University of Toronto Scarborough
Department of Computer and Mathematical Sciences
STAC32 (K. Butler), Midterm Exam
December 8, 2025

Aids allowed (on paper, no computers):

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

This exam has 13 numbered pages of questions including 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.

Ontario municipalities

The province of Ontario is divided into 414 local municipalities (towns, cities, etc). These are stored in a data file called `on.txt`, in the same folder from which you are currently running R Studio. The file is laid out as shown in Figure 2. There are no tabs in the data file; everything that looks like spaces is spaces. The columns are respectively the name of the municipality, its Census Subdivision Type (one of "village", "town", "township", "municipality", "city"), and its population at the last census.

(1) (3 points) What code will read the data from the file into a dataframe called `ontario`, and display at least some of that dataframe?

From the Figure, the values are in aligned columns (or, separated by varying amounts of whitespace), so `read_table` is called for:

```r
ontario <- read_table("on.txt")
```

```
-- Column specification --------------------------------------------------------
cols(
  name = col_character(),
  csd_type = col_character(),
  population = col_double()
)
```

```r
ontario
```

name	csd_type	population
Addington-Highlands	township	2534
Adelaide-Metcalfe	township	3011
Adjala-Tosorontio	township	10989
Admaston/Bromley	township	2995
Ajax	town	126666
Alberton	township	954
Alfred-and-Plantagenet	township	9949
Algonquin-Highlands	township	2588
Alnwick/Haldimand	township	7473
Amaranth	township	4327

Make sure your `read_table` clearly has an underscore in the middle and not a dot. We did not learn `read.table` in this course.

Two points for code to read the file, using `read_table`. No points for something that looks like `read.table`. One point for something that will display at least some of the dataframe you read in. (This is meant to be a warmup.)

Points:

- 3: correctly using `read_table` to read from the file, and doing something to display at least some of the data
- 2: as 3, but not correctly displaying the data
- 2: as 3, but using `read_table` incorrectly (somehow)
- 1: code to correctly display the data, but using something other than `read_table` (that includes `read.table`, which was not in this course).

(2) (2 points) Why do you think some of the `names` in Figure 2 have dashes or underscores in them that don't seem to belong? Explain briefly.

The dashes and underscores are replacing spaces. If the names had spaces in them (for example, "North Stormont") `read_table` would not know that this is supposed to be a municipality name as a whole, and would try to make North the `name` and Stormont the `csd_type`, which is wrong because North Stormont as a whole is actually the name of a township.

Points:

- 2: the dashes and underscores are replacing spaces, because otherwise the spaces would be taken as a separation between variables (rather than a municipality name with spaces *in* it, that is two words). You can use an example to make the point about spaces.
- 1: as 2, but without a good reason

Extra: these are actually the `towny` data from the `gt` package, which is for making data tables. There were more columns than these originally, but I only gave you these three:

```r
towny <- gt::towny
towny %>% select(name, csd_type, population_2021) -> munic
munic %>% slice_sample(n = 20)
```

name	csd_type	population_2021
Greater Sudbury	city	166004

name	csd_type	population_2021
Johnson	township	749
Sioux Lookout	municipality	5839
Ingersoll	town	13693
West Elgin	municipality	5060
Coleman	township	517
Hilton	township	382
Dawson	township	399
Kirkland Lake	town	7750
Milton	town	132979
Burpee and Mills	township	382
Norfolk	city	67490
Belleville	city	55071
North Bay	city	52662
Kerns	township	330
Arnprior	town	9629
Brantford	city	104688
Merrickville–Wolford	village	3135
Westport	village	634
Lake of the Woods	township	308

I wanted to give you a `read_table` problem, so I wanted to align these in columns (although any variable number of spaces between data values would have worked):

```
strings <- with(munic, sprintf("%-35s %-15s %8d", name, csd_type, population_2021))
strings[1:10]
```

```
 [1] "Addington Highlands                 township            2534"
 [2] "Adelaide Metcalfe                   township            3011"
 [3] "Adjala-Tosorontio                   township           10989"
 [4] "Admaston/Bromley                    township            2995"
 [5] "Ajax                                town              126666"
 [6] "Alberton                            township             954"
 [7] "Alfred and Plantagenet              township            9949"
 [8] "Algonquin Highlands                 township            2588"
 [9] "Alnwick/Haldimand                   township            7473"
[10] "Amaranth                            township            4327"
```

`sprintf` makes formatted text: respectively, `name` left-justified in a field of width 35 characters, `csd_type` left-justified in a field of width 15 characters, and population *right*-justified in a field of width 8 characters. There is one value of `string` for each municipality (ie., over 400) so I display the first ten only for checking. You can see that these ones are correctly aligned columns.

The format codes (and, indeed, `sprintf`) come from the C programming language: `s` means text, `d` means a (whole) number, the numeric values denote the number of characters wide, a negative number means left-justified, and a positive number means right-justified.

Next, I write these to a file, which will then be formatted text, as I want it. Well, more or less:

```
writeLines(strings, "on0.txt")
```

Here's what that file looks like now, or at least some of it:[1]

```
head on0.txt -n 20
```

```
Addington Highlands                 township           2534
Adelaide Metcalfe                   township           3011
Adjala-Tosorontio                   township          10989
Admaston/Bromley                    township           2995
Ajax                                town             126666
Alberton                            township            954
Alfred and Plantagenet              township           9949
Algonquin Highlands                 township           2588
Alnwick/Haldimand                   township           7473
Amaranth                            township           4327
Amherstburg                         town              23524
The Archipelago                     township            979
Armour                              township           1459
Armstrong                           township           1199
Arnprior                            town               9629
Arran-Elderslie                   municipality       6913
Ashfield-Colborne-Wawanosh      township           5884
Asphodel-Norwood                  township           4658
Assiginack                          township           1008
Athens                              township           3042
```

[1]This `head` comes from the BASH shell, my command line.

There were actually three problems (one of which you see above), which I solved by hand-editing (probably not the most efficient way):

- the second and third columns can get misaligned if the municipality names had one or more dashes in them originally. (But not always.) I added extra spaces to make these line up.
- Some of the municipality names were *really* long, like longer than 35 characters, so I abbreviated them to something still readable.
- A good number of the municipality names had spaces *inside* them (such as North Stormont, or, closer to home, Richmond Hill). I replaced those spaces with dashes or underscores as seemed appropriate. That was actually the motivation for this question, once I saw what happened if I *did not* do this.

The tidied-up file is what is in my `on.txt` that you had to imagine was in your folder.

The `readr` package (part of the Tidyverse, where `read_csv`, `read_table`, and the others live) also has `read_fwf` ("fixed-width format"), where the values don't have to be delimited by anything and the columns of values are lined up, that is, the same width all the way down. Though we didn't see that in the course, this is actually the *right* way to read data like this (I didn't need to do the third bullet point above in my `on0.txt`) because you can then have spaces *inside* data with no problem. Here's how that goes:

```
ontario_fwf <- read_fwf("on0_edited.txt",
                        fwf_widths(c(35, 15, 12),
                                   col_names = c("name", "csd_type", "population")))
```

```
Rows: 414 Columns: 3
-- Column specification --------------------------------------------------------

chr (3): name, csd_type, population

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.
```

```
ontario_fwf %>% slice_sample(n = 20)
```

name	csd_type	population
Drummond/North Elmsley	township	8183
Leeds and the Thousand Islands	township	9804

name	csd_type	population
Dubreuilville	township	576
Essa	township	22970
Belleville	city	55071
Orangeville	town	30167
Temiskaming Shores	city	9634
South Glengarry	township	13330
Enniskillen	township	2825
Minto	town	9094
Whitewater	township	7225
Spanish	town	670
East Zorra-Tavistock	township	7841
Middlesex Centre	municipality	18928
Otonabee–South Monaghan	township	7087
Parry Sound	town	6879
Cramahe	township	6509
Algonquin Highlands	township	2588
Mississauga	city	717961
Lucan Biddulph	township	5680

This seems to assume that your file name *does not* have column names. What you do is to specify the column widths[2] as the first thing in `fwf_widths` inside `read_fwf`, and then the matching names you want to use for the columns in `col_names` inside `fwf_widths`. This successfully reads the municipality names including the spaces they have in them, but you'll note that everything is read in as text, including the (quantitative) populations.[3] Fixing this last thing is, as they say, left as an exercise for the reader.

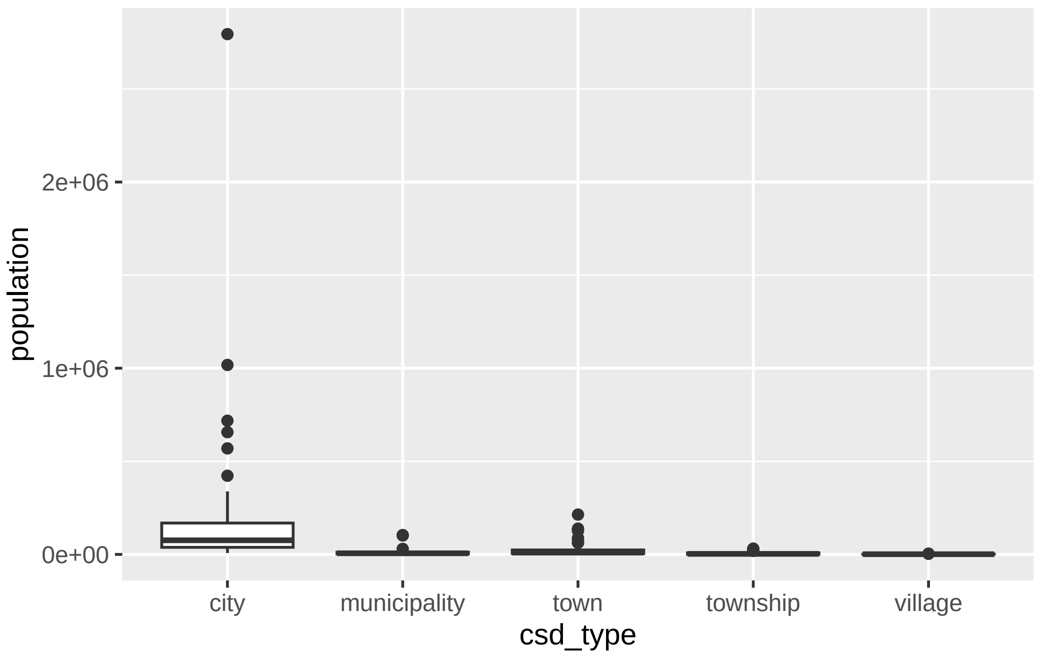
(3) (2 points) What code would make a suitable graph of the dataframe columns *not* including `name`?

The relevant columns are `csd_type`, categorical, and `population`, quantitative, so the suitable graph is a boxplot:

```
ggplot(ontario, aes(x = csd_type, y = population)) + geom_boxplot()
```

[2]It seems to need some extra characters to read the populations properly, hence 12 rather than 8, or else they get cut off.

