University of Toronto Scarborough
Department of Computer and Mathematical Sciences
STAC32 (K. Butler), Final Exam
December 16, 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. The state of Illinois recorded traffic fatality data for the years 1962 to 1971. The interest is in a possible reduction in deaths when new safety regulations went into effect after 1966. The data file is shown in Figure 2, which is saved in the file `il.txt` in the folder in which R Studio is currently running. The three columns are:

   - `when`: either `before` or `after`, whether before or after the new safety regulations went into effect
   - `year`: the actual year for which the number of deaths was recorded
   - `deaths`: the number of traffic-related deaths in that year, per 100 million vehicle miles.

   Note that when I ask for code in this exam, I want the code and *not* the output; in any case, you usually won't know what the output is going to be.

   (a) [3] What R code would read the data from the file into a dataframe called `illinois` and display that dataframe?

My answer:

A warmup. The data file in the Figure is in aligned columns (note that the number of spaces between columns varies), so you need `read_table`:

```
illinois <- read_table("il.txt")
```

```
-- Column specification --------------------------------------------------------
cols(
  when = col_character(),
  year = col_double(),
  deaths = col_double()
)
```

```
illinois
```

```
# A tibble: 10 x 3
   when    year deaths
   <chr>  <dbl>  <dbl>
 1 before  1962    4.9
 2 before  1963    5.1
 3 before  1964    5.2
 4 before  1965    5.1
 5 before  1966    5.3
 6 after   1967    5.1
 7 after   1968    4.9
 8 after   1969    4.7
 9 after   1970    4.2
10 after   1971    4.2
```

I give the output in my solutions so that you can see that my code works (or doesn't work);

you just need the code.

Optionally, you can define `il.txt` into another variable with a name like `my_file` first, and then read the data in from that.

Make sure that the thing you write between `read` and `table` looks like an underscore. We did not learn `read.table` in this course.

Two points for the `read_table`, correctly coded; one (rather cheap) point for displaying what you read in.

`read_delim` "works", but does not do the job:

```
read_delim("il.txt", " ")
```

```
New names:
* `` -> `...2`
* `` -> `...3`
* `` -> `...4`
```

```
Warning: One or more parsing issues, call `problems()` on your data frame for details,
e.g.:
  dat <- vroom(...)
  problems(dat)
```

```
Rows: 10 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: " "
chr (1): when
dbl (4): ...3, ...4, year, deaths
lgl (1): ...2

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.
```

```
# A tibble: 10 x 6
   when   ...2   ...3  ...4  year deaths
   <chr>  <lgl> <dbl> <dbl> <dbl>  <dbl>
 1 before NA     1962    NA   4.9   NA
 2 before NA     1963    NA   5.1   NA
 3 before NA     1964    NA   5.2   NA
 4 before NA     1965    NA   5.1   NA
 5 before NA     1966    NA   5.3   NA
 6 after  NA       NA  1967  NA     5.1
 7 after  NA       NA  1968  NA     4.9
 8 after  NA       NA  1969  NA     4.7
 9 after  NA       NA  1970  NA     4.2
10 after  NA       NA  1971  NA     4.2
```

The extra spaces get read in as additional empty columns, and since there are a variable number of them, the actual data (the years and the numbers of deaths) don't all come out in the same columns. So it "works" in the sense of not giving an error, but it does not work in the sense of giving us something we can use.

(b) [2] A graph is shown in Figure 3; what code would produce that graph?

My answer:

This is actually a scatterplot of the two quantitative variables (we are not plotting the column called `when` which is categorical). The number of deaths is on the y axis, and the time is on the x axis. On the graph are the observations plotted as points and the points are joined by lines:

```
ggplot(illinois, aes(x = year, y = deaths)) + geom_point() + geom_line()
```

One point for the `ggplot` piece, two half points for the two `geom_` pieces. If you add anything else, expect to lose a half point for each extraneous thing you add. Code for some other kind of graph is not likely to be worth anything.

I might also have coloured the before and after points differently, but I didn't, so `when` should not feature in your code here.

(c) [2] Look again at the graph is shown in Figure 3. What do you conclude from it that will interest the people that collected the data?

My answer:

Interest was in whether the number of deaths decreased after 1966. There seems to be a gradual increase up to 1966, but a steep drop afterwards (which suggests that the safety regulations had a positive effect).

Compare what happened before (up to) 1966 and what happened after.

(d) [2] Explain briefly why the graph in Figure 3 *does not* help us to decide whether to run a t-test or some other test, to see whether the number of deaths has decreased after 1966.

My answer:

To use a t-test, we need the numbers of deaths up to and after 1966 to have a normal distribution (because the sample size of 5 in each group is very small). The scatterplot (time plot) does not tell us this; to assess normality we need a boxplot or a normal quantile plot of the numbers of deaths up to and then after 1966.

Make the point somehow that the time plot doesn't give us what we need, or what we would need instead.

Saying that the graph doesn't include `when` is worth something (one point, by itself), but if you go that route, you also need to say why that stops us assessing whether a t-test is the thing to do.

(e) [2] Some output is shown in Figure 4. What do you conclude, in the context of the data? Explain briefly. (You may assume that it is reasonable to run this test.)

My answer:

We are testing the null hypothesis that the (population) mean number of deaths (per 100,000 vehicle miles) is the same up to and after 1966, against the alternative that it is less after 1966 (the `alternative` is saying how `after` compares with `before` in that order). Make sure you are clear about what is less than what. The P-value of 0.025 is less than 0.05, so we can reject the null in favour of this alternative, and conclude that the mean number of deaths is less after 1966 than it was before.

Make the grader's life easier (and thus, improve your chances of getting full points) by saying:

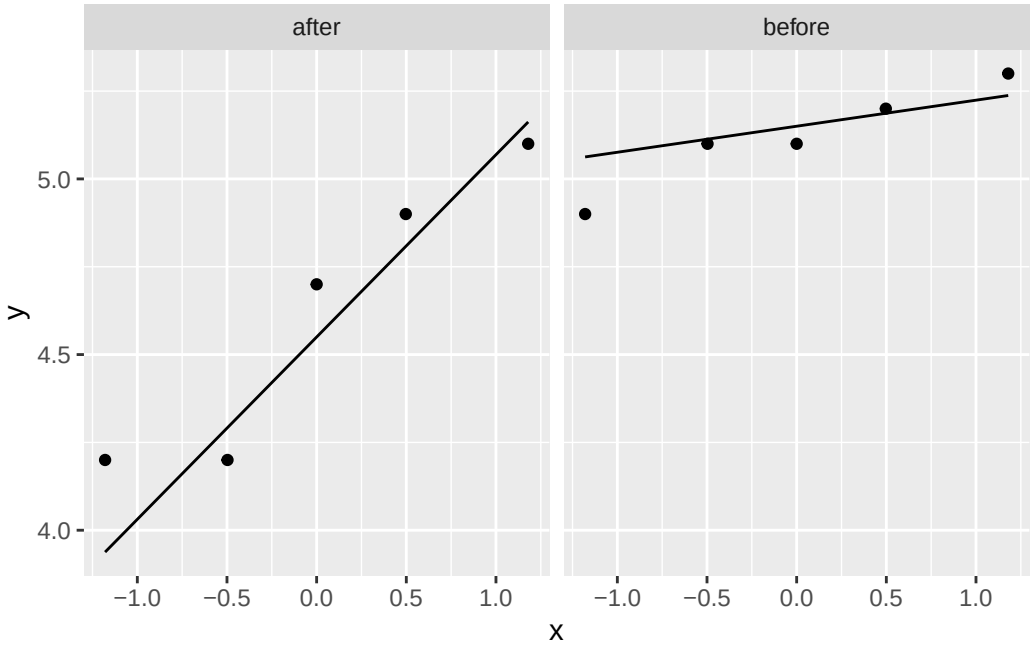
- what the null hypothesis is
- what the alternative hypothesis is
- what the P-value is
- whether you reject the null, and thus what your conclusion is in terms of mean number of deaths.

Minus a half point for each one of those missing, down to a minimum of a half point if you have anything correct. I'm prepared to accept missing hypotheses if it is clear from your later work that you know what they are, but you *need* to state the P-value and give a conclusion about mean number of deaths.

Extra:

To assess the normality, we might look at normal quantile plots:

```
ggplot(illinois, aes(sample = deaths)) + stat_qq() +
  stat_qq_line() + facet_wrap(~ when)
```

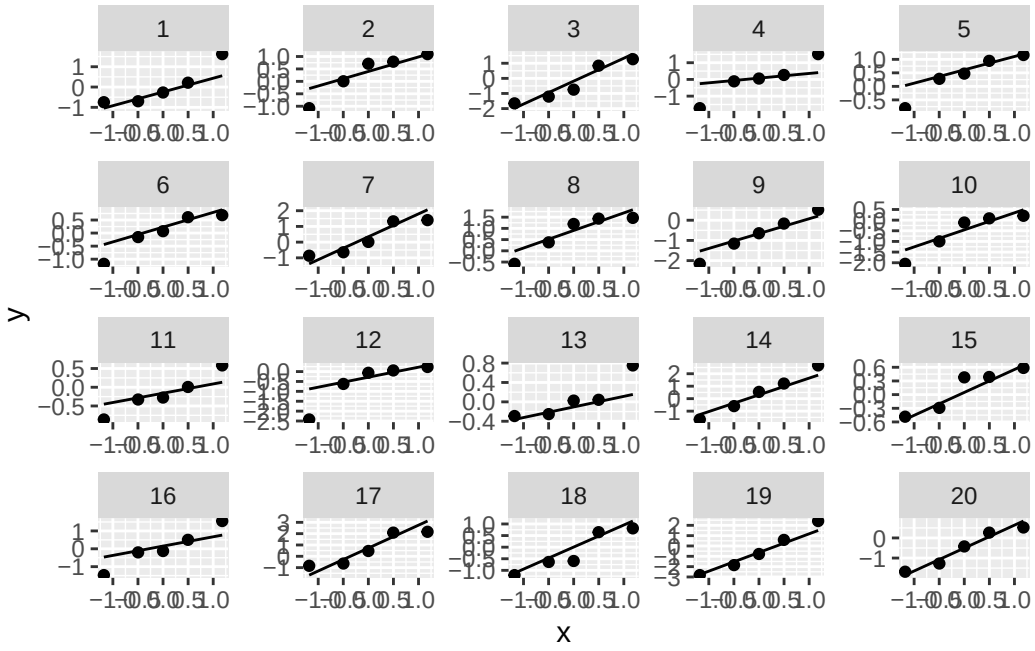


I don't know what you think of those. My immediate take is that for such small samples, they are not too bad. Having said that, the lowest value in the `before` sample is rather close to being an outlier (it is quite a bit further from the line than the others), and the values in the `after` sample are not especially close to the line (any of them, really), but they don't deviate from the line in a systematic way that suggests something like skewness.

From the graph in Figure 3, there are actually something like time trends before and (especially) after 1966, so that it doesn't really make much sense to talk about "a" mean up to 1966 and "a" mean after, which is what our t-test is really doing.

The justification for the t-test here is that samples of size 5 from normal distributions in general may not look very normal, or at least no more normal than the data we have. Let's generate a bunch of them and make normal quantile plots from them:

```
tibble(sim = 1:20) %>%
  rowwise() %>%
  mutate(my_sample = list(rnorm(5))) %>%
  unnest(my_sample) %>%
  ggplot(aes(sample = my_sample)) + stat_qq() + stat_qq_line() +
    facet_wrap(~ sim, scales = "free")
```



This is what I mean by saying that samples of this size actually from a normal distribution don't really look any more normal than ours do. With samples this small, it's hard to tell whether the populations are normal or not. Specifically, these *actually are* normal data, but you can identify problems in several of them:

- outliers: 1, 2, 5, 6, 12, 13
- skewness: 1, 2, 10, 12
- long tails: 4, 11, 16

so that the problems you see here are actually "type II errors": this is just what a normal quantile plot of a small sample of data from a normal distribution looks like. That is to say, with a small sample, you really cannot tell whether it comes from a normal distribution or not.

