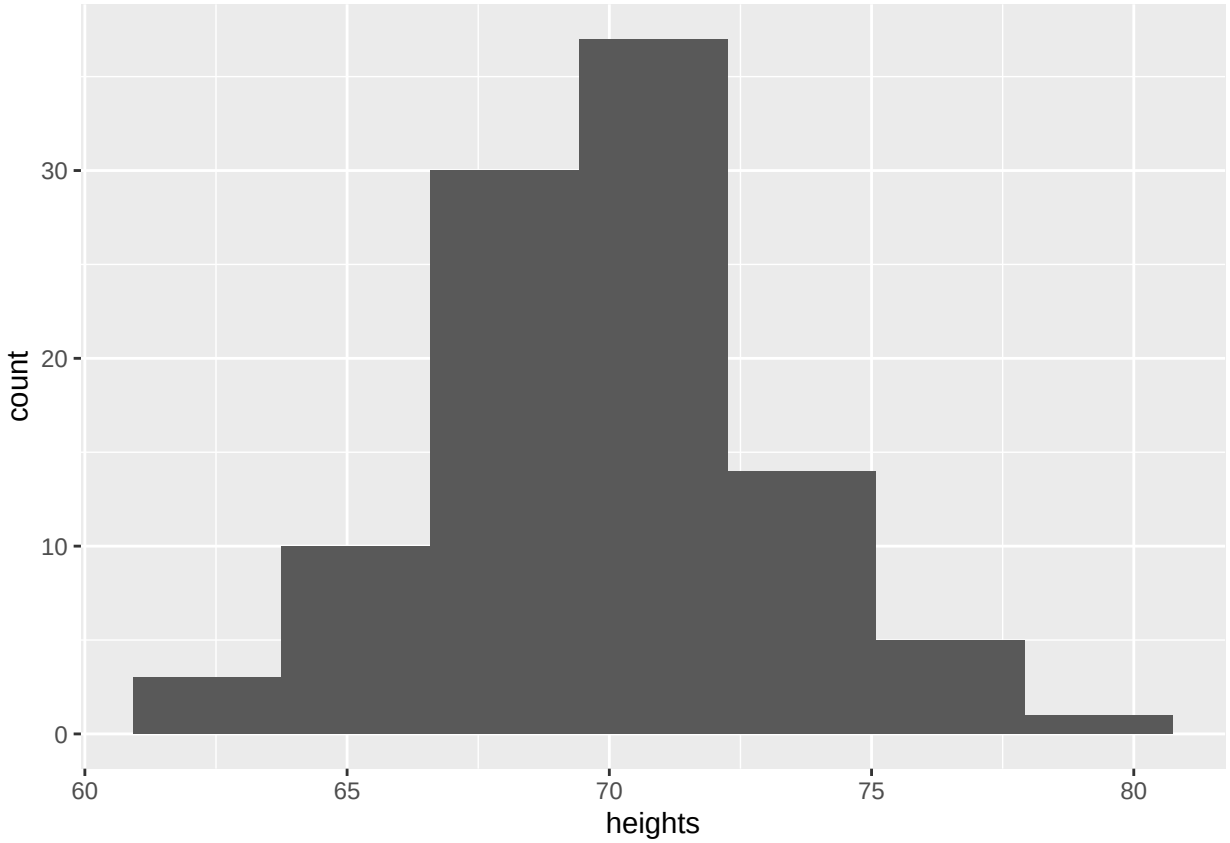


Figure 13: Heights graph

```
## $above_below
## below above
##    68    32
##
## $p_values
##   alternative      p_value
## 1       lower 0.0002043886
## 2       upper 0.9999084284
## 3   two-sided 0.0004087772
```

Figure 14: Heights sign test

```
x <- 20:35
tibble(x, prob = dbinom(x, 100, 0.5)) %>%
  mutate(prob = round(prob, 6))
```

```
## # A tibble: 16 x 2
##        x     prob
##    <int>    <dbl>
##  1    20 0
##  2    21 0
##  3    22 0
##  4    23 0
##  5    24 0
##  6    25 0
##  7    26 0.000001
##  8    27 0.000002
##  9    28 0.000004
## 10    29 0.00001
## 11    30 0.000023
## 12    31 0.000052
## 13    32 0.000113
## 14    33 0.000232
## 15    34 0.000458
## 16    35 0.000864
```

Figure 15: Binomial table for $n = 100, p = 0.5$