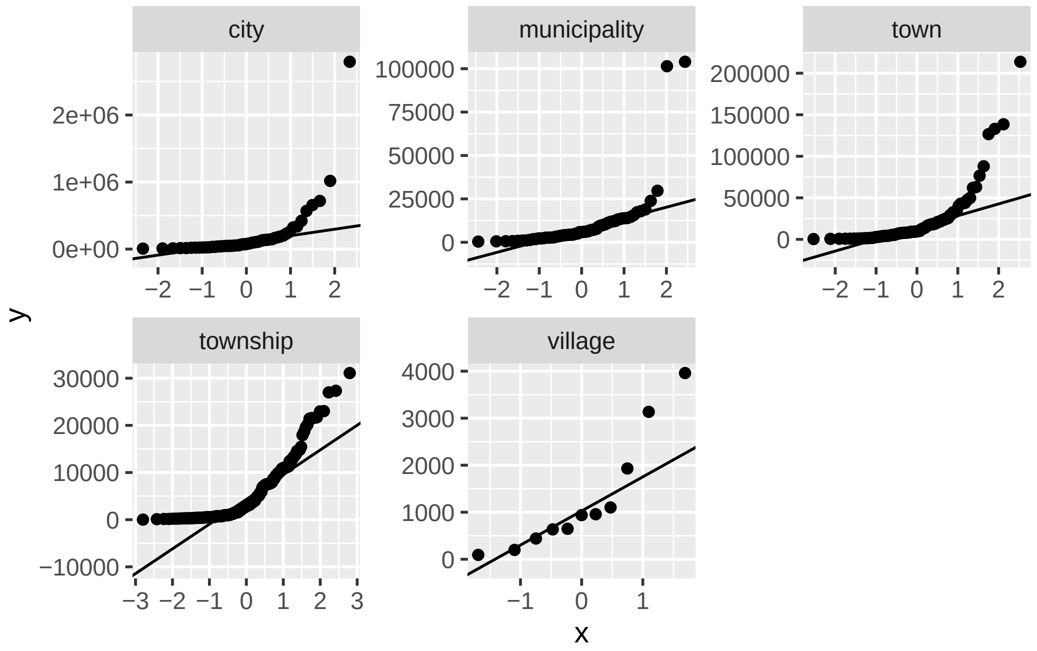
[3]Because, in this display, they are left-justified. If they had been read in as numbers, they would have been right-justified.



This is not actually a very illuminating graph, but you have no way to know that.

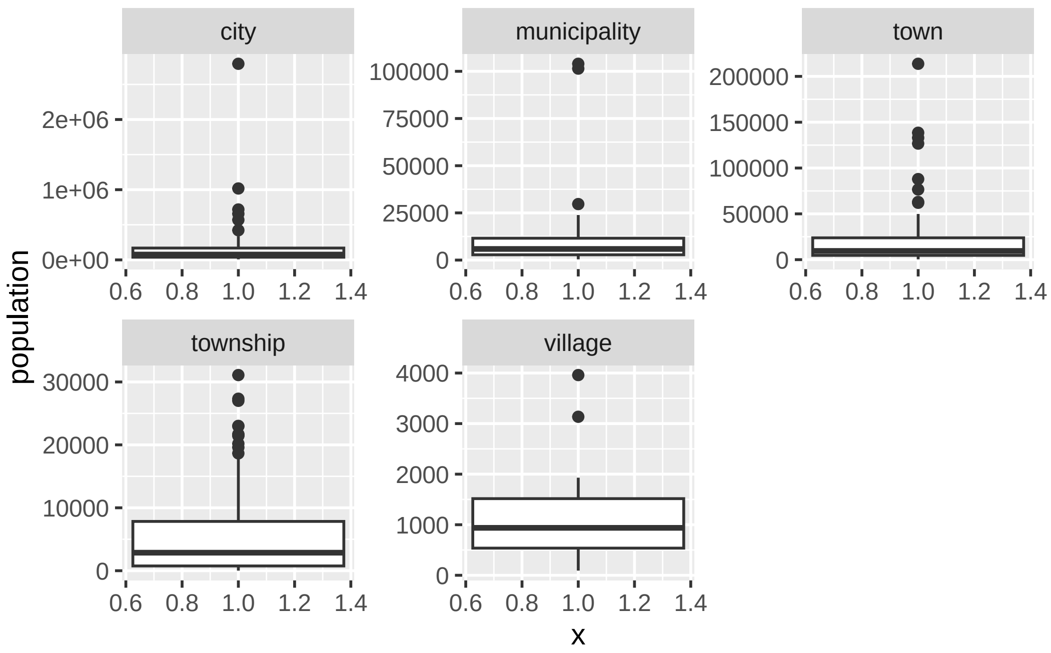
Normality is not of specific interest here, so a facetted normal quantile plot is a second-best choice:

```
ggplot(ontario, aes(sample = population)) + stat_qq() +
  stat_qq_line() + facet_wrap(~ csd_type, scales = "free")
```



Using separate scales in each facet turns out to be a good idea because the different types of municipality are of very different sizes. I can't very well penalize you for not thinking of that on an exam, though. If you were doing this on your own time, I *would* expect you to think of it, however, because your aim would be to assess each of the distributions for normality (and the plot with separate scales shows nicely that all five distributions are right-skewed.) The purpose of a boxplot is to *compare* distributions (of population) across groups (municipality types), so putting *boxplots* in facets is possible (with scales), like this:

```
ggplot(ontario, aes(x = 1, y = population)) + geom_boxplot() +
  facet_wrap(~ csd_type, scales = "free")
```



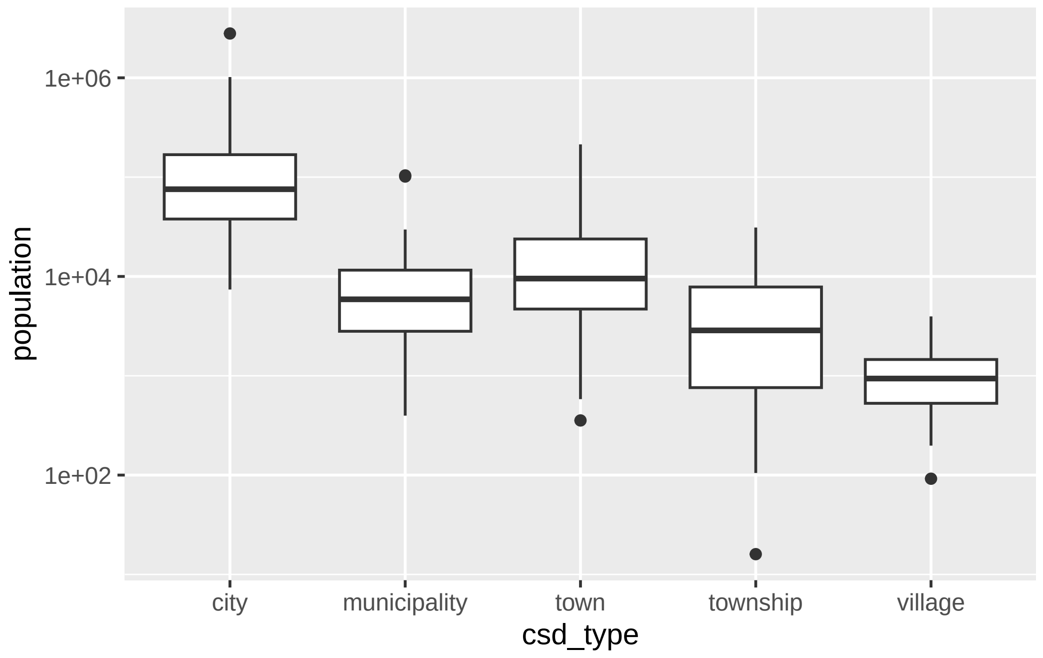
but it doesn't (easily) enable you to compare populations across municipality types. Doing a facetted boxplot without using `scales` is no different than a regular boxplot, and is much harder to get right.

Points:

- 2 for a correctly drawn boxplot.
- 1.5 for a boxplot but with a small error in the code.
- 1.5 for a correctly drawn normal quantile plot (with a facet-wrap, but not necessarily a `scales`.)
- 1 for a facetted boxplot, because "why", plus my discussion above.
- 1 for a normal quantile plot with a small error in the code.
- 1 for a boxplot with more than one error (where it is still obvious that you are trying to draw a boxplot).
- 0.5 for other code with something relevant correct.

Extra: the boxplots actually look kind of silly because most of the municipalities are very small (and, of course, the largest city is Toronto, way bigger even than the other cities). Of course, you have pretty much no way to know that on an exam. It might actually make sense to put the populations on a log scale so that the very big ones don't dominate the plot:

```
ggplot(ontario, aes(x = csd_type, y = population)) + geom_boxplot() +
  scale_y_continuous(trans = "log10")
```



The tick marks on the y-axis are equally spaced *in powers of 10*: that is, top to bottom, 1 million, 10 thousand and 100. These distributions are now much less right-skewed in shape.

(4) (3 points) What code would compute the median population of the municipalities of each census subdivision type, along with the number of municipalities there were of each census subdivision type?

This is a group-by and summarize, like this:

```
ontario %>% group_by(csd_type) %>%
  summarize(pop_median = median(population), n = n())
```

csd_type	pop_median	n
city	75649.0	52
municipality	5886.5	68
town	9525.0	88

csd_type	pop_median	n
township	2864.0	195
village	938.0	11

Give your summaries names that say something about what they are. `n` is a standard name for sample size, so that is a good name.

Ordinarily, you would probably put the sample size first and the median second (and I have no objection if you do it that way around), but I phrased the question this way ("along with" means to get both in one table) to imply that you needed to do this all in one step, rather than doing something like this first:

```
ontario %>% count(csd_type)
```

csd_type	n
city	52
municipality	68
town	88
township	195
village	11

and then trying to figure out what to do next.

Points:

- 3: group-by plus summarize with `n()` and the two summaries having good names
- 2.5: at least one unclear name of summary
- 2: an error in code but otherwise correct
- 1.5: getting the two summaries separately (eg. the counts using `count` and then getting the medians).
- 1: the two summaries separately but with a small error
- 1: an attempt to get the two summaries together but with at least two errors (but with clear enough code that it is obvious that this is what you are trying for)
- 0.5: something relevant correct

(5) (3 points) What code would display the city (according to `csd_type`) that has the smallest population, and none of the other municipalities?

This implies that the best approach is to pick out the cities only, first, and then to find the one of those that has the smallest population. The slickest way is this:

```
ontario %>%
  filter(csd_type == "city") %>%
  slice_min(population, n = 1)
```

name	csd_type	population
Dryden	city	7388

You might not have heard of this place before. It is between Thunder Bay (on the western end of Lake Superior) and the Manitoba border, and such is the size of things in northern Ontario, it is actually halfway between Thunder Bay and *Winnipeg*.

The above is the full-credit way to do it. Half a point off for sorting and slicing, because you can do it more concisely than that:

```
ontario %>%
  filter(csd_type == "city") %>%
  arrange(population) %>%
  slice(1)
```

name	csd_type	population
Dryden	city	7388

You can also get the answer (along with some other things you don't want) using a group-by:

```
ontario %>%
  group_by(csd_type) %>%
  slice_min(population, n = 1)
```

name	csd_type	population
Dryden	city	7388
Killarney	municipality	397

name	csd_type	population
Latchford	town	355
Cockburn_Island	township	16
Thornloe	village	92

or the equivalent with sorting and slicing. 1.5 points for this.

Note that if you group-by and summarize here:

```
ontario %>%
  group_by(csd_type) %>%
  summarize(pop_min = min(population))
```

csd_type	pop_min
city	7388
municipality	397
town	355
township	16
village	92

you get those populations, but *not* which cities they belong to (which was the point of the question).

Points:

- 3: `filter` plus `slice_min`, correctly.
- 2.5: a 3-point answer with a small error
- 2.5: `filter` plus sorting plus `slice`, correctly
- 2: 3-point answer with large or more than one error
- 2: 2.5 point answer with small error
- 1.5: group-by plus `slice_min` or sort-plus-`slice`, correctly (but displaying other things as well)
- 1.5: 2.5 point answer with large or more than one error
- 1: group-by plus summarize (which loses the city names)
- 1: 1.5 point answer with small error
- 0.5: something relevant correct

Silicon dioxide

The desired percentage of silicon dioxide in a certain type of cement is 5.5. As part of quality control, it is desired to design testing procedures so that for samples with mean percentage of silicon dioxide 5.6, there is a high chance of successfully rejecting a mean of 5.5 against a two-sided alternative hypothesis. Assume that the percentage of silicon dioxide in an observation of this type of cement has a normal distribution with standard deviation 0.2.