In our case, the t-test and the Mood median test actually give about the same P-value, which suggests that it doesn't matter much which test we do.

Here is the Mood median test:

```
median_test(illinois, deaths, when)
```

```
$grand_median
[1] 5

$table
        above
group    above below
  after      1     4
  before     4     1

$test
       what      value
1 statistic 3.60000000
2        df 1.00000000
3   P-value 0.05777957
```

Most of the "after" values are below the grand median of 5, and most of the "before" values are above, so we are on the correct side, and we can halve the given two-sided P-value:

```
0.0578 / 2
```

```
[1] 0.0289
```

to get something very close to what came out of the t-test (0.0248). This indicates that there are no major problems with either test, and that we should therefore use the t-test because it uses the data more efficiently.

2. Figure 5 shows some code to estimate the power of a test, along with the output from the code. Based on this Figure, answer the following questions.

   (a) [2] What null hypothesis is being tested? Here and elsewhere in this question, if you use any symbols, you should define what they mean.

   My answer:

   The null hypothesis is that the population mean is 100. (The value of `mu` on the line with `t.test` in it.)

   If your answer is something like $H_0 : \mu = 100$, you have used a symbol ("mu"), so you need to say in addition "where μ is the population mean". Only one point if you use μ or "mu" without saying what it means. You know that the test is of a population mean because it is a t-test.

   (b) [2] What is the alternative hypothesis of the test?

   My answer:

   The alternative hypothesis is that the population mean is less than 100. (Say what it is that is less than what value.)

   If you lost a point in the previous part for not defining μ, then saying $H_a : \mu < 100$ is enough for the two points here, because you haven't made an additional mistake.

   (c) [3] What population is being sampled from, as far as you can tell from the Figure?

   My answer:

   A normal population (or normal distribution) with mean 90 and SD 25. (One point for each of those three things.) Show that you know what `rnorm` does and how it works: its three inputs are the sample size, the population mean, and the population SD, in that order.

   Minus a half point for talking about *sample* mean. The whole point here is that we are sampling from a *population* whose mean is 90, and we want to see whether the sample mean (which we don't know until we take the sample) is far enough away from 100 to reject a null hypothesis that the population mean is 100 (which we actually know to be false).

   (d) [1] What size of sample is being taken?

   My answer:

   40 (from the line defining `my_sample`, inside the `rnorm`). Just the number will do.

(e) [3] Explain specifically what the output *means*, for the benefit of a manager who does not know what the code does. (The manager does not want to know what the code *does*; they only care about what the output tells them.)

My answer:

The power of a t-test to reject a mean of 100 when sampling from a normal population with mean 90 and SD using a sample size of 40 is (estimated to be) 0.79.

"The power is 0.79" is not enough. The manager needs to know the power of what null against what alternative with what true distribution. That's why I said "specifically", and why the question includes the words after the comma, and the last sentence in parentheses.

One point each for:

- the power is 0.79
- something relevant about the true population (eg. mean is 90)
- testing (and hoping to reject) that the population mean is 100.

There might be a half point for saying something relevant that falls outside this (for example, something that shows that you know what power *is*). Thus, you may find that you have both positive and negative points on this part. (Crowdmark does the deductions from the maximum of 3 first, or at least I have made sure that it did here.)

(f) [2] If I instead used a sample size of 60 (as opposed to the value used in the code in Figure 5), what can you say about what the power would be? Explain briefly.

My answer:

The power would be higher than 0.79, because the sample size is larger than the one used in the Figure, and therefore the power would also be larger.

More than that we cannot say, because we would need to actually run the code to find out how much bigger the power would be.

If you previously came to the conclusion that the sample size used in the simulation was greater than 60 (such as deciding that it was 100), the right answer here is that the power is *less* than 0.79, for the same reason as above.

(g) [3] There is a way to *calculate* the power in this situation, rather than estimating it by simulation. Why is that, and what code would do the calculation?

My answer:

We are sampling from a *normal* distribution, which is the assumption made by `power.t.test`. If we had been sampling from any other distribution, we would not have been able to do that. (Say why the population being normal is important, not just that the population is normal.) One point for this, a half point only if "the population is normal" is your complete answer without saying why that matters.

Hence, work out what the inputs to `power.t.test` need to be for a null mean of 100, a true mean of 90, a true SD of 25, a sample size of 40, and a one-sided one-sample t-test:

```
power.t.test(n = 40, delta = 100-90, sd = 25,
             type = "one.sample",
             alternative = "one.sided")
```

```

     One-sample t test power calculation

              n = 40
          delta = 10
             sd = 25
      sig.level = 0.05
          power = 0.7997378
    alternative = one.sided
```

The values in `delta` can be the other way around. Note that `type` specifies the number of *samples*, and `alternative` for this is either `one.sided` or `two.sided`: which one side does not matter. (`power.t.test` works out which one side you mean from the value of `delta`, so that in fact if your `delta` is the other way around, it will use the other tail, with everything being

flipped around so that the answer is the same.)

Two points for that code. Minus a half point per error, because there is a lot to get right.

The value shown in Figure 5 is close to this (but not exactly equal because that is based on a simulation).

3. A scientist obtained 24 determinations of the amount of copper in whole wheat flour, in parts per million. It is desired to assess whether the "average" (mean or median, as appropriate) amount of copper is 4 parts per million, or whether it is different from that.

   (a) [2] A normal quantile plot of the data is shown in Figure 6. The scientist decided to run a sign test rather than a t-test. How does this Figure, along with anything else you have learned so far about the data, support the scientist's decision? Explain briefly.

My answer:

The usual two things, a point each:

- the distribution is not close to normal because of the two outliers at the upper end, one of which is very much higher than the other values (is an extreme outlier). Mention at least the extreme upper outlier. "Outlier" is better than "skewed" because for the latter, the lowest few values would be above the line and there would be a clear curve shape, which there is not here.
- the sample size of 24 is probably not large enough to overcome the effects of the outliers, particularly the extreme high one.

You learned about the sample size in the description of the data in the question, so if it is relevant, and it is, you should be prepared to discuss it. Alternatively, the discussion here is whether the values are sufficiently close to being normal, and that discussion always involves sample size, so you should be looking around to see what the sample size is. At worst, you can count the dots on the normal quantile plot.

Extra: if this were your data, it would be up to you to decide whether to run a t-test or a sign test. In this problem, though, I have made it easier for you by saying which conclusion you need to come to, so the issue for you is to figure out how to get there. You might disagree with running the sign test here, but that is not the point; here, the issue is how the scientist justified their decision, so you have to put yourself in their shoes, and ask what argument *they* would make to come to the decision they did.

   (b) [2] What code would run a suitable sign test here? The data values are in a column called `copper` in a dataframe called `flour`.

My answer:

This: dataframe, column, null median:

```
sign_test(flour, copper, 4)
```

```
$above_below
below above
   22     2

$p_values
  alternative     p_value
1       lower 0.000017941
2       upper 0.999998510
3   two-sided 0.000035882
```

The output (see Figure 7) contains P-values for two-sided and both one-sided tests, so it is an error to give any kind of `alternative` here. Also, `smmr` is already loaded (see Figure 1), so you don't need a `library(smmr)` here, though it's not an error if you have it.

One point only if anything of any importance is wrong. (1.5 only if the only error is a trivial one, in the grader's estimation.) 0.5 if you somehow make more than one error but something is correct.

(c) [3] The output for the sign test for which you gave code in the previous part is shown in Figure 7. What do you conclude, in the context of the data?

My answer:

We are testing the null hypothesis that the population median copper content is 4 (parts per million), against the alternative that the median is not equal to 4. The (two-sided) P-value is 0.00004, very small, so we reject the null hypothesis and conclude that the population median copper content is different from 4 (parts per million). Or say that we have evidence that the population median copper content differs from 4.

Your conclusion *must* get to a statement about population median copper content (which is what "in the context of the data" means). You also need to write down the P-value you used, to make it clear that you used the correct one. (It is a good habit to get into to write down the P-value every time you do a test.) You help yourself also by writing down the null and alternative hypotheses, and then it should be straightforward to look at your P-value and decide which of the hypotheses you will go for. (Writing down your process is also a very good way to improve your chances of part marks if you go wrong, because it might then become clear than you got about half of it right and you can get about half of the marks. Writing down only "don't reject null" here is wrong and a very fast zero.)

Marking guide:

- 3 points for getting to a correct statement about the population median copper content
- 2 points for getting to "reject the null" or similar, without saying anything about copper content, but giving the appropriate P-value
- 2 points for a statement about copper content without saying which P-value you used, or for using the wrong one
- 1 point for getting to "reject the null" without giving the P-value you used or saying anything about copper content

less half points for other small errors. Note that "reject the null" by itself, though correct (as far as it goes), is only 1 point out of 3 because you have only done 1/3 of the work required.

(d) [2] How could you have guessed, without looking at the P-values in Figure 7, whether the P-value you wanted would be small or not? Explain briefly.

My answer:

Almost all the data values (22 out of 24) were below the null median of 4, with only two being above 4. With this much imbalance between how many values were above and below 4, we would have expected to see a small P-value. Alternatively, if the null hypothesis was correct, we would expect to see about half the values above 4 and half of them below 4. We very much did *not* see that, therefore we would expect to be rejecting the null hypothesis and would expect to get a small P-value.

Extra: because the extreme outlier is at the top end, the mean would be much *larger* than the median (pulled up by the outlier), and therefore the evidence that the *mean* differs from 4

would be much weaker:

```
with(flour, t.test(copper, mu = 4))
```

```

    One Sample t-test

data:  copper
t = 0.25933, df = 23, p-value = 0.7977
alternative hypothesis: true mean is not equal to 4
95 percent confidence interval:
 2.043523 6.517311
sample estimates:
mean of x
 4.280417
```

The outlier pulls the mean up so far that the sample mean is now *greater* than 4 (the sample median is much less), and also inflates the SD so much that the confidence interval for the mean goes all the way up to 6.5. As a consequence, the population mean is not at all significantly different from 4 (P-value 0.7977).

When two competing tests lead to such different conclusions, something must have gone wrong with one of them. In this case, it's the t-test that is not to be trusted because of that extreme outlier. The sign test is not at all bothered by the outlier; it just counts as one of the two values above the hypothesized median, and it doesn't matter how extreme the outlier is. (This is more support for the scientist using the sign test rather than the t-test.)

(e) [2] What code would obtain a confidence interval for the population median copper content?

My answer:

Dataframe and column (same as the code for the sign test), so this is rather a giveaway:

```
ci_median(flour, copper)
```

```
[1] 2.804609 3.696309
```

Extra: if you read the Extra above about the t-test, you see that this interval for the median is much shorter (and also much more sensible) than the CI for the mean.

(f) [3] The bootstrap sampling distribution of the sample mean is shown in Figure 8. How does this support the scientist's decision to use a sign test, and why did this distribution come out with the shape it did?

My answer:

This distribution is very much *not* normal (the first point), and therefore we should not be using a t-test and the decision to use the sign test was correct (the second point). The sample size is *not* relevant here, and it is an error to mention it, because the sampling distribution of the sample mean already accounts for the sample size (the bootstrap samples are of the same size as the original sample).

The distribution has an odd shape, with several (apparently three) peaks. The extremely high outlier will have a large influence on the sample mean, and this particular shape comes about because of *how many times* that high outlier appears in the bootstrap sample (it being taken with replacement, so any of the original sample values can appear any number of times). Something close enough to this for the third point. (I was willing to give the third point if you said that the mean of the bootstrap sample will depend on *whether or not* the outlier was in it, which is not quite right, but is close enough to the right idea.)

Specifically, the high outlier is about 30, and the other values are about 3, as you can read from Figure 6, so roughly speaking, if the high outlier does not appear at all in the bootstrap sample, the sample mean will be about 3 (the first peak), if it appears once the sample mean will be about

```
(23*3 + 1*30)/24
```

```
[1] 4.125
```

(the second peak), if it appears twice, the sample mean will be about

```
(22*3 + 2*30)/24
```

```
[1] 5.25
```

(the third peak), and so on. The high outlier is unlikely to appear much more than that in a bootstrap sample, so there are no more visible peaks. (It looks as if the sample with mean 8 had the outlier in it five times: see the Extra below.)