(6) (3 points) What code would estimate, *by simulation*, the probability of successfully rejecting a mean of 5.5 against a two-sided alternative when the mean percentage is actually 5.6, with a sample size of 16, under the assumptions given above?

Power by simulation, so generate some large number of samples from the truth (each one of size 16), then test against a mean of 5.5. The things you need are:

- an initial dataframe
- work rowwise
- generate random samples from the truth (using the true mean)
- test against the null (two-sided is the default, so it is unnecessary or wrong to have an `alternative`)
- grab the P-value of each simulated test
- count how many of them are 0.05 or smaller:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(rnorm(16, 5.6, 0.2))) %>%
  mutate(my_test = list(t.test(my_sample, mu = 5.5))) %>%
  mutate(p_value = my_test$p.value) %>%
  count(p_value <= 0.05)
```

p_value <= 0.05	n
FALSE	549
TRUE	451

You won't have my answer, of course.

Minus a half point per error (there are lots of ways to go wrong), down to 1 if you have something substantial correct, 0.5 if you have anything relevant, such as `power.t.test` code that appears to be correct.

(7) (1 point) Why would it also have been appropriate to use `power.t.test` here?

The distribution of percent of silicon dioxide is (assumed to be) normal.

Make sure you say what it is that is normally distributed. No points for just "normal distribution" without saying what it is that is supposed to be normal.

(8) (2 points) Some output from `power.t.test` is shown in Figure 3. Suppose it is desired to have probability 0.98 of rejecting a mean percentage of 5.5 when the percentage is actually 5.6. What does the Figure tell you about the sample size that will be necessary? Explain briefly.

The power for a sample size of 16 is only about 0.46, so to get a larger power (0.98), we will need a larger sample size than 16.

Points:

- 2: state that power for $n = 16$ is too small, so will need a larger sample size
- 1: assert that larger sample size is needed without being convincing enough about why

Extra: this is the exact version of power you estimated using simulation. I got the answer both ways, and my simulation came out reasonably close to the exact answer.

(9) (2 points) What code will calculate the exact sample size required to reject a mean percentage of 5.5 when the mean is actually 5.6, under the assumptions of the previous question?

"Calculate" means "not by simulation". A second hint is that for a simulation, you have to start with a sample size, and estimate a power for it; you can't use a simulation to estimate a sample size, except by a process of trial and error.

You'll notice that I gave you most of the code in Figure 3. So all you have to do is to remove the `n =` and replace it with a `power =`, specifically

```r
power.t.test(power = 0.98, delta = 5.6-5.5, sd = 0.2,
             alternative = "two.sided", type = "one.sample")
```

```

     One-sample t test power calculation

              n = 66.39753
          delta = 0.1
```

```
             sd = 0.2
      sig.level = 0.05
          power = 0.98
    alternative = two.sided
```

(This is meant to be easy.)

One point if you give correct simulation code for a sample size bigger than 16, and then explain what you are going to do with it. For example, something like this, copy-pasting the code from earlier:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(my_sample = list(rnorm(50, 5.6, 0.2))) %>%
  mutate(my_test = list(t.test(my_sample, mu = 5.5))) %>%
  mutate(p_value = my_test$p.value) %>%
  count(p_value <= 0.05)
```

p_value <= 0.05	n
FALSE	63
TRUE	937

I guessed a sample size of 50 (anything bigger than 16 will do), and then you say "use this result to try a different sample size, repeating until you get close enough to a power of 0.98". In this case, I have the answer, so I would know to go bigger still than 50, but of course on an exam you won't.

(Once you read the next question, you'll see where the output came from, and you will have a chance to re-think an answer based on simulation.)

Points:

- 2: correct `power.t.test` code, or description of changes from *my* `power.t.test` code.
- 1.5: 2-point answer but with one error
- 1: apparently correct simulation code for a sample size greater than 16 *and* explanation of what you will do with the result (eg. describing trial and error process)
- 1: 2 point answer with 2 or more errors but something correct
- 0.5: correct simulation code for larger sample size but no explanation of its value

(10) (2 points) The output from your code of the previous question is shown in Figure 4. What sample size should the quality control team use, to make sure they have at least the desired power? Explain briefly.

The sample size given in the output is 66.398. The actual sample size must be a whole number, and to get at least the desired power, they need to round *up*: that is, to use a sample size of $n = 67$.

Points:

- 2: sample size of 67, with some sort of explanation of why we need to round up (eg. "to get power at least 0.98")
- 1.5: sample size of 67 without sufficient explanation
- 1: sample size of 66 with some sort of justification (eg. "will give us power close to 0.98")
- 0.5: sample size of 66 without explanation

Extra: I had to do a bit of fiddling to get you an n that needed rounding up, hence the rather odd desired power of 0.98.

If you had access to R, you could verify that rounding *up* is what you needed to do, thus:

```r
power.t.test(n = 66, delta = 5.6 - 5.5, sd = 0.2,
             alternative = "two.sided", type = "one.sample")
```

```

     One-sample t test power calculation

              n = 66
          delta = 0.1
             sd = 0.2
      sig.level = 0.05
          power = 0.9793918
    alternative = two.sided
```

```r
power.t.test(n = 67, delta = 5.6 - 5.5, sd = 0.2,
             alternative = "two.sided", type = "one.sample")
```

```

     One-sample t test power calculation
```

```
              n = 67
          delta = 0.1
             sd = 0.2
      sig.level = 0.05
          power = 0.9808893
    alternative = two.sided
```

A sample size of 66 narrowly *fails* to get you enough power, but a sample size of 67 gets you there with a little to spare.

Fixing a car

Fifteen auto mechanics certified to work on a certain kind of car were randomly selected, and the time required (in minutes) for each of them to diagnose a certain problem on that car was recorded. The data, in dataframe `cars`, are shown in Figure 5. We will be assessing whether there is evidence that the true average diagnostic time is less than 35 minutes, where "average" could be mean or median.

(11) (2 points) A graph is shown in Figure 6. Based on this graph, and anything else you know about the data so far, what are *two* reasons to prefer a sign test over a t-test to assess the evidence of interest?

The usual two things: assess normality and talk about sample size (in the context of the normality you have).

- The distribution is not symmetric and therefore not normal. Or, there is an upper outlier. Or, there is a longer tail on the left.
- The sample size of $n = 15$ is not especially large, and so we cannot expect to get too much help from the Central Limit Theorem.

Hence, we should prefer a sign test over a t-test here.

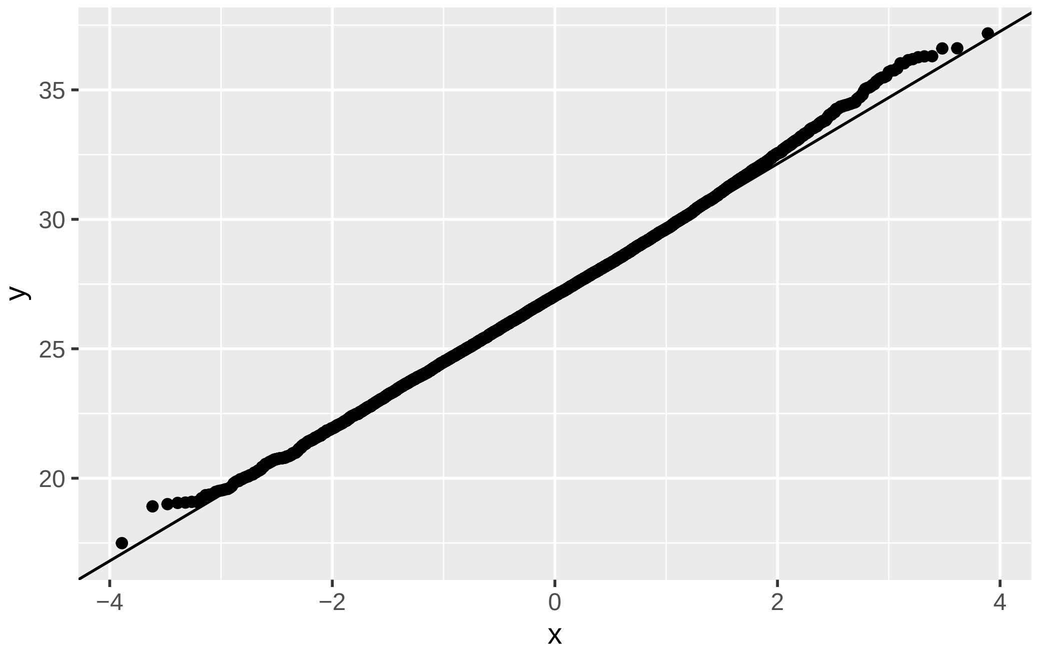
Points:

- 2: some good reason that the distribution of diagnostic times is not normal, plus an assertion that the sample size is not large, or not large enough to overcome the non-normality
- 1: a reasonable argument that the t-test is actually *all right*, such as that 15 is a large enough sample size that the Central Limit Theorem will help
- 1: discussing normality but not sample size
- 1: discussing sample size but normality

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

```
Loading required package: lattice
```

```
plot_dist(cars$diagnostic_time)
```



This, surprisingly enough, is really good. Maybe the sample size was large enough after all, or the histogram made the distribution look less normal than it actually was. But you didn't know any of this, so you had to work with what you had (and I told you which conclusion to draw, so you needed to find evidence in favour of that conclusion).

(12) (2 points) The results of running a suitable test are shown in Figure 7. What do you conclude from this Figure, in the context of the data?

This is a sign test (as implied by the word "suitable" and the discussion in the previous question).

The null hypothesis is that the population median is 35 and the alternative is that the population median is less than 35 (from the data description). The appropriate P-value is the lower-tail 0.0009. This is (much) less than 0.05, so reject the null hypothesis in favour of the alternative and conclude that the median diagnosis time is less than 35 minutes.

Make sure you state the P-value, state or strongly imply that you know what the hypotheses are, and state a conclusion about diagnosis times.

Points:

- 2: cite correct P-value, reject null, conclude that median diagnosis time is less than 35 minutes
- 1.5: as 2, but without citing correct P-value
- 1: citing the two-sided P-value, or drawing a conclusion like "different from 35 minutes", or both
- 0.5: citing the two-sided P-value and then drawing a one-sided conclusion
- 0.5: "reject the null hypothesis" without saying more

Extra: it is perhaps not entirely clear *why* we would be interested in a median diagnosis time of 35 minutes. Perhaps the certification the mechanics take requires them to make this kind of diagnosis in 35 minutes, and we are testing to see whether (some time after passing the certification) they are still able to do this.

(13) (2 points) Would you expect a value of 35 to be inside a *99%* confidence interval for the median, or not? Explain briefly.

99% confidence corresponds to a two-sided α of 0.01, and therefore a one-sided α of 0.005. We rejected a null hypothesis that the median was 35 with a two-sided P-value 0.0018, less than 0.01 (or a one-sided P-value of 0.0009, less than 0.005) so we would expect 35 to be *outside* the confidence interval.

"Outside the interval because we rejected the null" is a one-point answer. To get the second point, you need to make the connection between the confidence level, the corresponding α for a test, and whether the P-value was less than that. Strictly, you should use the *two-sided* P-value from Figure 7, or halve the α, but I'm willing to let that go if you get the rest of it right. The purpose of this question was to see whether you know how the confidence level, α, and inside or outside are connected.

Points:

- 2: 99% CI goes with $\alpha = 0.01$, and two-sided P-value 0.0018 less than 0.01 (or one-sided P-value 0.0009 less than 0.01), so 35 is *outside* the confidence interval.
- 1: "outside because we rejected the null"
- 1: as 2, but concluding that 35 is *inside* on the same evidence.

To verify:

```
ci_median(cars, diagnostic_time, conf.level = 0.99)
```

```
[1] 15.60293 31.89675
```

The value 35 is indeed clearly outside the interval (as we would expect, because the two-sided P-value is some way less than 0.01). Also, the interval is rather wide, reflecting that we didn't have much data, and the diagnosis times that we observed were rather variable.

Because of the way we construct the confidence interval for the median (directly from whether or not we reject the null in the corresponding sign test), this correspondence for the sign test is an exact one.

Cooling pork

Two different cooling methods for pork meat are being compared. The meat from each pig was divided into two parts. One part (chosen at random) was exposed to `rapid` cooling, and the other part was put through a cooling `tunnel`. After the meat had been cooled, the meat was assessed for tenderness using a standard scale, where a higher score means the meat is more tender. The data are shown in Figure 8. The other columns of interest to us are an ID for the `pig`, and the difference `delta` between the tenderness of the meat cooled by the two different methods.

(14) (2 points) How do you know this is a matched-pairs study?

Meat from the same pig is cooled by both methods, or each pig produces two observations (tenderness of meat cooled by both methods), or pairs of tenderness observations come from the same pig.

Points:

- 2: something equivalent to above
- 1: as 2, but argument not precise enough

Extra: we are not considering the `ph` column here (which was the pH of the diet the pig was on). Considering this would result in some extra complications, which I might discuss in a later Extra.

(15) (3 points) Some graphs are shown in Figure 9 and Figure 10. Using the appropriate one(s) of these graphs, how can you support the experimenter's assertion that a *t*-test will be appropriate? Your answer should clearly indicate which of the graphs in the two Figures you are using.

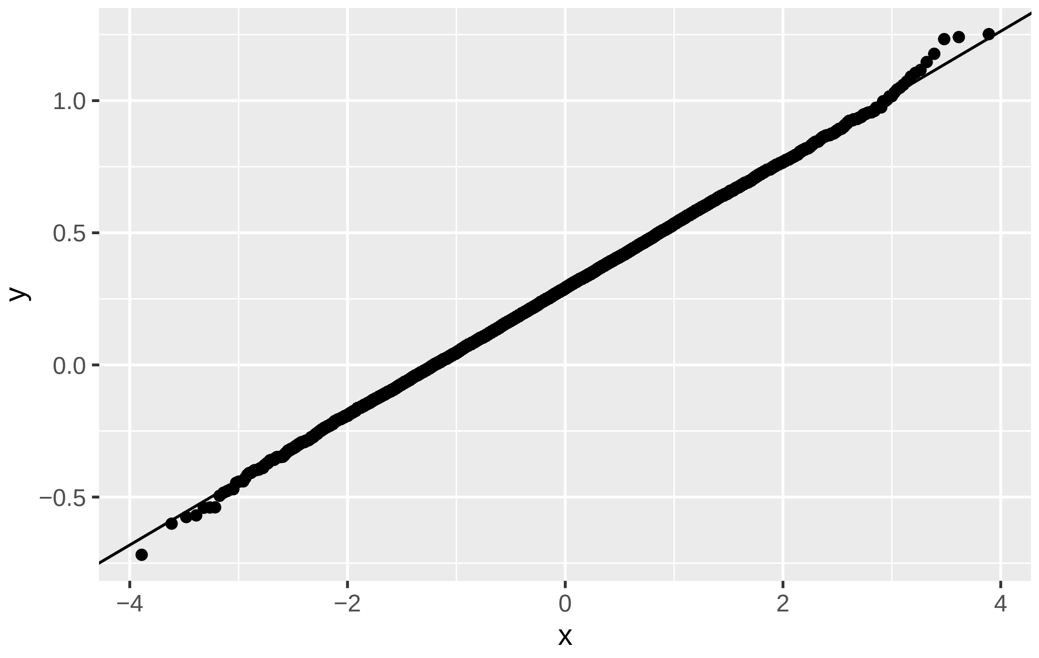
This will be a matched-pairs t-test, so the relevant graph is the one of the differences in Figure 10. (The two graphs in Figure 9 are not relevant at all, because the normality of the two variables individually does not matter.)

This distribution is slightly long-tailed relative to the normal. To argue that this is not a problem, you will need to argue that a sample size of 18 (18 pigs, that is 18 pairs of observations) is large enough to overcome the non-normality that you see.

Points:

- 3: reference to Figure 10 and not Figure 9, assessment of normality (long-tailed), assertion that sample size is large enough to overcome this (or that CLT will help enough).
- 2: as 3, but without correct discussion of sample size
- 2: as 3, but without correct of normality
- 1.5: as 3 but with any discussion of Figure 9
- 1: logically arguing that a t-test is *not* appropriate

Extra: of course, I got the bootstrap sampling distribution of the sample mean difference:



and, as you see, there is no problem there at all. (It actually surprised me how normal this was.)

(16) (2 points) Output from a suitable t-test is shown in Figure 11. What code produced this output?

Literally this:

```
with(cooling, t.test(tunnel, rapid, paired = TRUE))
```

```

    Paired t-test

data:  tunnel and rapid
t = 1.177, df = 17, p-value = 0.2554
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -0.2285208  0.8051875
sample estimates:
mean difference 
      0.2883333 
```

I would also accept this, though as you see, the output is slightly different:

```
t.test(cooling$tunnel, cooling$rapid, paired = TRUE)
```

```

    Paired t-test

data:  cooling$tunnel and cooling$rapid
t = 1.177, df = 17, p-value = 0.2554
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -0.2285208  0.8051875
sample estimates:
mean difference 
      0.2883333 
```

In practice, you could also run this as a one-sample test on the differences, but that is not what was done here, because the output is definitely not the same, even though the P-value is:

```
with(cooling, t.test(delta, mu = 0))
```

```

	One Sample t-test

data:  delta
t = 1.177, df = 17, p-value = 0.2554
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.2285208  0.8051875
sample estimates:
mean of x 
0.2883333 
```

One point for this kind of answer.

Points:

- 2: One of the first two variations above, correctly coded
- 1: as 2, but with the variables the other way around (on the output, the variables are in the same order as in the input)
- 1: the third variation above (one-sample on differences), correctly coded
- 1: as 2, but omitting the `paired` or with other error
- 0.5: something important correct, but not enough for 1

(17) (2 points) What do you conclude from Figure 11, in the context of the data?

The null hypothesis is that the two cooling methods result in the same mean tenderness, against a two-sided alternative than the mean tenderness values are different. The P-value is 0.2544, so the null hypothesis is not rejected. Therefore there is no evidence of any difference in tenderness between the two methods.

Say what the P-value is, and at least strongly imply that you know what the hypotheses are so that you make the right decision. You could also express everything in terms of differences, since that is what the test is working with behind the scenes.

Points:

- 2: cite P-value, fail to reject null, and conclude no difference in tenderness between two cooling methods
- 1.5: as 2, but without citing P-value
- 0.5: display lack of knowledge of or confusion about hypotheses

- 0.5: get hypotheses right, but draw wrong conclusion

Extra: earlier, I talked about the column `ph`, which was related to the diet the pigs had. Each pig has only one `ph`, because they only had one diet, so an analysis of the effect of `ph` is a two-sample one. Hence this experimental design is *both* matched-pairs *and* two-sample, all at once!

The most obvious way to handle this is to stick with the values `delta` for the response (otherwise you have *two* response variables, which gets you into multivariate methods like MANOVA, which you will see in D29), and then to a two-sample test on the differences:

```
t.test(delta ~ ph, data = cooling)
```

```

	Welch Two Sample t-test

data:  delta by ph
t = -0.69433, df = 14.281, p-value = 0.4986
alternative hypothesis: true difference in means between group high and group low is not eq
95 percent confidence interval:
 -1.4015806  0.7150871
sample estimates:
mean in group high  mean in group low 
        0.07857143         0.42181818 
```

This shows that there is no difference in differences (!) between the two `ph` groups. But this is not really want we wanted to know. The inference seems to be that neither `ph` group shows any difference in terms of cooling method, but it takes a couple of steps to get there:

- overall, there is no difference between cooling methods in terms of tenderness (from before)
- the two `ph` groups don't differ in terms of tenderness difference either
- and hence the two `ph` groups don't differ in terms of actual tenderness either.

This is, as you see, a lot more thinking, and requires the ability to not get fazed by words like "difference between differences".

(18) (2 points) The `rapid` method is both quicker and cheaper to run than the `tunnel` method. Under what circumstances should the `tunnel` method be preferred? Is that the case here? Explain briefly.

The only reason for preferring the `tunnel` method is that it produces better results, ie. that the meat cooled by this method is more tender. Our test shows that that there is no evidence of any difference in tenderness, so there is no reason to prefer the `tunnel` method on the evidence of the data we have.

Points:

- 2: recognizing that the `tunnel` method needs to be better than the `rapid` method to be preferred, and concluding that since it isn't significantly better, we should prefer the `rapid` method (as good, cheaper, quicker).
- 1: as 2 but mistakenly concluding that the `tunnel` method actually *is* better
- 1: concluding that the two methods are equally good, therefore it doesn't matter which one we prefer.

Adhesives

A materials scientist is studying two adhesives. The scientist conducts a small experiment by using each adhesive to glue together a certain material. The material is divided into ten pairs of pieces, and each pair of pieces is glued together using one of the adhesives, chosen at random so that five pairs of pieces are glued with each adhesive. After the adhesives have set, the force required to break each pair of pieces apart is measured. This force is called the "bonding strength" and is measured in Newtons. A larger value means that the adhesive is stronger. The data are shown in Figure 12, with the adhesives labelled as `adhesv.1` and `adhesv.2`. The dataframe is called `adhesives`.

Before looking at the data, the scientist suspects that `adhesv.1` is the stronger of the two adhesives, and would like to see whether there is evidence in favour of this suspicion in the data they collected.

(19) (2 points) The scientist is experienced in assessing strengths of adhesives, and immediately decides to run a Mood's median test rather than a two-sample t-test in order to test their suspicion, even without looking at a graph. Why do you think they might do that?