The third point for convincing me that you know why those three peaks came where they did. This is not something you've seen before; I'm expecting you to be able to reason it out from the information at your disposal.

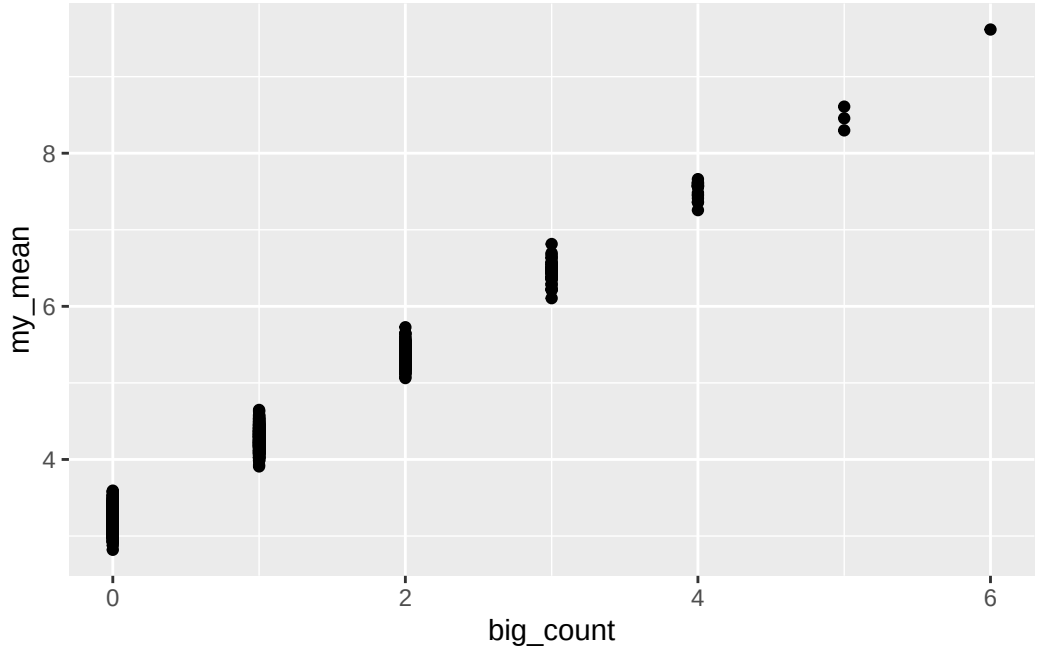
Extra: my original notes for this problem said the following:

That looks more like a hedgehog than a normal distribution! The reason is that the bootstrapped sample mean is heavily dependent on that one very large value; in particular, the mean of a bootstrap sample from these data is likely to depend mostly on how many times that one very large value gets resampled: the very tall bars are zero, once, twice, three, four, and

five times (at the end). It is possible to resample that very large value several times, but, as you see, this becomes progressively less likely. The pattern would be expected to show up almost no matter what number of bins I had chosen: that is, it is a real feature of the distribution and not just an artefact of my having chosen 10 bins.

To confirm that, I can tweak the simulation to count the number of times that large value was observed as well as the mean:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(flour$copper, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  mutate(big_count = sum(my_sample > 20)) %>%
  ggplot(aes(x = big_count, y = my_mean)) + geom_point()
```



and then you can see just how much the sample mean was dependent on the number of times that very large value was observed. Indeed, if you know the mean of the bootstrap sample, you can tell exactly how many times that big value was in the sample.

According to what I was able to find out about this dataset, the very large value was apparently checked and found to be correct, even though it seems to be about ten times too large (and misplacing a decimal point *is* the kind of thing that happens with real-world data). I guess it is possible to get a contaminated sample that contains ten times as much copper as the others.

As a technical detail, `my_sample > 20` is either `TRUE` or `FALSE` for each observation. If you add these up, `TRUE` counts as 1 and `FALSE` as zero, so you end up with the number of times an observation (in the bootstrap sample) was greater than 20. This is a handy trick for counting how many times something is true in a sample.

4. 39 chickens were randomly allocated to one of three different "rations" (feeds), so that 13 chickens received each ration. The weight gain (in kg) of each chicken over a certain time period was recorded. We are interested in whether the different rations are associated with different average (mean or median) weight gains, and, if so, which rations cause the largest weight gains. Some of the data is shown in Figure 9, and a boxplot of the data is shown in Figure 10.

   (a) [3] Three possible analyses are shown in Figures 11, 12, and 13. Which of these analyses do you think is the most suitable for these data? Explain briefly.

My answer:

The three analyses are a regular ANOVA, a Welch ANOVA, and a three-group Mood's median test. To decide which of these is best, look at the boxplots. I think the ones for Rations 2 and 3 are close to symmetric, but the one for Ration 1 is slightly skewed to the right. The sample sizes are 13 in each group, and, bearing in mind what we have seen in the course, I think this is large enough to overcome the kind of skewness we see in Ration 1.

There is something funny happening with ration 3: the first quartile and the median are the same. This usually means that there are a lot of values all the same; Figure 9 shows that the values are all rounded to whole numbers, so it is entirely possible that a lot of the values were 6. The *tails*, however, indicate a symmetric distribution apart from that, and it's the tails that we need to be concerned about. Funny stuff in the middle of the distribution will be taken care of by the Central Limit Theorem even for modest sample sizes. See the Extra to this question.

So I am happy with the normality. What about equal spreads? The box for Ration 3 is definitely not as tall as for the other Rations, so I think equal spreads fails, and we should therefore do a Welch ANOVA.

If you would prefer to do a Mood's Median Test, you need to justify that we don't have sufficient normality given the sample size, and so you would have to describe the skewness in Ration 1 as being "severe" or something like that, too much for a sample size of 13 to overcome. (13 is not a large sample size by any means, but the Central Limit Theorem works pretty well even for modest sample sizes when what you have to deal with is skewness like this. Outliers, particularly far-out ones, are another story, as we saw for the copper data in an earlier question.)

Follow the process through:

- assess sufficient normality in each of the groups (for each of the rations) by looking at the boxplots and the sample sizes (two points)
- if you think normality is ok enough, assess equality of spreads (part of the above, if needed)
- based on these one or two steps, say which kind of analysis is suitable for these data: if normality is no good, Mood's median test; if normality and equal spreads are both OK, ANOVA; if normality is OK but equal spreads is not, Welch ANOVA. (The third point.)

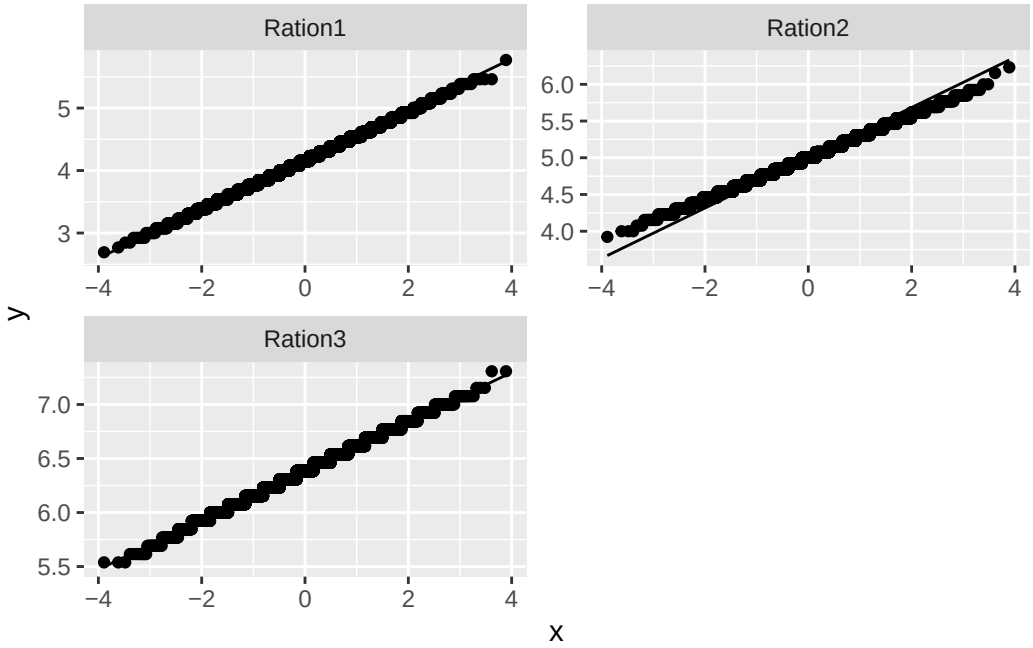
I think the Welch ANOVA is best, but you can also defend Mood's median test if you can make the case that Ration 1 is too skewed for a sample of size 13 to overcome. See the Extra below for what the "right" answer is.

Extra: the obligatory bootstrap sampling distributions of the sample means, for the three groups done all at once:

```
Rows: 39 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): ration
dbl (1): weight_gain