The reason for telling you that the scientist is experienced is to suggest that they know about the typical distribution of bond strength values. The relevant issue here, in deciding whether to run a two-sample t or Mood's median test, is whether that distribution is normal (enough) or not; apparently their experience suggests that the distribution is not close to normal. The sample size of 5 in each group (pairs of pieces of the material glued together) is very small, so the Central Limit Theorem will be of little help. That is to say:

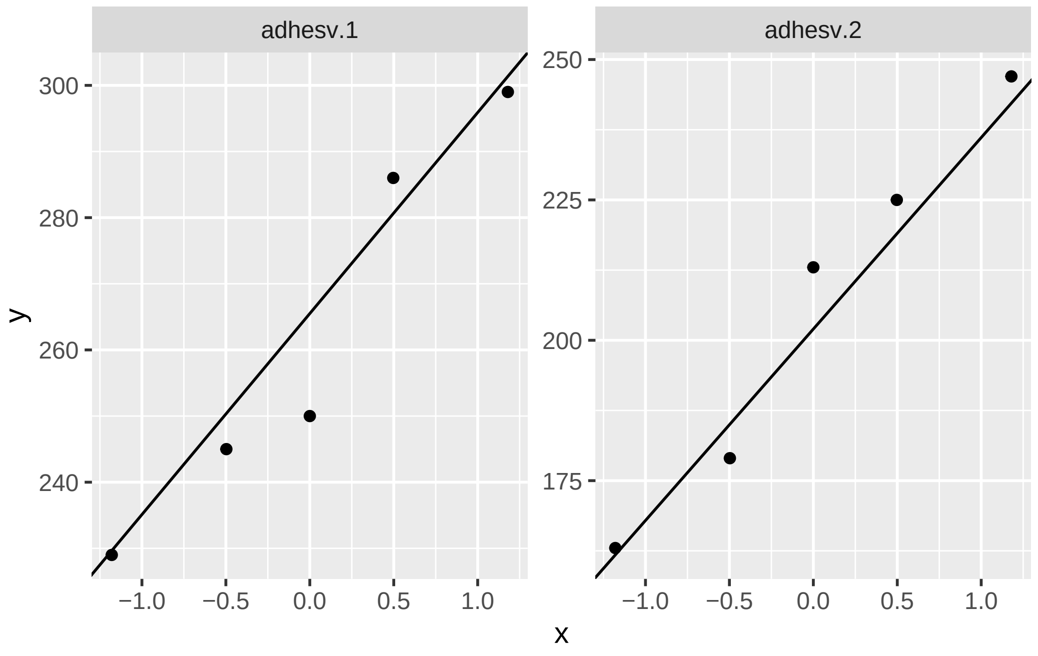
- the scientist *knows* (from past experience) that the bond strengths do not have a normal distribution
- the sample sizes of 5 in each group are small, so the Central Limit Theorem will be of no help.

Points:

- 2: both of the two points listed above, properly made
- 1: as 2 pts, but not a clear enough explanation
- 1: one of the two points above

Extra: with such small samples, a graph won't tell us much anyway. Here's a normal quantile plot:

```
ggplot(adhesives, aes(sample = strength)) + stat_qq() + stat_qq_line() +
  facet_wrap(~ adhesive, scales = "free")
```



The normality actually does not look so bad, but with such small samples it's really very difficult to tell. In circumstances like these, it's better to trust the scientist's broader experience (that bonding strengths are not normally distributed) and pick a test based on that.

(20) (2 points) What code will run a suitable Mood's median test for these data?

This:

```
median_test(adhesives, strength, adhesive)
```

```
$grand_median
[1] 237

$table
          above
group       above below
  adhesv.1     4     1
  adhesv.2     1     4

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

Dataframe, quantitative column, categorical column. No more than that.

Remember that Mood's median test is always two-sided; there is no option to run it one-sided. If you want a one-sided test, then you have to adjust the P-value appropriately (next question). Adding something like `alternative = "greater"` to your code is a (one-point) error.

Points:

- 2: code as above
- 1: as above, but with `alternative = "greater"`, or other error

(21) (3 points) The output from your code of the previous question is shown in Figure 13. Test the scientist's suspicion. What do you conclude, in the context of the data?

The scientist suspects that adhesive `adhesv.1` is stronger (has a larger median bonding strength). This is what they want to prove, so this is a *one*-sided alternative hypothesis, with the null being that the two adhesives have the same median bonding strength.

The problem is that the output is for a two-sided test. To make this a one-sided test, we have to check that we are on the correct side. Look at the table of aboves and belows:

most of the `adhesv.1` bonding strengths are larger than the overall median, and most of the `adhesv.1` bonding strengths are smaller than it. Thus we are on the correct side. (You need to say that we are on the correct side and *how you know*.)

So now, to get a one-sided P-value, we take the two-sided one and halve it:

```
0.0578 / 2
```

```
[1] 0.0289
```

(use your calculator for this and show what you get). This is now less than 0.05, so we *can* conclude that adhesive `adhesv.1` is stronger (in terms of median bonding strength) than `adhesv.2`.

Points:

- 3: proper justification of "correct side" plus halving of P-value plus conclusion in context of data
- 2: as 3, but without proper justification of correct side (halving of P-value without justification)
- 1: two-sided test logically inferring "no difference"
- 0.5: something relevant but not enough for 1 point.

Extra: often, we'll follow up a significant test with a confidence interval, in order to understand how different the medians are. We don't have a formal way to do that: the idea we used for a single median (seeing which medians are not rejected by a sign test) doesn't work here, because we now have *two* medians, and whether Mood's median test rejects depends on what the medians *are*, not just on how different they are.

There is, however, a simulation approach based on the bootstrap:

- take a bootstrap sample from group 1, and work out its median
- take a bootstrap sample from group 2, and work out its median
- take the difference between these two
- repeat many times
- take the middle 95% of the resulting distribution of differences

This is a little bit fiddly to do because of the two samples. There are different ways to handle this, but one way is to put each group in its own dataframe first:

```
adhesives %>% filter(adhesive == "adhesv.1") -> a1
a1
```

adhesive	strength
adhesv.1	229
adhesv.1	286
adhesv.1	245
adhesv.1	299
adhesv.1	250

```
adhesives %>% filter(adhesive == "adhesv.2") -> a2
a2
```

adhesive	strength
adhesv.2	213
adhesv.2	179
adhesv.2	163
adhesv.2	247
adhesv.2	225

Then:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(sample1 = list(sample(a1$strength, replace = TRUE))) %>%
  mutate(sample2 = list(sample(a2$strength, replace = TRUE))) %>%
  mutate(med1 = median(sample1), med2 = median(sample2)) %>%
  mutate(diff = med1 - med2) -> dist_diff
```

and then

```
dist_diff %>% ungroup() %>%
  summarize(lo = quantile(diff, 0.025),
            hi = quantile(diff, 0.975))
```

lo	hi
3	120

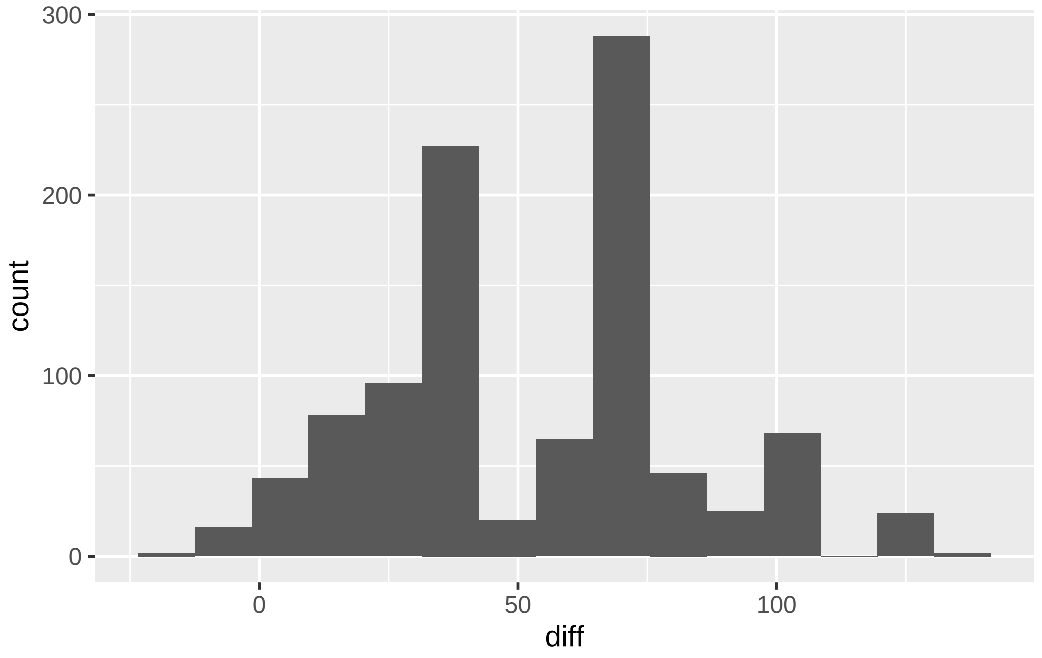
We conclude only that, with 95% confidence, the first adhesive is between 3 and 123 Newtons stronger on average than the second one. The interval contains only positive values (consistent with a conclusion that the first adhesive really is stronger), but it is so long that we know very little about *how much* stronger it is. Perhaps we should have not have expected any better from such small samples.

The `ungroup` is necessary because we previously had a `rowwise`, and the summary we are making here is of the whole dataframe `dist_diff`, so we need to undo the `rowwise` first.

This is a perfectly reasonable interval, although the wisdom is that taking the middle 95% of a bootstrap-flavoured distribution gives you an interval that is too narrow (and, to be more accurate, you should use a "bias-corrected and accelerated" one).

Another thing is that bootstrapping medians tends to give you a weird distribution:

```
ggplot(dist_diff, aes(x = diff)) + geom_histogram(bins = 15)
```



but, despite that, there is not a problem in just grabbing the middle 95% of it. The weirdness is because bootstrap samples from a sample can only contain values from the original sample, and so there is only a small number of possible medians for the bootstrap samples. The mean behaves rather more "smoothly", in that a sample mean does not have to be a sample value, and so bootstrap sampling distributions of sample *means* tend to have a nice regular shape, even if they are not always normal. Medians, not so much.

Carbon monoxide

The carbon monoxide level was measured (in parts per million) at three industrial sites at eight randomly selected times. The data are in Figure 14. The sites are called Site1 through Site3, and the carbon monoxide levels are in column `Monoxide`.

(22) (3 points) What code would run a suitable (ordinary) analysis of variance, and display the results?

This means `aov`, followed by `summary`:

```
carbon.1 <- aov(Monoxide ~ Site, data = carbon)
summary(carbon.1)
```

```
            Df    Sum Sq   Mean Sq F value Pr(>F)
Site         2 0.0004813 2.407e-04   2.411  0.114
Residuals   21 0.0020960 9.981e-05
```

Points:

- 3: as above, or using `tidy` in place of `summary`
- 2: correct `aov` but no or incorrect summary
- 2: as 3, but error in `aov`
- 1: as 2, but error in `aov`
- 1: more than one error, but it is still clear what you are trying to do

(23) (2 points) The output from your code of the previous question is shown in Figure 15. What do you conclude from this Figure, in the context of the data?

The null hypothesis is that all three sites have the same mean carbon monoxide level, and this is not rejected at $\alpha = 0.05$ (because the P-value is 0.114). Hence, our conclusion is that all the sites do have the same mean carbon monoxide level, or (equivalently) that there is no evidence of any differences in mean carbon monoxide level among the sites.

Points:

- 2: cite P-value, and give correct conclusion about carbon monoxide levels at sites
- 1.5: correct conclusion without citing P-value
- 0.5: "the carbon monoxide levels are not all different"
- 0.5: confusion about hypotheses or about interpretation of P-value, including "accept the null" with something else correct
- 0.5: "fail to reject null hypothesis" with no data context

- 0: as 0.5 but "accept the null hypothesis"

(24) (2 points) A graph is shown in Figure 16. What do you conclude from it, and what does it tell you about your previous analysis?

This is a normal quantile plot of the carbon monoxide levels at each site separately.

All three distributions appear to be skewed to the right (I think the curved shape applies to the whole of the distributions; it's not just that the highest values are too high). With the small sample sizes that we have ($n = 8$ in each group), these are not nearly normal enough. Hence, we should not have run the regular ANOVA, or equivalently we should not trust the results from it.