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.
```

```
chickens %>% nest_by(ration) %>%
  mutate(sim = list(1:10000)) %>%
  unnest(sim) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(data$weight_gain, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(sample = my_mean)) + stat_qq() +
  stat_qq_line() + facet_wrap(~ ration, ncol = 2, scales = "free")
```



These are actually *very* normal, even for Ration 1. The most troublesome one is actually Ration 2, which has something like short tails, except that the two largest observations are about as

expected (and in any case, short tails are not a problem when working with means). So some flavour of ANOVA is in actual fact fine, and the Welch one is best because of the smaller spread with Ration 3 seen in the boxplot.

The weight gain values are all recorded as whole numbers, so that the bootstrapped sample mean will always be some number of thirteenths, and there are therefore a smallish number of possible values for the mean of a bootstrap sample here. That explains the stair-step appearance of the normal quantile plot for Ration 3, and, if you look carefully, for the other Rations as well. Because Ration 3 had all those data values equal to 6, there are not as many possible bootstrap sample means here as there are for the other Rations, and so the stair-steps are easier to see here.

(b) [2] What do you conclude from your chosen analysis, as far as your chosen Figure allows you to conclude?

My answer:

The conclusion is actually the same in all three analyses: the P-value is (very) small, so we reject the null hypothesis, which states that all three Rations have the same mean (median) weight gain. We therefore conclude that not all the rations are the same in terms of mean (median) weight gain, or there are differences among the means (medians).

You should give the P-value for your chosen analysis, to make it clear that you have been consistent with what you said in the previous part:

- for the regular ANOVA, $9 \times 10^{-5} = 0.00009$
- for the Welch ANOVA, 0.00015,
- for Mood's median test, $8.4 \times 10^{-4} = 0.00084$.

Half a point for quoting the P-value, and the rest of it for getting to a good conclusion about weight gains for the different rations.

Remember that saying "the means are different" is wrong because all you know is that the means are not all the same. They could be all different (implied by "the means are different") but, for example, two of them could be equal and different from the other one. Be careful with your English. "The means are different" is $0.5 + 0.5 = 1$, assuming that you cited the right P-value. The grader cannot determine what you *meant*; they can only assess what you actually wrote.

This is as far as you go in this part (the reason for the bit after the comma in my question). Deciding which rations differ from which is the job of Tukey (or Games-Howell or pairwise median tests), which we haven't looked at yet. So it is an error to say more than "the means are not all equal" (or equivalent) here.

Extra: if you looked at all three P-values (as I just did), you'll see that they are all about the same (and thus lead to the same conclusion), but the P-value for Mood's median test is a little bigger than the others. This is a common kind of result, and says that any of the tests would be fine, but the appropriate ANOVA (Welch) is more powerful because it uses the actual

data values, which the Mood test does not (just counts above and below something, without thinking about *how far* above or below). This is another way to help convince yourself that one of the ANOVAs would be fine. (If the skewness had been more severe, the P-values would have been more different.)

(c) [3] Some followup analyses are shown in Figures 14, 15, and 16. Which one is the most appropriate followup for the analysis you used in the previous part, and what do you conclude from it, in the context of the data? If none of those followup analyses are appropriate, explain briefly why.

My answer:

Depending on what you thought was the appropriate analysis:

- if you used the regular ANOVA, use Figure 15
- if you used the Welch ANOVA, use Figure 16
- if you used Mood's median test, use Figure 14.

Note that the followups are in a different order than the first-stage analyses.

One point for saying which followup you did. Expect the grader to check back to see that it is consistent with the first-stage analysis you did, whatever that was. The one point here is for being consistent with yourself, whatever you said before.

The conclusion is actually the same, whichever followup you use: Ration 3 is significantly different from both of the other two Rations, which are not significantly different from each other. That's enough for the other two points. (Discussion of which Ration produces the largest weight gain is coming in the next part, but there is no harm in saying here that Ration 3 is actually *better* than the other two, in the sense of causing a significantly larger weight gain.) For this one, I'm not requiring that you give all three P-values, though there is certainly no problem if you do.

If, for some reason, you had concluded that were *no* significant differences in weight gain among the three rations, you would need to say here that there was no justification for running any kind of followup analysis (because there are, in that case, no differences between rations to find).

Extra: you might say that since all three methods produce basically identical results, it doesn't matter which one we use. This is true to an extent, but compare the P-values: the ones for the Mood's median test and the pairwise median tests are noticeably larger than for the other tests. Even though the conclusions don't differ, the evidence is less strong for the Mood's median test than for the other tests. This is actually quite common: you end up with similar conclusions, but the test with the fewest assumptions (Mood's median test here) has the highest P-values, and this is a sign that one of the ANOVAs is actually the best way to go. Remember that choosing a test on the basis of having the smallest P-value is scientifically dishonest (and thus is a *bad* reason for preferring a regular ANOVA here), but the Mood's median test is likely to be least powerful (and have a larger P-value) when one of the other tests is appropriate. This is because the two ANOVAs use the actual data values, but Mood's median test uses counts of how many data values are above or below something, without using the data values themselves.

If the tests had disagreed substantially (as happened with the copper data), it likely would mean that there was a problem with one of the assumptions, and thus that Mood's median test would be the way to go. But that doesn't seem to have happened here.

(d) [2] Based on what you have found so far, what is your recommendation for the ration or rations to recommend for future use in feeding chickens, given what we are interested in? Justify your choice or choices briefly.

My answer:

There are two things to say (one point each):

- Ration 3 has the largest median (from the boxplot) and mean (from the Tukey output) weight gain. Noting one of those is enough. Strictly, you should note the appropriate one for the analysis you did (the mean if you did an ANOVA, the median if you used Mood's median test.)
- Ration 3 gives *significantly* larger weight gain than the other two Rations.

Therefore we can recommend Ration 3 for future use, if our aim is to maximize weight gain (as was stated in the question).

Extra: if we had been trying to *minimize* weight gain (for whatever reason), our recommendation would have been *either* Ration 1 *or* Ration 2, because they do not differ significantly, and therefore sometimes Ration 1 will come out lowest, and sometimes Ration 2 will. The significant difference between Ration 3 and the other Rations means that you can expect Ration 3 to come out consistently the highest (as Fisher said, "a well-designed experiment will rarely fail to yield significance").

5. This question is about tidying data.

   (a) [2] Dataframe `d1` in Figure 17 contains some data from a two-sample experiment where six individuals were randomly assigned to one of two treatments A and B. What code would transform dataframe `d1` into the dataframe `d2`, shown in Figure 18?

   My answer:

   This is a standard pivot-longer. The new columns need to have the names `treatment` (for the treatment names) and `y` (for the response values), so use these in `names_to` and `values_to` (respectively):

   ```
   d1 %>% pivot_longer(-r, names_to = "treatment", values_to = "y")
   ```

   ```
   # A tibble: 6 x 3
         r treatment     y
     <dbl> <chr>     <dbl>
   1     1 A            10
   2     1 B            21
   3     2 A            12
   4     2 B            19
   5     3 A            13
   6     3 B            22
   ```

   (b) [2] What statistical purpose might you have for wanting to transform dataframe `d1` into `d2`, as in the previous part?

   My answer:

   The data need to be in "long" format, with all six values of `y` in one column, as in `d2`, in order to draw a graph (a boxplot) or to run a two-sample t-test. Either of those, or some other relevant technique that needs "long" data, is good. ("For analysis" is only a one-point answer, because by naming a specific technique, you make it clear that a specific layout of the data is needed. It might be that some other form of analysis needs the data in the wider format.)

   (c) [2] What code would transform dataframe `d2`, shown in Figure 18, into dataframe `d1`, shown in Figure 17?

   My answer:

   The opposite of the earlier part, so a standard pivot-wider:

   ```
   d2 %>% pivot_wider(names_from = treatment, values_from = y)
   ```

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

```
      r     A     B
  <dbl> <dbl> <dbl>
1     1    10    21
2     2    12    19
3     3    13    22
```

This works properly because column `r` properly indicates which row each value of `y` should go to. (You don't need to say this here, because I have another part coming up later that assesses this issue, but if you want to check, the result of the pivot-wider will have rows 1 through 3 in `r`, and columns called `A` and `B` instead of `y`, with one A and one B in each row.)

(d) [3] The data in `d3` (shown in Figure 19) are from an experiment on millipedes. The concentration of a certain amino acid was measured for males and females of each of two species, labelled S1 and S2. Tidy the data to have one column of amino acid values, called `amino`, with other columns labelling the sex and species of each millipede. Do this in the most efficient way.

My answer:

This is the first variation of `pivot_longer` where each column encodes two things, so the following is best, sex and species being separated by the letter `x`. Split the lines of code where you see fit if they get too long, anywhere that is obviously incomplete (for these, with a comma on the end of the line):

```
d3 %>% pivot_longer(everything(),
  names_to = c("sex", "species"),
  names_sep = "x",
  values_to = "amino"
)
```

```
# A tibble: 8 x 3
  sex    species amino
  <chr>  <chr>   <dbl>
1 male   S1       21.5
2 male   S2       14.5
3 female S1       14.8
4 female S2       12.1
5 male   S1       19.6
6 male   S2       17.4
7 female S1       15.6
8 female S2       11.4
```

This is the best way to specify the columns to pivot-longer here. The following also works, and is also full marks (as is anything else you can think of that says what the column names have in common):

```
d3 %>% pivot_longer(contains("x"),
  names_to = c("sex", "species"),
  names_sep = "x",
  values_to = "amino"
)
```

```
# A tibble: 8 x 3
  sex    species amino
  <chr>  <chr>   <dbl>
1 male   S1       21.5
2 male   S2       14.5
3 female S1       14.8
4 female S2       12.1
5 male   S1       19.6
6 male   S2       17.4
7 female S1       15.6
8 female S2       11.4
```

If you specify the columns as `malexS1:femalexS2`, that is harder work than necessary, and will cost you a half point.

The other way to do it is an ordinary pivot-longer followed by a separate (with, as ever, line breaks at appropriate places):

```
d3 %>% pivot_longer(everything(),
                    names_to = "sex_species", values_to = "amino") %>%
  separate_wider_delim(sex_species, "x", names = c("sex", "species"))
```

```
# A tibble: 8 x 3
  sex    species amino
  <chr>  <chr>   <dbl>
1 male   S1       21.5
2 male   S2       14.5
3 female S1       14.8
4 female S2       12.1
5 male   S1       19.6
6 male   S2       17.4
7 female S1       15.6
8 female S2       11.4
```

This works, but is an extra step (that you don't need if you understand `pivot_longer` well enough), so only two points. Use any name you like for the intermediate column (that I called `sex_species`).

(e) [3] We want to rearrange the data in `d3` (Figure 19) to have separate columns for the amino acid measurements for males and females (that is, we want columns *named* `male` and `female`, but a column called `species` that *contains* the values `S1` and `S2`). What code will carry out this rearrangement, as directly as possible?

My answer:

This uses the special "column name" `.value` (that isn't really a column name). This goes with the thing we were earlier calling `sex`, which is the *first* part of the column name in `d3`, because we want the males and females (the different values in `sex`) to go into their own columns. You do not need a `values_to` for this one, because you do not need a name for the column(s) that the values are going into (their names are going to be `male` and `female` that come from the column names in `d3`):

```
d3 %>% pivot_longer(
  everything(),
  names_to = c(".value", "species"),
  names_sep = "x"
)
```

```
# A tibble: 4 x 3
  species  male female
  <chr>   <dbl>  <dbl>
1 S1       21.5   14.8
2 S2       14.5   12.1
3 S1       19.6   15.6
4 S2       17.4   11.4
```

This is best.

```
d3 %>% pivot_longer(
  everything(),
  names_to = c(".value", "species"),
  names_sep = "x",
  values_to = "y"
)
```

```
# A tibble: 4 x 3
  species  male female
  <chr>   <dbl>  <dbl>
1 S1       21.5   14.8
2 S2       14.5   12.1
3 S1       19.6   15.6
4 S2       17.4   11.4
```

It is actually *not* an error to specify a `values_to`, but it doesn't get used for anything, so it

betrays a little less understanding of what is actually going on here, but I won't penalize you for that.

If you don't think of using `.value`, then find a way that works, such as the pivot-longer you did before, plus a pivot-wider of one of the things you just pivoted-longer:

```
d3 %>% pivot_longer(contains("x"),
  names_to = c("sex", "species"),
  names_sep = "x",
  values_to = "amino"
) %>%
  pivot_wider(names_from = sex, values_from = amino)
```

```
Warning: Values from `amino` are not uniquely identified; output will contain list-cols.
* Use `values_fn = list` to suppress this warning.
* Use `values_fn = {summary_fun}` to summarise duplicates.
* Use the following dplyr code to identify duplicates.
  {data} %>%
  dplyr::group_by(species, sex) %>%
  dplyr::summarise(n = dplyr::n(), .groups = "drop") %>%
  dplyr::filter(n > 1L)
```

```
# A tibble: 2 x 3
  species male      female
  <chr>   <list>    <list>
1 S1      <dbl [2]> <dbl [2]>
2 S2      <dbl [2]> <dbl [2]>
```

This actually doesn't work, because the pivot-wider doesn't have the right rows to put things in, and two values ended up in each cell. (This is always a potential issue with `pivot_wider`, and you therefore need to be thinking about it.) 1.5 for getting to here (if you did this, you probably got this score and no comments, because there was no point in writing the above out again), and a second for getting the values out:

```
d3 %>% pivot_longer(contains("x"),
  names_to = c("sex", "species"),
  names_sep = "x",
  values_to = "amino"
) %>%
  pivot_wider(names_from = sex, values_from = amino) %>%
  unnest(c(male, female))
```

```
Warning: Values from `amino` are not uniquely identified; output will contain list-cols.
* Use `values_fn = list` to suppress this warning.
* Use `values_fn = {summary_fun}` to summarise duplicates.
* Use the following dplyr code to identify duplicates.
```

```
  {data} %>%
  dplyr::group_by(species, sex) %>%
  dplyr::summarise(n = dplyr::n(), .groups = "drop") %>%
  dplyr::filter(n > 1L)
```

```
# A tibble: 4 x 3
  species  male female
  <chr>   <dbl>  <dbl>
1 S1       21.5   14.8
2 S1       19.6   15.6
3 S2       14.5   12.1
4 S2       17.4   11.4
```

which is admittedly a lot of work for two points, and shows the usefulness of knowing what `.value` does. I'm hoping for the best students to realize, even on an exam, that a `pivot_wider` by itself is not going to work: the result of the pivot-longer has three columns, `sex`, `species`, and `amino`. Only `species` is not used in the pivot-wider, and there are only two species (S1 and S2), so after pivoting-wider, there will only be two rows, and with the two columns `male` and `female`, there are only four cells for eight observations. Hence, two of them appearing in each cell. This stuff is tricky.

(f) [3] A dataframe `d4` is shown in Figure 20, with some code below it. What output will the code produces?

My answer:

```
d4 %>% pivot_wider(names_from = g, values_from = y)
```

```
# A tibble: 3 x 4
      r     A     B     C
  <dbl> <dbl> <dbl> <dbl>
1     1    10    NA    NA
2     2    NA    12    14
3     3    NA    17    NA
```

The value in `g` shows which *column* the value of `y` will go in. The value in `r`, which was not named in the `pivot_wider`, will show which *row* that value of `y` will go in. Suggestion: you know the columns will be A, B, C, and the rows will be 1, 2, 3 in a column called `r`, so draw an empty dataframe with those rows and columns, fill in the values of `y` you have in the appropriate places, and then fill in missings everywhere else. (This is one of those "not enough data" cases: there are four values of `y` and $3 \times 3 = 9$ cells they might go in. If it had been a "too much data" case, you would have found yourself writing more than one value in the same row and column, and this would have resulted in a warning and list-columns if you had actually been running it in R yourself.)

Adding an explanation might help you get some partial credit if it is not clear where your

answer came from and it was different from mine.

6. In a study of cheddar cheese from the LaTrobe Valley of Victoria, Australia, samples of cheese were analyzed for their chemical composition and were subjected to taste tests. Overall taste scores were obtained by combining the scores from several tasters. Our aim is to see whether taste score (`taste`) is influenced by any of the three explanatory variables, which are as shown below, measured in suitable units:

   - `Acetic`: concentration of acetic acid
   - `H2S`: concentration of hydrogen sulfide
   - `Lactic`: concentration of lactic acid

   Some of the data are shown in Figure 21.

   (a) [3] Some scatterplots are shown in Figure 22. Describe what these plots tell you.

My answer:

Use the ideas of "form, direction, strength" to guide you in interpreting scatterplots, including these ones. "Form" means the shape, linear or curved. "Direction" means an upward or downward trend. "Strength" means whether you think the trend is strong, moderate or weak.

These plots show the relationships between the taste score and each of the three explanatory variables. I would describe all three of these relationships as moderate upward trends, approximately linear. Describe the trend with `Lactic` as "slightly curved" if you want, and say that there is a little fanning out on `Acetic` if you want.

Comments:

- There is a fair bit of scatter, so you should use an adjective like "moderate" to describe the strength of the trends. I think it is better than "weak", but the points are too far away from the trends to justify calling it "strong" in any of the cases. (I will also take something like "there are several outliers on each graph" as another way of saying that the strength of the relationship is moderate.)
- You can certainly save yourself some writing by saying that all three trends are basically the same.
- I would not worry about the "kink" in the `Acetic` graph. I think that's just a weirdness of how the data came out.
- You might say that the `Acetic` graph is an upward trend for a while, then levels off.
- My take is that there is too much scatter in the `Lactic` plot to justify calling this a real curve: a straight line will fit these data almost exactly as well as a curve will.

A minimal but full-credit answer is that all three trends are upward, (approximately) linear, and moderate in strength. Add comments beyond that if you wish, but be prepared to lose marks if you add things that the grader disagrees with, that don't seem to be justifiable conclusions from the plots. If the grader thinks your description is a reasonable deduction from the graphs, you are good.

"Has a normal distribution" is not of clear relevance here. What has to have a normal distribution is the *residuals*, which we look at later; if you want to judge that here, look at whether the points are further above than below their trends, or whether there are outliers. If that is what you are assessing, say so clearly; the usual assessment at this point is whether the trends go up or down, whether they are linear or curved, and how strong they are.

If you said *something* relevant, it was pretty difficult to get less than 1.5 here.

(b) [2] In the code at the top of Figure 22, why did I need to use `pivot_longer`? Explain briefly.

My answer:

In the original data, the three explanatory variables are in different columns, but to make this plot (getting all three scatterplots with one `ggplot`), I needed to arrange all the explanatory variables in *one* column, with a second column indicating which explanatory variable each value came from. This is exactly what `pivot_longer` does.

Or start with what `pivot_longer` does, and then say why this is exactly what you need here.

For full credit, get close enough to showing that you know what `pivot_longer` does and what layout of data is needed to make these scatterplots. "The data are wide so we make them longer" is only a one-point answer, because you have not explained *why* we need the data longer.

(c) [2] A regression model is shown in Figure 23. What regression model would you fit next, using $\alpha = 0.01$? Explain briefly. (If you would not fit any other regression models, explain briefly why not.)

My answer:

I would fit a regression containing only `H2S` and `Lactic`: that is to say, I would take out `Acetic` because it is nowhere near significant (P-value 0.94), much the highest non-significant P-value.

This was admittedly rather easy to guess, even including the explanation. The bit about $\alpha = 0.01$ was to make you think about whether you needed to remove `Lactic` as well.

If you say that you should remove both `Acetic` and `Lactic`, the way the question was worded, this plus a good reason was also enough to say (since I asked you which regression model you would fit next, and saying this, with reason, answers the question).

It would (in retrospect) have been better to ask something vaguer like "what would you do next?", and then you would *also* have needed to say that you should compare the models with all three x's and with only `H2S`, to see whether (at $\alpha = 0.01$) you were justified in removing both explanatory variables. The question as it is now is easier than I would have liked, but in the same way that we can only grade what you write, you can only answer the question that *I* wrote!

Remember that the P-values in the Figure only relate to the contribution of that one x-variable over and above the others in *that* regression, so that removing one explanatory variable and re-fitting is fine (to decide what, if anything, to remove next), but removing more than one requires you to test whether removing all the ones you did was justifiable.

I also accepted `step` as an answer, properly explained (what it does, or why what it does is what we want here), since it will start by removing `Acetic` and then see whether anything else can be removed (probably not, because `step` likes to keep explanatory variables on the edge of significance).

Extra: if you decided to remove both explanatory variables, the procedure would be something like this:

```
cheddar.2 <- lm(taste ~ H2S, data = cheddar)
anova(cheddar.2, cheddar.1)
```

```
# A tibble: 2 x 6
  Res.Df   RSS    Df `Sum of Sq`     F `Pr(>F)`
   <dbl> <dbl> <dbl>       <dbl> <dbl>    <dbl>
1     28 3286.    NA          NA  NA     NA
2     26 2668.     2         618.  3.01   0.0667
```

This is (perhaps surprisingly) not even significant at 0.05, let alone 0.01, so the conclusion from here is that taking out both `Acetic` and `Lactic` is fine: the smaller model with just `H2S` does not fit significantly worse than the bigger model with all three explanatory variables, so we go with the smaller, simpler model.

(d) [3] Figures 22 and 23 appear to say something contradictory (that is, what you learn from the two Figures is different in some way). What is it that is contradictory, and what would be a good reason why it happened? Explain briefly.

My answer:

There is an upward trend in the scatterplot between `Acetic` and `taste`, so there seems to be a relationship, but `Acetic` is far from significant in the regression. One point. (Or something equivalent, like saying that all three scatterplot trends are going up, but only `H2S` was significant.)

The rest of the discussion below applies to the remaining two points at stake.

Why did this happen? Think carefully about what the regression output says: `Acetic` does not *add* anything to the regression containing `H2S` and `Lactic`. So, whatever way these two (in combination) predict `taste`, it looks as if `Acetic` predicts `taste` in the same sort of way, and therefore `Acetic` has nothing to add over and above the other two. Two points for this.

You might also make the point that there is a sort-of upward relationship between `taste` and `Acetic` in the scatterplot, but it is weaker than the other two scatterplots. This is a sort-of answer: it doesn't really explain why `Acetic` is nowhere near significant in the regression. The scatterplot really suggests that `Acetic` would be maybe not significant, but at least close, absent any other insight. This is a one-point take.

We haven't talked about multicollinearity in this course (we do a little bit in D29), but you may have seen it in your second course, the one that talked about regression. What that says is that having the explanatory variables correlated with the *response* is a good thing (evidently), but having the explanatory variables correlated with *each other* is a *bad* thing, because it makes it difficult to tell which one(s) of them are really responsible for predicting the response variable. This shows up in P-values for explanatory variables that are larger (less significant) than you would expect, like `Acetic` is here. So another reasonable answer is `Acetic` is correlated with the other explanatory variables, which would also mean that it does not have anything to add to the prediction if the two other explanatory variables that it is correlated with are also in the regression. Two points for that, or something close enough to it.

The grading on this one was (unavoidably) subjective. I awarded two points for a convincing explanation, and a fraction of those points depending on how convincing you were in explaining what was going on.

Extra: the function `cor` gets correlations of all the variables in a dataframe:

```
cor(cheddar)
```

```
           taste    Acetic       H2S    Lactic
taste  1.0000000 0.5495393 0.7557523 0.7042362
Acetic 0.5495393 1.0000000 0.6179559 0.6037826
H2S    0.7557523 0.6179559 1.0000000 0.6448123
Lactic 0.7042362 0.6037826 0.6448123 1.0000000
```

All of the variables have at least a strongish positive correlation. I'm guessing that the high positive correlation between `Acetic` and the other two explanatory variables, along with `Acetic` having the *lowest* correlation with `taste`, are enough to say that once you have the other two, you don't need `Acetic` as well. (This is a way of getting more precisely at the idea that all three explanatory variables predict `taste` in about the same way, but `Acetic` does it worse than the others. This point also says something about how you can get away with removing everything but `H2S`: this predicts `taste` best out of all three, and once you have that in the regression, you don't really need either of the others because they are correlated with `H2S`).

(e) [3] Some residual plots are shown in Figures 24, 25, and 26. Describe any problems you see in these plots, or say that there are none, as appropriate. In either case, explain how you came to your conclusion.

My answer:

The easiest answer is to say that there are no problems: there is a random scatter of points in Figure 24, and in each of the three plots in Figure 26 (which is what we want), and the normal quantile plot has points mainly close to the line (indicating that the residuals have close to a normal distribution).

About the only other defensible answers I can see are:

- there is one large positive residual. This shows up most clearly on the normal quantile plot (all the points except for the top right one are close to the line), and *this same* residual also shows up on the other plots once you know to look for it: at the top middle of Figure 24 and the second and third plots in Figure 26, and at the top right of the first plot in the same Figure. (You can tell from each plot that the large residual is about 25, so it must be the same one, because there is only one residual that big.)
- there is a little fanning-out in the `Acetic` plot: that is, the residuals for larger values of `Acetic` go up and down further from zero than the ones for smaller values. According to what we learned in lecture, this would mean a transformation of `Acetic` would be worth trying, maybe something like taking log (to make the higher values not so much higher). A squared term would be good if we saw a curve here (which we don't).

Make sure you say (i) what you see, (ii) whether it's good or bad, and (iii) how you know.

Everything else is textbook-good, as far as I'm concerned. Those plots with the large residual in the top middle do not go up and down again because there is no indication in the rest of the points that this is what is happening.

Extra: one downside of giving you this problem on an exam is that this presentation doesn't quite represent how you would do it. You would probably remove `Acetic` first (especially with the problem laid out this way), and *then* look at residuals, which would mean that the residuals would be from a different model (though, in fact, I tried it both ways and the residuals are very much the same). I could have fitted the model without `Acetic` and then given you the residuals from that, only then you would have had the clue that a model without `Acetic` might have been the answer earlier. Likewise, the next part (the Box-Cox) maybe ought to have gone

first: after we've taken all the trouble to decide on a good model, *then* we're seeing whether it is of the right form (and maybe deciding that it was after all)? It can be difficult on an exam to keep the parts reasonably self-contained, so that if you miss one part, you don't get too messed up for the rest of it.

(f) [2] Some further analysis is shown in Figure 27. What do you conclude from this Figure? Explain briefly.

My answer:

This is a Box-Cox analysis, as indicated by the code above the plot. That is, it is looking for a possible transformation of `taste` to get a better (more linear) fit. There are two ways I think you could reasonably go:

- the CI for λ goes from about 0.3 to just below 1, and the only "round number" value in this interval is 0.5. Power 0.5 is square root, so that would suggest using the square root of `taste` in the regression instead of `taste` itself.
- $\lambda = 1$, which would mean no transformation, is only the tiniest distance outside the CI for λ, so using `taste` itself, as we did in the rest of the question, is pretty much a reasonable thing to do.

Show that you know (i) how to read the plot, (ii) what the value of λ ("lambda") tells you to do. Roughly speaking, a point for each of those.

7. In this course, we had 8 assignments, of which the best 6 are counted towards your grade. There is no function in R to calculate the mean of the largest 6 values out of 8 in a column, such as the column `x` in the dataframe `d` shown in Figure 28. In this question, we will write such a function. (Assume that the assignments shown in Figure 28 have all been graded out of 10 points.)

   (a) [3] What code would make a dataframe that contains the largest 6 out of these 8 scores?

My answer:

The easiest way is to use `slice_max`, thus:

```
d %>% slice_max(x, n = 6)
```

```
# A tibble: 6 x 1
      x
  <dbl>
1    10
2    10
3     9
4     8
5     8
6     7
```

If you don't remember that, the other way is to sort the whole column `x` and then `slice` the ones you want to keep. Since we want to keep the six largest values, we should sort in *descending* order so that the largest ones are at the top:

```
d %>% arrange(desc(x)) %>%
  slice(1:6)
```

```
# A tibble: 6 x 1
      x
  <dbl>
1    10
2    10
3     9
4     8
5     8
6     7
```

We want a method that will work no matter what the numbers in `x` actually are, so that you cannot just `filter` the numbers that are 7 or larger, because the assignment marks for a different student will be different.

A different approach that will work and is not unreasonably longer than these is also fine. The techniques here are from the first part of the course, which I am expecting you to remember or to be able to find in your notes, to apply to a function-writing problem.

Extra: as ever, this problem is a simplified version of real life. In this course, each assignment is out of a different number of points, so I convert each one into a percent before finding the best 6. But that seemed to be an unnecessary complication for this problem (which was meant to be about writing functions), so I kept things as simple as I could.

(b) [3] What code would you add to your code of the previous part to find the mean of the largest six values of `x` *as a number*, and not as a dataframe?

My answer:

The last two lines of this. This is a `summarize` plus something else:

```
d %>% slice_max(x, n = 6) %>%
  summarize(mean_score = mean(x)) %>%
  pull(mean_score)
```

```
[1] 8.666667
```

or any other way you have to get the mean score out as a number, such as

```
d %>% slice_max(x, n = 6) %>%
  summarize(mean_score = mean(x)) %>%
  pluck("mean_score")
```

```
[1] 8.666667
```

(with `pluck`, you need quotes around the column name or else you get an error), or (without `summarize`):

```
d %>% slice_max(x, n = 6) -> dd
mean(dd$x)
```

```
[1] 8.666667
```

Without extra cleverness, I don't see how you can do this without saving the result of the previous part.

Or even:

```
d %>% slice_max(x, n = 6) %>%
  summarize(mean_score = mean(x)) %>%
  as.numeric()
```

```
[1] 8.666667
```

This last takes the whole dataframe that comes out of the `summarize` and makes it into a "numeric thing" by pulling out all the numbers in it (there is only one).

I am happy to accept either the two lines you would add, or the whole pipeline. Two points for working out the mean somehow (call it whatever you like), and one for turning it into a number.

(c) [3] Write a function called `best` that accepts as input a dataframe containing a column `x`, and outputs the mean of the six largest values in `x` as a number.

My answer:

This means collecting together the two things you did before and putting them into a function properly coded, such as this:

```
best <- function(dd) {
  dd %>% slice_max(x, n = 6) %>%
  summarize(mean_score = mean(x)) %>%
  pull(mean_score)
}
```

Call your input dataframe whatever you like, as long as you *use the same dataframe name in the body of your function.*

I recognize that your code may differ from mine. In this part, I am mainly concerned that you structure your function correctly, so you get full marks for the function structure:

- `best` defined as the word `function` with some input,
- an open curly bracket,
- a function body that uses your input dataframe and the code from the previous two parts, or something that looks like an attempt to find the six largest values and then the mean of them,
- a close curly bracket.

One point each for:

- the top line,
- having curly brackets around the rest of it,
- having a sensible-looking function body.

The grader does not need to check the last too carefully, just that the pipeline starts with the same dataframe as listed on the top line, and that it seems to be the same code as you had in the previous two parts.

That means that you could get the previous two parts completely wrong (or not even know how to do them), and still get full marks for this part by copying your answers to those parts into the function body, or writing where the function body goes "the right answer to the previous

two parts". Show me that you know what to do, even if you can't do it, and you can get some points.

This function above is good R style, returning the output from the pipeline. It also works to *save* the output from the pipeline, in, say, `ans`, and then having a `return(ans)` on the next line. This might be good style in Python, but it is not good style in R (even though it works), so it will cost you a half point.

(d) [2] What code would run your function with the dataframe shown in Figure 28 as input?

My answer:

As simple as this, the dataframe in the Figure being called `d`:

```
best(d)
```

```
[1] 8.666667
```

You don't, as ever, need the output, just the line of code.

Once again, you can get this without being able to answer anything else in this question, so it is not a good idea to say to yourself "I don't know anything about functions" and skip the entire question. As long as you know how to *use* a function (which you have actually been doing all through the course), you can get two marks here.

(e) [2] The result of running your function on the data in Figure 28 is shown in Figure 29. (The `[1]` next to the output indicates that the function returns a single value.) By looking at Figure 28, but *without doing any calculation*, how do you know that this result is at least close to being correct?

My answer:

The lowest two values of `x` in the Figure are (clearly) 3 and 4, with the others being between 7 and 10. Hence the mean of the six highest values must be somewhere between 7 and 10, as 8.67 is.

That kind of thinking. I'm checking to see whether you have some kind of intuition about what the right answer should be. When you are testing a function, you need to see (i) whether it works at all, and (ii) whether the answer you get is at least broadly reasonable. (If it is, you might haul out your calculator and see whether it is actually *correct*, but if it is not reasonable, you know that you have some more work to do.)

(f) [2] What code would you add to your function to give an error if the input dataframe was based on something other than eight assignments?

My answer:

The number of assignments in the input dataframe is the number of values of `x`, which is also the number of rows of the dataframe. So you need to find (before doing any calculation) how many rows the dataframe has, and use `stopifnot` to throw an error if that number of rows is not 8. The most direct way is this:

```
best <- function(dd) {
  stopifnot(nrow(dd) == 8)
  dd %>% slice_max(x, n = 6) %>%
  summarize(mean_score = mean(x)) %>%
  pull(mean_score)
}
```

(all you need is the `stopifnot` line.)

Any other way of getting hold of the number of rows in the input dataframe is also good. This way works:

```
best <- function(dd) {
  dd %>% summarize(n_row = n()) %>% pull(n_row) -> n_rows
  stopifnot(n_rows == 8)
  dd %>% slice_max(x, n = 6) %>%
  summarize(mean_score = mean(x)) %>%
  pull(mean_score)
}
```

Or something equivalent to that. Only the lines inside the curly brackets down to (and including) the `stopifnot` line are needed. The extra work is needed now because `summarize` returns a dataframe, and as in a previous part, we need to turn it into a number (via one of the ways we saw before).

I don't think you have seen `stop` in this course, but I will accept that instead of `stopifnot`. The logic to this is backwards from `stopifnot`, because now you are saying under what conditions the function should stop with an error (the condition inside `stopifnot` is what should be true if you want to *continue*). `stop` works with `if`, like this:

```
best0 <- function(dd) {
  if (nrow(dd) != 8) stop("input needs to have 8 rows")
  dd %>% slice_max(x, n = 6) %>%
  summarize(mean_score = mean(x)) %>%
  pull(mean_score)
}
```

Inside the `stop` goes the error message that will be stopped with if the thing inside the `if` is true. This is more customizable than `stopifnot`, but you have to customize it. `stopifnot` creates an error message for you, but `stop` does not.

I asked specifically for your revised function to give an error if the input does not have eight rows. You can try to make an if-else to test whether the input has eight rows and do something different according to whether it does or does not, but that only returns something different (which might be some text like the word "error", rather than an actual error.)

Extra: To verify that I get an error when I should, I also need an input dataframe with something other than 8 rows, so make one first (with only three rows):

```
dx <- tibble(x = 1:3)
best(dx)
```

```
Error in best(dx): n_rows == 8 is not TRUE
```

and the second variant of the function, with the error message that I put in the `stop`:

```
best0(dx)
```

```
Error in best0(dx): input needs to have 8 rows
```

and this still works, on the original dataframe:

```
best(d)
```

```
[1] 8.666667
```

(g) [3] The course grade will usually be based on the best six assignments, but will sometimes be based on a different number. What changes would you make to your function to allow the calculation to be based on a different number of best assignments, but to use the best 6 assignments if the number of best ones to use is not specified? Adapt your code from part (c) for this.

My answer:

This is allowing the number of best assignments to be a second input, with a default value:

```
best <- function(dd, n_best = 6) {
  dd %>% slice_max(x, n = n_best) %>%
  summarize(mean_score = mean(x)) %>%
  pull(mean_score)
}
```

Points: one each for three things, either in your function written out in full, or as listed changes to your previous work:

- a second input to the function with a suitable name (that reminds you what it is)
- a default value of 6 for that second input
- replacing the 6 in your `slice_max` (or whatever other mechanism you used to decide how many best values to keep) by the second input, whatever you called it.

If you write your function out in full, you would do well to identify the changes you made (eg. by circling them) to make it easier for the grader to find them (and therefore easier for you to

get credit for your work).

The aim of this question was for you to handle the default input properly, so if you had those three things above, I didn't mind so much what the rest of your code said: as long as it looked like a relevant calculation using your second input, I was happy. There were a few too many people using something like `n` that appeared from nowhere: if you use it in your function, it has to be either input to your function, or calculated within the function.

Extra: To verify that my code works, using the newest version of my function:

```
best(d, 3)
```

```
[1] 9.666667
```

```
best(d)
```

```
[1] 8.666667
```

The three highest values are 10, 10, and 9, so the mean of those should be (and is) between 9 and 10. The second test should give (and does give) the same answer as before, taking the value of `n_best` to be 6.

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(MASS)
library(tidyverse)
library(smmr)
library(broom)
```

Figure 1: Packages that are loaded in this exam

when	year	deaths
before	1962	4.9
before	1963	5.1
before	1964	5.2
before	1965	5.1
before	1966	5.3
after	1967	5.1
after	1968	4.9
after	1969	4.7
after	1970	4.2
after	1971	4.2

Figure 2: Illinois traffic deaths data

```
-- Column specification --------------------------------------------------------
cols(
  when = col_character(),
  year = col_double(),
  deaths = col_double()
)
```

Figure 3: Illinois traffic deaths: graph

```
    Welch Two Sample t-test

data:  deaths by when
t = -2.5717, df = 5.0359, p-value = 0.0248
alternative hypothesis: true difference in means between group after and group before is less than 0
95 percent confidence interval:
       -Inf -0.1088462
sample estimates:
 mean in group after mean in group before
                4.62                 5.12
```

Figure 4: Illinois traffic deaths: output

```
  tibble(sim = 1:1000) %>%
    rowwise() %>%
    mutate(my_sample = list(rnorm(40, 90, 25))) %>%
    mutate(t_test = list(t.test(my_sample, mu = 100, alternative = "less"))) %>%
    mutate(p_val = t_test$p.value) %>%
    count(p_val <= 0.05)
```

```
# A tibble: 2 x 2
  `p_val <= 0.05`     n
  <lgl>           <int>
1 FALSE             210
2 TRUE              790
```

Figure 5: Power analysis

```
Rows: 24 Columns: 1
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (1): copper

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.
```