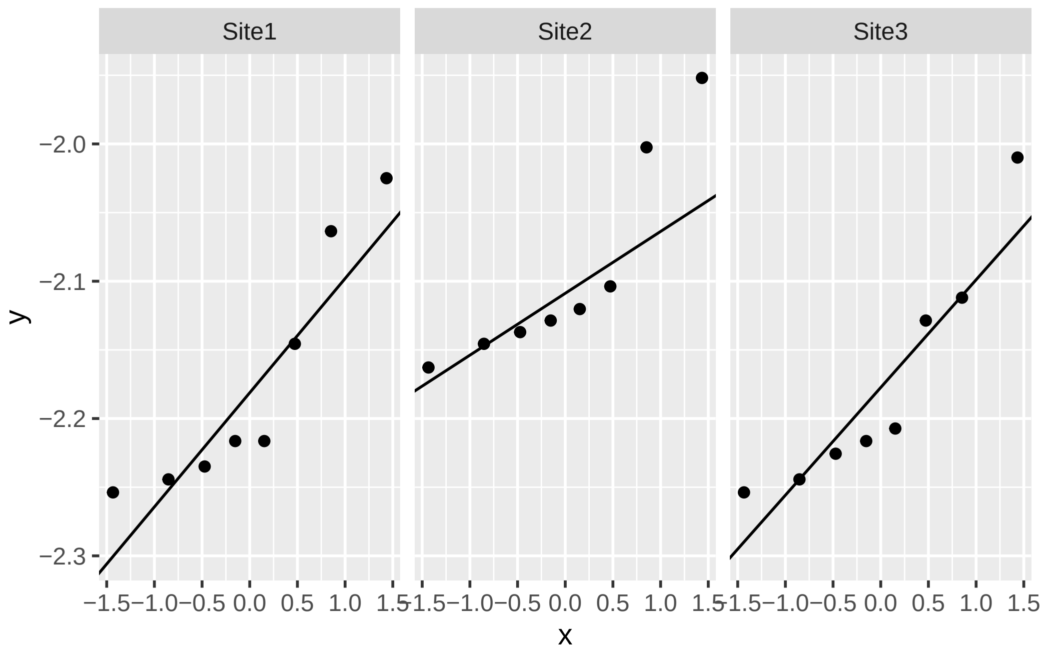
The number of observations *per group* is what matters, not the total number of observations.

Points:

- 2: carbon monoxide levels at each site are skewed to right; small samples not enough to overcome non-normality (or equivalent); should not have run (cannot trust) regular ANOVA
- 1.5: as above, but upper-tail outliers (instead of skewed to right)
- 1: 2-point answer, but missing one of the three things
- 1: arguing that the distributions are not far from normal, and the sample sizes of 8 per group are large enough to overcome the skewness. (I don't think this is the best answer, but there is some credit for it.)
- 0.5: arguing that the distributions are not far from normal, and/or the sample size of 24 is large enough to overcome the non-normality.
- 0.5: one of the three things for 2 points

Extra: all three groups being non-normal in the same way is the kind of situation in which a transformation (of `Monoxide`) may help. The idea is that we want to bring those upper tails down a bit, so something like a log transformation is the kind of thing to try:

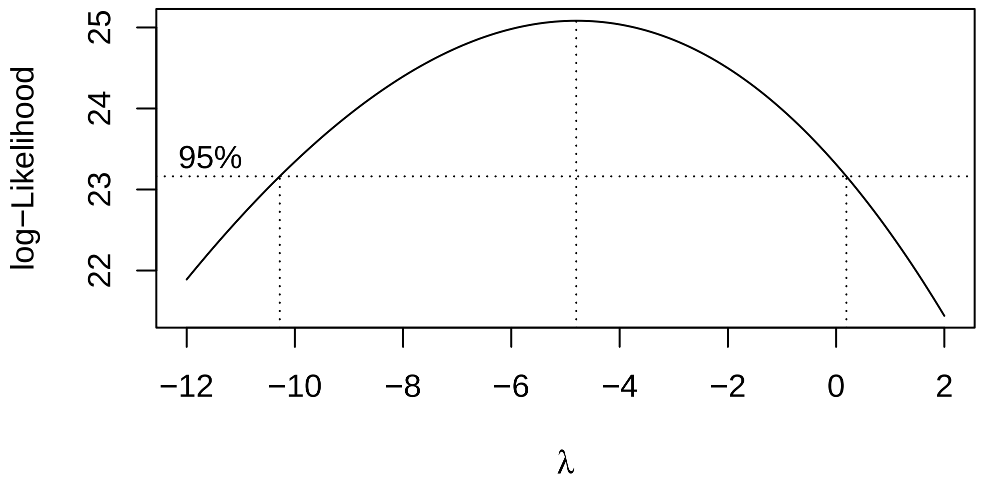
```
ggplot(carbon, aes(sample = log(Monoxide))) + stat_qq() +
  stat_qq_line() + facet_wrap(~ Site)
```



This is still right-skewed (in fact, the picture hasn't changed much apart from the y-axis), so taking logs hasn't gone far enough.

The log and the square root transformations are part of the "ladder of powers", which we learned about in the context of the Box-Cox transformation in regression. In this case, Box-Cox doesn't help at all because there isn't enough data to make a good decision about the transformation to use:

```
boxcox(Monoxide ~ Site, data = carbon, lambda = seq(-12, 2, 0.1))
```



Take a look at the lambda scale!

(25) (2 points) An alternative analysis in shown in Figure 17. What is this analysis, and what do you conclude from it, in the context of the data?

It is a Mood median test.

The P-value is 0.069, which is smaller but not less than 0.05, so we conclude that there are no differences among the *median* carbon monoxide levels at the three sites.

Once again, it is a mistake to go further (in this case, with pairwise median tests), because there are no differences between sites to find.

(The conclusion is the same one of "no differences" that the ANOVA gave us, but now there are no differences among *medians*, so the conclusion is not quite the same.)

Points:

- 2: naming Mood's median test; giving P-value; conclude no differences among *median* carbon monoxide levels at the three sites.
- 1.5: as 2, but not naming the test.
- 1.5: as 2, but not giving P-value.
- 1: as 2, but asserting no difference in *means*
- 1: asserting difference in median but without giving P-value or naming test

- 0.5: something confused but possibly relevant, such as "not all the medians are the same".

Extra: It is better than the one in Figure 15 because we previously concluded that the carbon monoxide levels at each site did not have close enough to a normal distribution. I didn't ask you about this because we already had a question asking something similar. So there are no points here for saying that it is better than a regular ANOVA.

In case you were thinking about pairwise median tests, you shouldn't expect to see much there either (indeed you shouldn't even do them because the overall Mood's median test was not significant):

```
pairwise_median_test(carbon, Monoxide, Site)
```

g1	g2	p_value	adj_p_value
Site1	Site2	0.0455003	0.1365008
Site1	Site3	0.7967624	1.0000000
Site2	Site3	0.3173105	0.9519315

None of the properly adjusted P-values are smaller than 0.10, even, and the only one that even gets close to 0.10 is site 1 vs site 2, which are the two extreme sites on the boxplot.

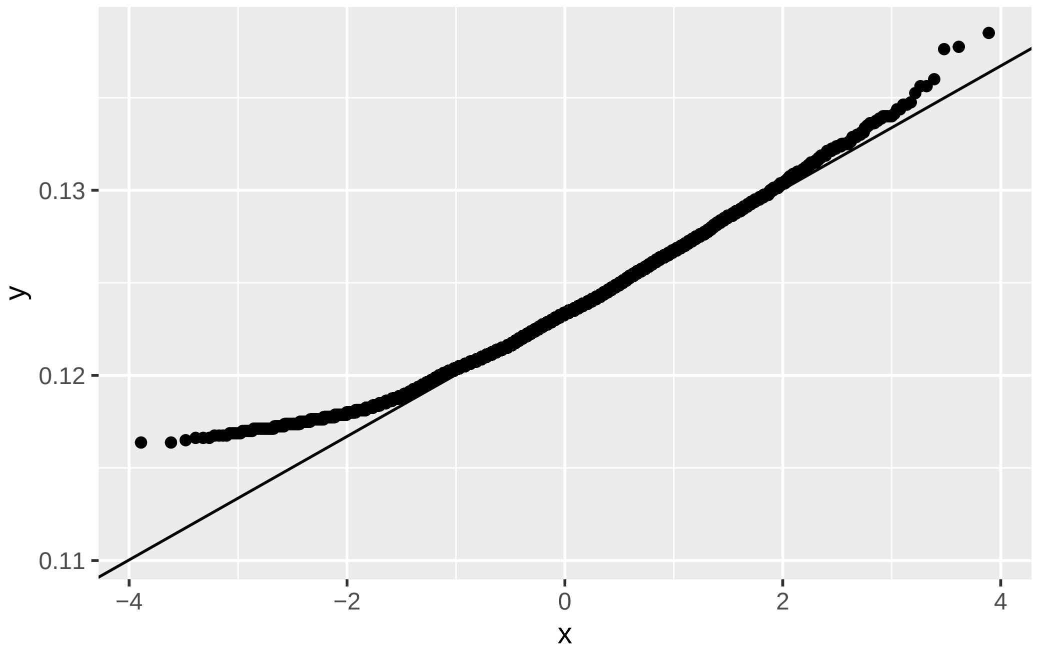
Extra 2: to convince ourselves that the sample size is not big enough to overcome the non-normality (even though the Central Limit Theorem works pretty well in general), we can follow the same strategy as for a two-sample test: find the group that looks worst, and see what its bootstrap sampling distribution of the sample mean looks like. I think Site 2 is the worst, because it seems to have an upper outlier as well as being right-skewed, so let's focus on that:

```
carbon %>% filter(Site == "Site2") -> site2
site2
```

	Monoxide	Site
9	0.122	Site2
10	0.119	Site2
11	0.115	Site2
12	0.120	Site2
13	0.117	Site2

	Monoxide	Site
14	0.135	Site2
15	0.118	Site2
16	0.142	Site2

```
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(site2$Monoxide, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(sample = my_mean)) + stat_qq() + stat_qq_line()
```



This is definitely not normal. It is still clearly right-skewed (so that the sample size of $n = 8$ is definitely not big enough to overcome the non-normality), but you might be surprised to see that the problem is not that the upper tail is too long, but that the lower tail is too *short*.

In any case, though, the ANOVA and Mood's median test have P-values in the same ballpark, and lead to the same conclusion of no differences.

(26) (2 points) A colleague says to you "your analysis must be wrong, because obviously Site 2 has higher carbon monoxide levels on average". How do you react to this?

Our analysis says that there are no differences in mean (or median) carbon monoxide levels among the three sites, and thus that any differences we observed, such as the higher values for Site 2, are just chance, as inferred from the data we have. There is no evidence that Site 2 is *actually* higher in terms of mean carbon monoxide than the others, in the sense that if you were to take more observations from the three sites, you would consistently see Site 2 coming out highest.

Some kind of discussion of the difference between samples and populations is called for here.

Points:

- 2: a complete discussion of the relevant issues (Site 2 is higher than the others by chance, or just because it's higher for these data doesn't mean that it will always be higher)
- 1: something relevant, but not a complete answer

Brushing teeth

A study was carried out to compare the effectiveness of brushing with two types of toothbrushes. The toothbrushes were called Conventional and Hugger. Of the subjects, 14 identified as female and 12 as male. Each subject, in the full study, used both types of toothbrush (at different times), and had their dental plaque index measured both before and after brushing. For the questions below, give code to tidy the input dataframe (displayed in a Figure) into the form requested.

(27) (2 points) The dataframe `tb1` shown in Figure 18 shows the plaque index scores after brushing for each subject using each of the two toothbrushes. Arrange these data to have one column of plaque index scores called `After`, and a second column called `toothbrush` showing which toothbrush was used.