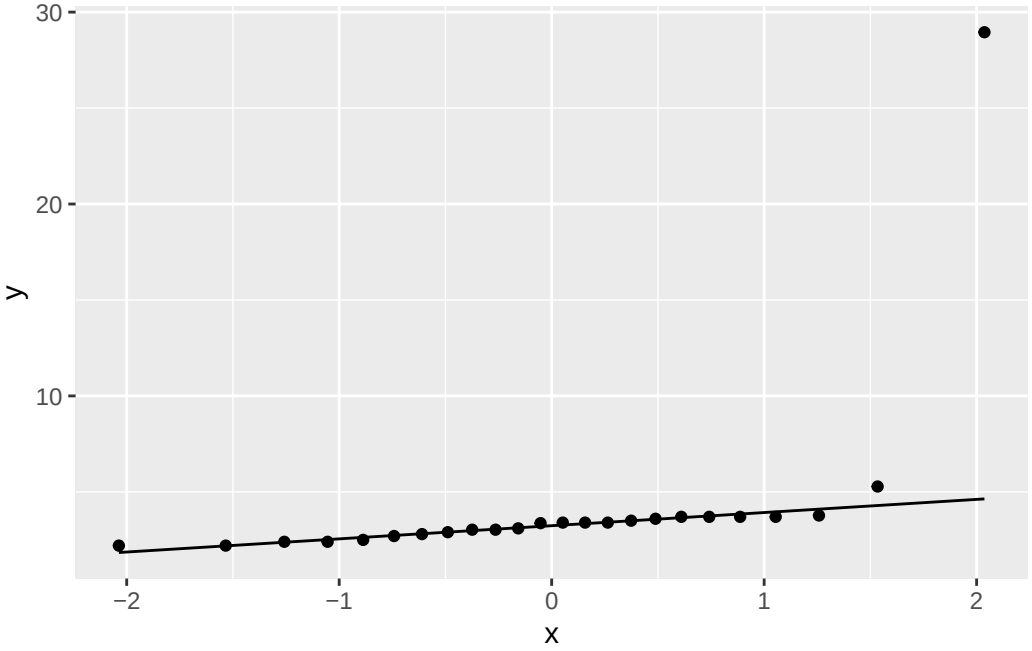


Figure 6: Normal quantile plot of copper data

```
$above_below
below above
   22     2

$p_values
  alternative     p_value
1       lower 0.000017941
2       upper 0.999998510
3   two-sided 0.000035882
```

Figure 7: Sign test for copper data

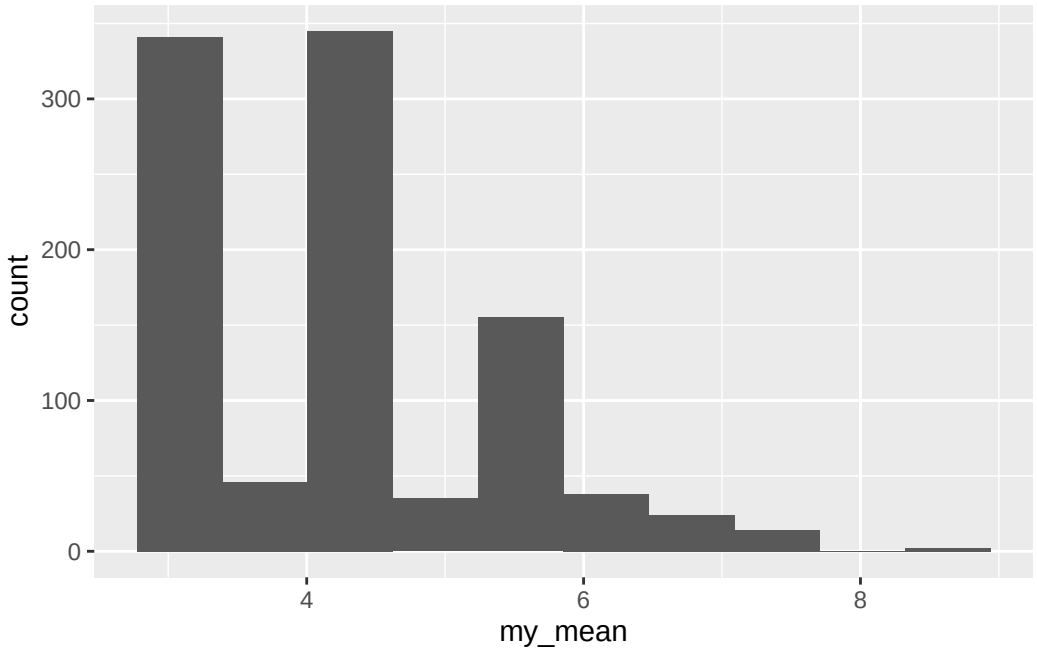


Figure 8: Bootstrap sampling distribution of sample mean for copper data

```
Rows: 39 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): ration
dbl (1): weight_gain

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.
```

```
  chickens
```

```
# A tibble: 39 x 2
   ration  weight_gain
   <chr>         <dbl>
 1 Ration1           4
 2 Ration2           3
 3 Ration3           6
 4 Ration1           4
 5 Ration2           4
 6 Ration3           7
 7 Ration1           7
 8 Ration2           5
 9 Ration3           7
10 Ration1           3
# i 29 more rows
```

Figure 9: Chicken weight gain data (some)

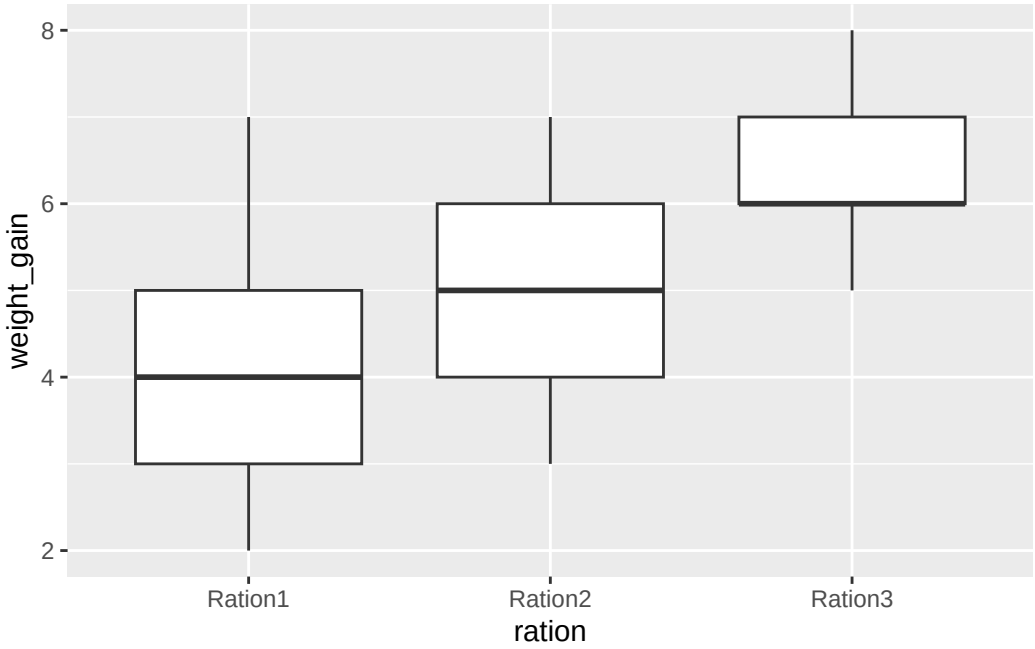


Figure 10: Boxplot of chicken weight gain data

```
chickens.1 <- aov(weight_gain ~ ration, data = chickens)
summary(chickens.1)
```

```
            Df Sum Sq Mean Sq F value   Pr(>F)
ration       2  32.97  16.487   12.17 9.17e-05 ***
Residuals   36  48.77   1.355
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 11: Chicken weight gain analysis 1

```
oneway.test(weight_gain ~ ration, data = chickens)
```

```
    One-way analysis of means (not assuming equal variances)

data:  weight_gain and ration
F = 13.207, num df = 2.000, denom df = 23.094, p-value = 0.00015
```

Figure 12: Chicken weight gain analysis 2

```
median_test(chickens, weight_gain, ration)
```

```
$grand_median
[1] 5

$table
         above
group     above below
  Ration1     2     8
  Ration2     4     4
  Ration3    11     0

$test
       what        value
1 statistic 1.415882e+01
2        df 2.000000e+00
3   P-value 8.422685e-04
```

Figure 13: Chicken weight gain analysis 3

```
pairwise_median_test(chickens, weight_gain, ration)
```

```
# A tibble: 3 x 4
  g1      g2       p_value adj_p_value
  <chr>   <chr>      <dbl>       <dbl>
1 Ration1 Ration2 0.180        0.539
2 Ration1 Ration3 0.000415     0.00125
3 Ration2 Ration3 0.00494      0.0148
```

Figure 14: Chicken weight gain followup analysis 1

```
TukeyHSD(chickens.1)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = weight_gain ~ ration, data = chickens)

$ration
                     diff        lwr      upr     p adj
Ration2-Ration1 0.8461538 -0.2697301 1.962038 0.1670506
Ration3-Ration1 2.2307692  1.1148853 3.346653 0.0000620
Ration3-Ration2 1.3846154  0.2687315 2.500499 0.0121438
```

Figure 15: Chicken weight gain followup analysis 2

```
library(PMCMRplus)
gamesHowellTest(weight_gain ~ factor(ration), data = chickens)
```

```
    Pairwise comparisons using Games-Howell test
data: weight_gain by factor(ration)
        Ration1 Ration2
Ration2 0.23593 -
Ration3 0.00039 0.00416

P value adjustment method: none
alternative hypothesis: two.sided
```

Figure 16: Chicken weight gain followup analysis 3

```
d1
```

```
# A tibble: 3 x 3
      r     A     B
  <dbl> <dbl> <dbl>
1     1    10    21
2     2    12    19
3     3    13    22
```

Figure 17: Dataframe `d1`

```
d2
```

```
# A tibble: 6 x 3
      r treatment     y
  <dbl> <chr>     <dbl>
1     1 A            10
2     1 B            21
3     2 A            12
4     2 B            19
5     3 A            13
6     3 B            22
```

Figure 18: Dataframe `d2`

```
d3
```

```
# A tibble: 2 x 4
  malexS1 malexS2 femalexS1 femalexS2
    <dbl>   <dbl>     <dbl>     <dbl>
1    21.5    14.5      14.8      12.1
2    19.6    17.4      15.6      11.4
```

Figure 19: Dataframe `d3`

```
d4
```

```
# A tibble: 4 x 3
      r g         y
  <dbl> <chr> <dbl>
1     1 A        10
2     2 B        12
3     2 C        14
4     3 B        17
```

```
d4 %>% pivot_wider(names_from = g, values_from = y)
```

Figure 20: Dataframe `d4` and some code that uses it

```
Rows: 30 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (4): taste, Acetic, H2S, Lactic

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.
```

```
cheddar
```

```
# A tibble: 30 x 4
   taste Acetic   H2S Lactic
   <dbl>  <dbl> <dbl>  <dbl>
 1  12.3   4.54  3.14   0.86
 2  20.9   5.16  5.04   1.53
 3  39     5.37  5.44   1.57
 4  47.9   5.76  7.50   1.81
 5   5.6   4.66  3.81   0.99
 6  25.9   5.70  7.60   1.09
 7  37.3   5.89  8.73   1.29
 8  21.9   6.08  7.97   1.78
 9  18.1   4.90  3.85   1.29
10  21     5.24  4.17   1.58
# i 20 more rows
```

Figure 21: Cheddar cheese data

```
cheddar %>% pivot_longer(-taste, names_to = "x_name", values_to = "x_value") %>%
  ggplot(aes(x = x_value, y = taste)) + geom_point() + geom_smooth() +
  facet_wrap(~ x_name, scales = "free", ncol = 2)
```

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

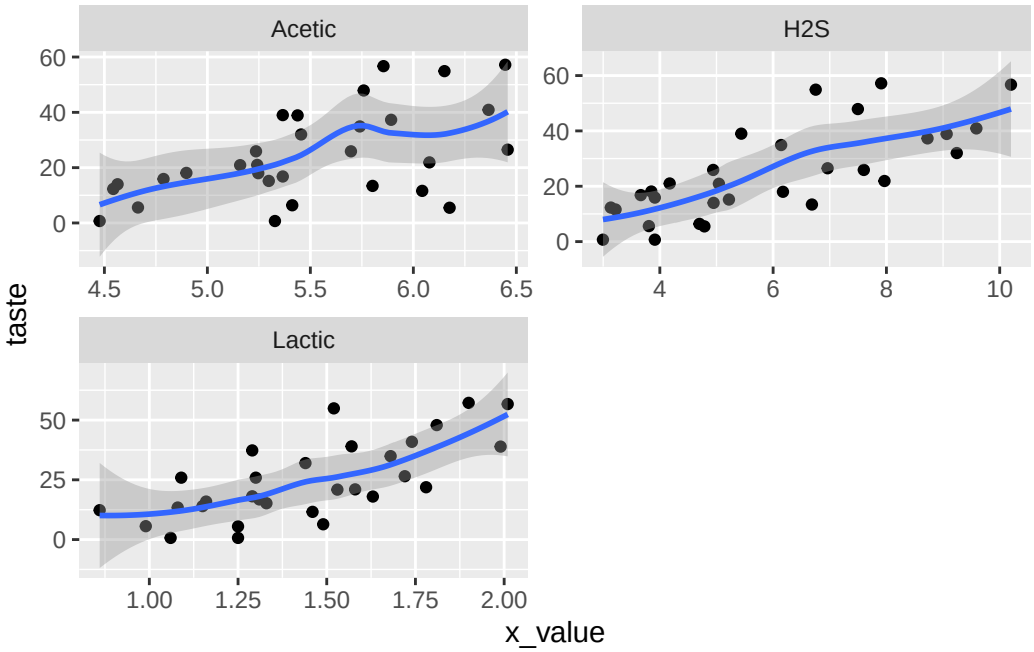


Figure 22: Cheddar scatterplots

```r
cheddar.1 <- lm(taste ~ Acetic + H2S + Lactic, data = cheddar)
summary(cheddar.1)
```

```
Call:
lm(formula = taste ~ Acetic + H2S + Lactic, data = cheddar)

Residuals:
    Min      1Q  Median      3Q     Max
-17.390  -6.612  -1.009   4.908  25.449

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -28.8768    19.7354  -1.463  0.15540
Acetic        0.3277     4.4598   0.073  0.94198
H2S           3.9118     1.2484   3.133  0.00425 **
Lactic       19.6705     8.6291   2.280  0.03108 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.13 on 26 degrees of freedom
Multiple R-squared:  0.6518,    Adjusted R-squared:  0.6116
F-statistic: 16.22 on 3 and 26 DF,  p-value: 3.81e-06
```

Figure 23: Cheddar regression model 1

```
ggplot(cheddar.1, aes(x = .fitted, y = .resid)) + geom_point()
```

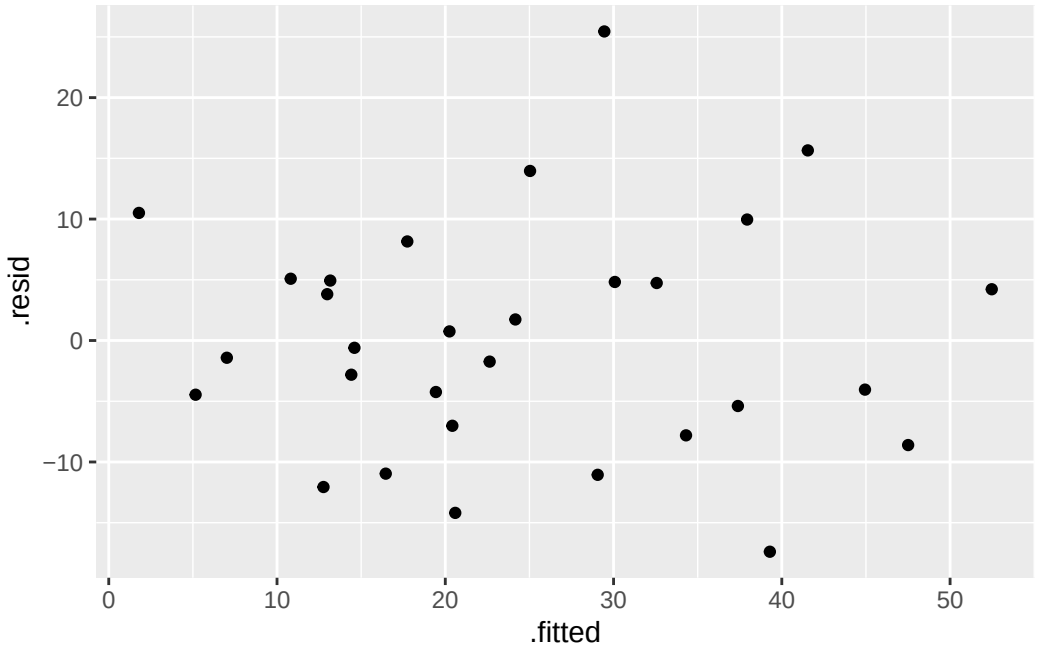


Figure 24: Cheddar residual plots 1

```
ggplot(cheddar.1, aes(sample = .resid)) + stat_qq() + stat_qq_line()
```

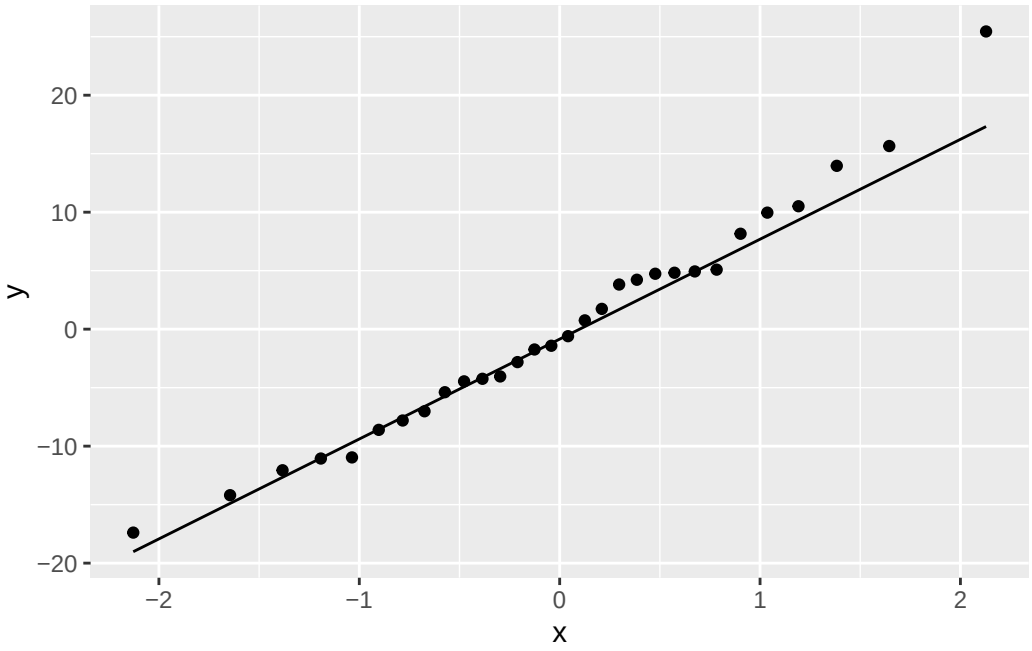


Figure 25: Cheddar residual plots 2

```
cheddar.1 %>% augment(cheddar) %>%
  pivot_longer(c(H2S, Acetic, Lactic), names_to = "x_names", values_to = "x_values") %>%
  ggplot(aes(x = x_values, y = .resid)) + geom_point() +
  facet_wrap(~ x_names, scales = "free", ncol = 2)
```

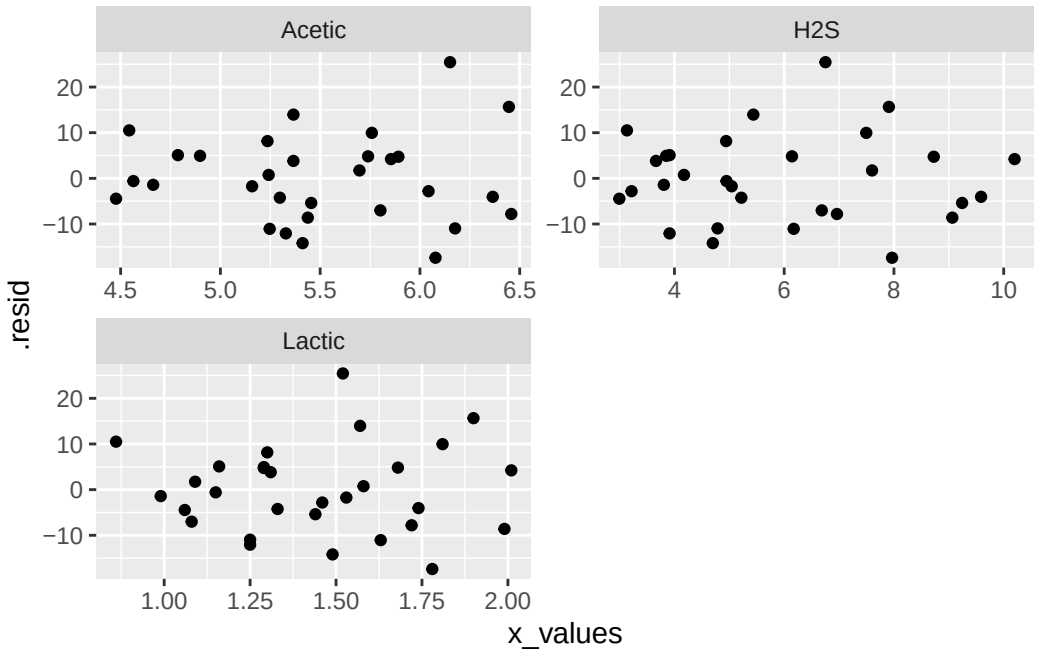


Figure 26: Cheddar residual plots 3

```
boxcox(taste ~ Acetic + H2S + Lactic, data = cheddar)
```

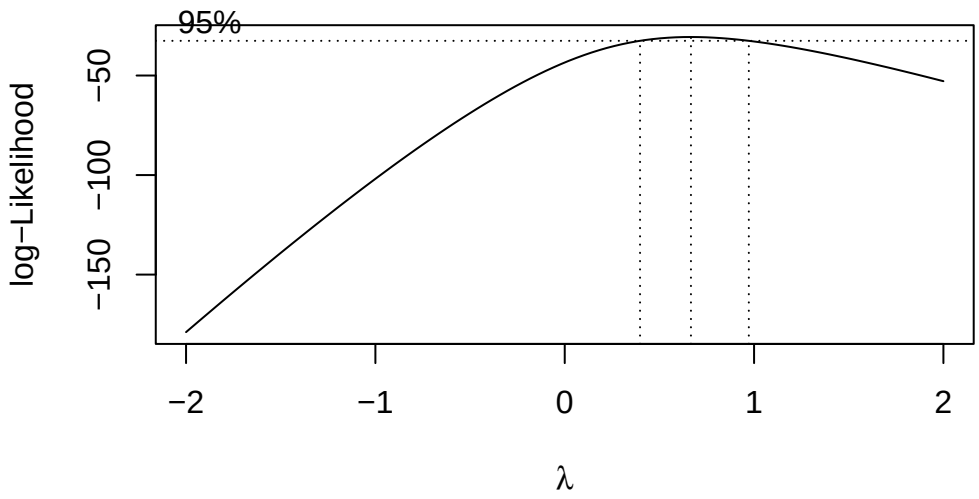


Figure 27: Cheddar further analysis

```
d
```

```
# A tibble: 8 x 1
      x
  <dbl>
1    10
2     4
3     8
4     7
5     3
6     9
7     8
8    10
```

Figure 28: Assignment marks for a student

```
[1] 8.666667
```

Figure 29: Result of running function