This is a standard `pivot_longer`. I gave you the names for the new columns, so make sure you use them:

```
tb1 %>% pivot_longer(-Subject, names_to = "toothbrush", values_to = "After")
```

Subject	toothbrush	After
1	Hugger	0.43
1	Conventional	0.75
2	Hugger	0.08

Subject	toothbrush	After
2	Conventional	0.55
3	Hugger	0.18
3	Conventional	0.08
4	Hugger	0.78
4	Conventional	0.75
5	Hugger	0.03
5	Conventional	0.05
6	Hugger	0.23
6	Conventional	1.60
7	Hugger	0.20
7	Conventional	0.65
8	Hugger	0.00
8	Conventional	0.13
9	Hugger	0.05
9	Conventional	0.83
10	Hugger	0.30
10	Conventional	0.58
11	Hugger	0.33
11	Conventional	0.38
12	Hugger	0.00
12	Conventional	0.63
13	Hugger	0.90
13	Conventional	0.25
14	Hugger	0.24
14	Conventional	1.03
15	Hugger	0.15
15	Conventional	1.58
16	Hugger	0.10
16	Conventional	0.20
17	Hugger	0.33
17	Conventional	1.88
18	Hugger	0.33
18	Conventional	2.00
19	Hugger	0.53
19	Conventional	0.25
20	Hugger	0.43
20	Conventional	0.18
21	Hugger	0.65
21	Conventional	0.85

Subject	toothbrush	After
22	Hugger	0.20
22	Conventional	1.15
23	Hugger	0.25
23	Conventional	0.93
24	Hugger	0.15
24	Conventional	1.05
25	Hugger	0.05
25	Conventional	0.85
26	Hugger	0.25
26	Conventional	0.88

Any way of selecting the columns to pivot longer is fine; I happen to think that "everything but Subject" is the most concise. Here and elsewhere, you may split your lines of code anywhere that conveys the the idea that the line of code continues, such as after the commas. (Sometimes these `pivot_longer`s get long and you won't be able or want to keep all of it on one line.) On an exam, you can also draw an arrow connecting one piece of the code to the next piece and label it "keep on same line" or similar.

Points:

- 2: correct pivot-longer
- 1: any errors that will stop it from working, as long as something non-trivial is correct.

(28) (3 points) The dataframe in Figure 19 shows the plaque index scores after brushing for each combination of toothbrush and gender. Arrange these data to be suitable for an analysis of variance with all the plaque index scores in one column called `After`, and with separate columns indicating each subject's `toothbrush` and `gender`.

This is best done as the variation on pivot-longer with *two* columns in the `names_to`, which are implied (by my font choice in the question) to be called `toothbrush` and `gender` respectively.

```
tb2 %>% pivot_longer(-Subject, names_to = c("toothbrush", "gender"),
                     names_sep = ":",
                     values_to = "After")
```

Subject	toothbrush	gender	After
1	Hugger	F	0.43
1	Hugger	M	0.15
1	Conventional	F	0.75
1	Conventional	M	1.58
2	Hugger	F	0.08
2	Hugger	M	0.10
2	Conventional	F	0.55
2	Conventional	M	0.20
3	Hugger	F	0.18
3	Hugger	M	0.33
3	Conventional	F	0.08
3	Conventional	M	1.88
4	Hugger	F	0.78
4	Hugger	M	0.33
4	Conventional	F	0.75
4	Conventional	M	2.00
5	Hugger	F	0.03
5	Hugger	M	0.53
5	Conventional	F	0.05
5	Conventional	M	0.25
6	Hugger	F	0.23
6	Hugger	M	0.43
6	Conventional	F	1.60
6	Conventional	M	0.18
7	Hugger	F	0.20
7	Hugger	M	0.65
7	Conventional	F	0.65
7	Conventional	M	0.85
8	Hugger	F	0.00
8	Hugger	M	0.20
8	Conventional	F	0.13
8	Conventional	M	1.15
9	Hugger	F	0.05
9	Hugger	M	0.25
9	Conventional	F	0.83
9	Conventional	M	0.93
10	Hugger	F	0.30
10	Hugger	M	0.15
10	Conventional	F	0.58

Subject	toothbrush	gender	After
10	Conventional	M	1.05
11	Hugger	F	0.33
11	Hugger	M	0.05
11	Conventional	F	0.38
11	Conventional	M	0.85
12	Hugger	F	0.00
12	Hugger	M	0.25
12	Conventional	F	0.63
12	Conventional	M	0.88
13	Hugger	F	0.90
13	Hugger	M	NA
13	Conventional	F	0.25
13	Conventional	M	NA
14	Hugger	F	0.24
14	Hugger	M	NA
14	Conventional	F	1.03
14	Conventional	M	NA

If you don't see that, you have a two-point option, which is an ordinary pivot-longer followed by a `separate_wider_delim`. Use a name of your choice at the first stage:

```
tb2 %>% pivot_longer(-Subject, names_to = "treatment", values_to = "After") %>%
  separate_wider_delim(treatment, ":", names = c("toothbrush", "gender"))
```

Subject	toothbrush	gender	After
1	Hugger	F	0.43
1	Hugger	M	0.15
1	Conventional	F	0.75
1	Conventional	M	1.58
2	Hugger	F	0.08
2	Hugger	M	0.10
2	Conventional	F	0.55
2	Conventional	M	0.20
3	Hugger	F	0.18
3	Hugger	M	0.33
3	Conventional	F	0.08
3	Conventional	M	1.88

Subject	toothbrush	gender	After
4	Hugger	F	0.78
4	Hugger	M	0.33
4	Conventional	F	0.75
4	Conventional	M	2.00
5	Hugger	F	0.03
5	Hugger	M	0.53
5	Conventional	F	0.05
5	Conventional	M	0.25
6	Hugger	F	0.23
6	Hugger	M	0.43
6	Conventional	F	1.60
6	Conventional	M	0.18
7	Hugger	F	0.20
7	Hugger	M	0.65
7	Conventional	F	0.65
7	Conventional	M	0.85
8	Hugger	F	0.00
8	Hugger	M	0.20
8	Conventional	F	0.13
8	Conventional	M	1.15
9	Hugger	F	0.05
9	Hugger	M	0.25
9	Conventional	F	0.83
9	Conventional	M	0.93
10	Hugger	F	0.30
10	Hugger	M	0.15
10	Conventional	F	0.58
10	Conventional	M	1.05
11	Hugger	F	0.33
11	Hugger	M	0.05
11	Conventional	F	0.38
11	Conventional	M	0.85
12	Hugger	F	0.00
12	Hugger	M	0.25
12	Conventional	F	0.63
12	Conventional	M	0.88
13	Hugger	F	0.90
13	Hugger	M	NA
13	Conventional	F	0.25

Subject	toothbrush	gender	After
13	Conventional	M	NA
14	Hugger	F	0.24
14	Hugger	M	NA
14	Conventional	F	1.03
14	Conventional	M	NA

Points:

- 3: pivot-longer with two columns in `names_to`, correctly coded
- 2.5: as 3 pts but one small error, such as missing the `names_sep`
- 2: ordinary pivot-longer plus `separate_wider_delim`, correctly coded
- 2: as 3 pts, but more than one error such that it is still obvious that this is what you were trying to do
- 1.5: as 2, but one small error, such as missing the `names_sep`
- 1: something important correct but not enough for more points such as ordinary pivot-longer without the `separate_wider_delim`
- 0.5: something potentially relevant.

(29) (2 points) (No code required here, just explanation.) The dataframe shown in Figure 19 has some missing values in it (indicated by `NA`). Do these indicate genuinely missing data? Explain briefly.

Go back to the original description of the data: there were 14 females in the study and only 12 males, so that Figure 18 correctly has $14 + 12 = 26$ rows. In Figure 19, there were no males numbered 13 or 14, so there is no plaque-index-after-brushing value possible to go here. These are therefore not actual missing data, but just a consequence of the way the data are laid out. Sometimes these are called "structural missing values": they are missing because there *cannot* be any data there, not because there should have been some but it happened not to be recorded.

Points:

- 2: clear recognition that the missings were caused by different numbers of males and females and are not actual missing data
- 1: some reasonable ideas but not enough for 2 points

Heights and weights

Parents and doctors often use growth charts to evaluate how a child is developing. These charts show an average value for height and weight at different ages as well as a "normal range" of height and weight at that age. We will be concerned with the average values for height (in inches) and weight (in pounds) for children identified as male and female, aged from 13 to 15 years. These values are shown in dataframe `hw1` in Figure 20, and in a different format as dataframe `hw2` in Figure 21.

(30) (3 points) What code would convert dataframe `hw1` into dataframe `hw2`?

This is a pivot-longer where we have to create new columns using `.value`:

```
hw1 %>% pivot_longer(-gender,
                     names_to = c(".value", "age"),
                     names_sep = ":") -> hw2
hw2
```

gender	age	ht	wt
female	13	101.0	61.7
female	14	105.0	62.5
female	15	115.0	62.9
male	13	100.0	61.5
male	14	112.0	64.5
male	15	123.5	67.0

It is enough to show the code that does the pivoting. You don't need to save and display the result. This is the best answer.

A second-best answer is to pivot-longer the normal way and think about what you'll have:

```
hw1 %>% pivot_longer(-gender,
                     names_to = "name",
                     values_to = "value")
```

gender	name	value
female	ht:13	101.0
female	wt:13	61.7

gender	name	value
female	ht:14	105.0
female	wt:14	62.5
female	ht:15	115.0
female	wt:15	62.9
male	ht:13	100.0
male	wt:13	61.5
male	ht:14	112.0
male	wt:14	64.5
male	ht:15	123.5
male	wt:15	67.0

The next step here is a `separate_wider_delim`, to sort out the column I called `name`:

```
hw1 %>% pivot_longer(-gender,
                     names_to = "name",
                     values_to = "value") %>%
  separate_wider_delim(name, ":",
                       names = c("stat", "age"))
```

gender	stat	age	value
female	ht	13	101.0
female	wt	13	61.7
female	ht	14	105.0
female	wt	14	62.5
female	ht	15	115.0
female	wt	15	62.9
male	ht	13	100.0
male	wt	13	61.5
male	ht	14	112.0
male	wt	14	64.5
male	ht	15	123.5
male	wt	15	67.0

Notice the difficulty I am having in finding names for things, always a warning sign.

Finally, we want to pivot-wider the column I called `stat`, carrying along the values in `value`:

```
hw1 %>% pivot_longer(-gender,
                     names_to = "name",
                     values_to = "value") %>%
  separate_wider_delim(name, ":",
                       names = c("stat", "age")) %>%
  pivot_wider(names_from = stat,
              values_from = value)
```

gender	age	ht	wt
female	13	101.0	61.7
female	14	105.0	62.5
female	15	115.0	62.9
male	13	100.0	61.5
male	14	112.0	64.5
male	15	123.5	67.0

and we are there, though after a lot of work. An alternative version of this skips the `separate_wider_delim`, but it is still necessary (this way) to pivot-wider after pivoting longer:

```
hw1 %>% pivot_longer(-gender,
                     names_to = c("stat", "age"),
                     values_to = "value",
                     names_sep = ":") %>%
  pivot_wider(names_from = stat,
              values_from = value)
```

gender	age	ht	wt
female	13	101.0	61.7
female	14	105.0	62.5
female	15	115.0	62.9
male	13	100.0	61.5
male	14	112.0	64.5
male	15	123.5	67.0

Either of these two ways is confusing because you have to keep straight the `age` and the thing I called `value`, which is either a height or a weight.

Points:

- 3 for the first way, correctly coded.
- 2 for the second way, correctly coded.
- 2 also for the first way with enough correct to show that you have tried to make it work (for example, missing the `names_sep = ":"`).
- 1.5 for the second way with only one error (for example, missing the `names_sep = ":"`).
- 1 for the second way with more than one error (but something correct).
- 1 for the first way with less than half of it correct, in the grader's estimation.

The grader can also award up to 1 if it is not clear which way you were trying to go, but something relevant is correct.

(31) (2 points) What code would start from dataframe `hw2` in Figure 21 and convert it into dataframe `hw1` in Figure 20? It is enough to produce a dataframe with the same columns and values as `hw1`, but in a different order.

The previous one was a pivot-longer, so this must be a pivot-wider. We have to produce two columns of values, though, so we need to put the two columns *both* in `values_from`:

```
hw2 %>% pivot_wider(names_from = age,
                    values_from = c(ht, wt),
                    names_sep = ":")
```

gender	ht:13	ht:14	ht:15	wt:13	wt:14	wt:15
female	101	105	115.0	61.7	62.5	62.9
male	100	112	123.5	61.5	64.5	67.0

This is the same as `hw1`, albeit with the columns in a different order (which is not a concern).

You have seen one of these in Worksheet 9; if you didn't notice it there, I think it's reasonable to be able to guess that this code will do something like the right thing. (The deal is that instead of making columns with names `13` through `15`, it will glue what they are values of *at* ages 13 through 15 onto the front of the number, with the column names we are making separated by a colon.)

Something analogous to the previous question will also work, but not as concisely (three steps rather than the one you could have done it in):

```
hw2 %>% pivot_longer(ht:wt, names_to = "statname", values_to = "stat") %>%
  unite(combo, statname, age, sep = ":") %>%
  pivot_wider(names_from = combo, values_from = stat)
```

gender	ht:13	wt:13	ht:14	wt:14	ht:15	wt:15
female	101	61.7	105	62.5	115.0	62.9
male	100	61.5	112	64.5	123.5	67.0

Points:

- 2 for the first one-step way, done correctly
- 1 for the second way, done correctly
- 1 for the first way with errors, but with enough correct to make it clear that the first way was being attempted
- 0.5 for the second way with errors, but with something correct.

The grader should not penalize for omitting the `:` in `names_sep` (or `sep`) *if this was already penalized in the previous question.*

Extra: I don't know about you, but the only way I could make my second method here work is to run each line and see what it did, and then decide what to do next. On an exam, you will have to imagine (or sketch out) what you have to do to make it work. For those of a mathematical bent, this actually is the inverse of the second method in the previous question, but you have to read up from the bottom in one of them:

- the `pivot_longer` here is the inverse of the *last* `pivot_wider` in the previous question
- the `unite` here is the inverse of the `separate_wider_delim` previously
- the `pivot_wider` here is the inverse of the *first* `pivot_longer` previously.

Rainfall and growing corn

Six US states have for many years grown large amounts of corn (Iowa, Nebraska, Illinois, Indiana, Missouri, Ohio). Data on the average corn yield in these states was recorded for each year from 1890 to 1927, along with the average rainfall in those same states in the same years. The data are shown in Figure 22, in dataframe `corn`. Note: corn yield is measured in bushels per acre: it is the amount of corn harvested divided by the area of land on which the corn is grown.

(32) (2 points) A scatterplot of corn yield against rainfall is shown in Figure 23. Why does it seem sensible to add a squared term in rainfall to a regression predicting the corn yield?

The trend seems not to be linear: it increases and then levels off or starts to decrease. Therefore we can add rainfall-squared to account for the curve.

Points:

- 2 for the above or equivalent
- 1 for saying "not linear" without saying how that implies to add a squared term.

(33) (2 points) A regression was fitted, as shown in Figure 24. According to this output, was it a good idea to include rainfall squared in the regression? Explain briefly.

The squared term is significantly different from zero: the P-value is 0.014, less than 0.05, and so the squared term is worth keeping in the regression.

Do not be tempted by the small R-squared; the fit, even now, is not very good, but the P-value says that the R-squared for the regression without the squared term would be *even less*. If you had the output from a regression without rainfall squared, you could compare the R-squared values and note that even though the one here is small, it is clearly bigger than when you omit rainfall squared. But you don't have that, so you cannot make that comparison.

Points:

- 2 for "significantly different from zero" with correct P-value (the one for the squared term)
- 1 for "the squared term is significantly different from zero" without giving the correct P-value.
- 0.5 at the grader's discretion if something relevant is said without being enough for 1 point.

(34) (2 points) Based on what you know or can guess about growing crops on farms and the effect of rainfall, why does it make sense that the estimate of the squared term has the sign (positive or negative) that it does? Explain briefly.

The Estimate is -0.23, negative. This means that the fitted parabola has a maximum (it opens downwards), something you might also guess from Figure 23. One point for making the connection between the Estimate (negative) and the shape of the fitted curve. (We talked about this in lecture because there was a student question about it.) I was also prepared to consider an answer like "predicted yield increases, but as rainfall gets large, the predicted yield decreases" on the grounds that the Estimate for rainfall is positive but the Estimate for rainfall-squared is negative.

As to why it makes sense: things grown on farms (like corn) need rain to grow. Up to a point, the more rain the better, but too much rain can be bad for crop growth. This suggests that there will be an optimal amount of rain; less or more than that will be associated with a smaller yield. This is consistent with the shape of relationship we saw (a parabola with a maximum).

The second point for saying why the shape of the fitted curve makes practical sense: there is something like a "best" amount of rainfall, and too much is bad. This article gives a number of reasons for why that is.

Points:

- 2 for my answer above (negative, parabola shape, why it makes sense) or equivalent
- 1 for "negative" and what it implies for parabola shape.
- 1 for asserting that there is an optimal amount of rainfall without linking it to the Estimate (this includes a claim that the negative Estimate implies this without linking to parabola shape)
- 0.5 for stating that the Estimate is negative without getting further (or saying something like "more rainfall is worse").

(35) (2 points) A plot is shown in Figure 25. Does this plot indicate any problems with the regression? Explain briefly.

This is a plot of residuals against fitted values. I would say that there is no pattern here (if you know the fitted value, it tells you nothing about the residual), and therefore that the regression is satisfactory.

You might see a little "fanning-in" here, but the evidence for this is weak, largely in the most positive and most negative residuals happening to have a smallish fitted value, but a conclusion based on only two points out of 37 is not a reliable one. One point for claiming fanning-in with a good enough explanation of what that means.

Points:

- 2 for "no pattern" and therefore regression is satisfactory
- 1 for "no pattern" without talking about problems with regression
- 1 for "fanning in" and therefore regression is not satisfactory
- 0.5 for anything else possibly relevant. "The points are mostly on the right" is not relevant here.

(36) (2 points) Another plot is shown in Figure 26. What do you learn from this plot? Explain briefly.

This is a plot of the residuals (from the regression with rainfall and rainfall squared) against year, a variable that was not in the original regression. (This is therefore a plot of residuals against an explanatory variable.) This plot is showing a (mostly) upward trend with year, which indicates that year should be included in the regression. The ideal would be a random plot again, but if we see a trend, that should be incorporated into the regression.

I added the smooth trend to help you see the upward trend of the points. Don't be deceived by the dip at the end; this is almost entirely based on there being one very negative residual in 1924 despite the overall upward trend.

Points:

- 2 for "upward trend" and "should add `Year` to regression"
- 1.5 for "upward trend with curve" ditto
- 1 for "upward trend" or "upward trend with curve" without saying what we should do
- 0.5 for a different trend but a consistent indication of what we should do, such as "no trend, therefore we were correct to leave `Year` out of the regression".

(37) (2 points) A second regression is shown in Figure 27. (This regression may or may not be suggested by your previous work.) Interpret the Estimate for `Year`.

The Estimate for Year is 0.136. This says that as year increases by one, predicted corn yield increases by 0.136 bushels per acre *even if rainfall stays the same*. (You can say "holding rainfall constant" if you want, but you cannot control rainfall, so it makes less sense to pretend you can). One point for describing the effect of being a year later on the corn yield, and one point for saying that this is if rainfall stays the same.

This question has nothing to do with the P-value. It is true that there is a significant effect of `Year`: that is, it was worthwhile to add `Year` to the regression, which means that the Estimate is significantly different from zero and therefore its value is worth interpreting, but that part of the logic is not what I am asking about here.

Points:

- 2 for "if year increases by 1, Yield predicted to increase by 0.14, rainfall same". I would also accept "holding rainfall constant" or "all else equal" or equivalent, though "if rainfall stays same" is better.
- 1 for "if year increases by 1, Yield predicted to increase by 0.14" without talking about rainfall or requiring other explanatory variables to be the same.

Extra: this predicted increase in corn yield over time suggests that there are something like improvements in technology happening, that allow farmers to grow more corn on the same amount of land in later years compared to earlier ones even if the amount of rainfall is the same. This is an example of "ceteris paribus" (see the regression worksheet) again: the multiple regression has allowed us to disentangle the effect of year while accounting for rainfall, and also the effect of rainfall while accounting for year. (The Estimate for rainfall-squared is still negative, so even after you account for year, there is still an optimal amount of rainfall for growing corn.)

Cylinder

A cylinder with radius r and height h is shown in Figure 28. It has volume $\pi r^2 h$ and total surface area $2\pi r(r+h)$. (The total surface area is of the round sides of the cylinder *and* the two ends.)

(38) (4 points) Write an R function called `cylinder` that accepts any input radius `r` and height `h` (in that order), and returns the volume and total surface area of a cylinder of that radius and height, as a vector containing those two values in that order. For full credit, write your function so that it is clear what it is doing. Hint: R has a variable called `pi` that contains the value of π.

Steps:

- the header line
- calculate the volume, labelling it as such
- calculate the surface area, labelling it as such
- return a vector containing these two values.

```
cylinder <- function(r, h) {
  volume <- pi * r^2 * h
  surface_area <- 2 * pi * r * (r+h)
  c(volume, surface_area)
}
```

Give the intermediate results names that indicate what they are, and use `c` to collect them together into a vector to be returned. Remember that R evaluates the last line of a function and returns that; using a `return` in this context is poor style. Also, remember that you need a `*` to multiply numbers in R, and you need extra brackets for the last term of the surface area formula so that the addition is done first.

You could (in theory) write your function like this:

```
cylinder0 <- function(r, h) {
  c(pi * r^2 * h, 2 * pi * r * (r+h))
}
```

but this is also poor style, because unless your reader happens to have the formulas for the volume and surface area of a cylinder in their head already, it is impossible for them (or for you in six months) to understand what this function is doing. It is even much more difficult for the grader to check: did you get both those formulas right, and get them in the right order?

Points:

- 4: function with correct header line, lines calculating volume and surface area named appropriately, and line calculating and returning the volume and surface area.
- 3.5: the above, but with one small error or unclear names of results
- 3: 4-point answer with missing multiplication signs (more than one), or missing brackets around `(r+h)`
- 3: not calculating volume and surface area on separate lines
- 3: using a `return` statement
- 2: a 3-point answer with additional errors
- 2: 4-point answer with 2 or more errors
- 1: something relevant correct (which could be the top line of the function).

Points are based on: minus one for each of the "poor style" things, minus a half or full point for any additional errors (according to the grader's estimation of how serious they are). Minus a half for a "typo" (such as forgetting one of the multiplication signs), minus one for missing all of the multiplication signs or the brackets around `r+h`.

(39) (2 points) How would you use your function to find the volume and surface area of a cylinder with height 10 cm and radius 3 cm? (I don't need the answer, just the code that would work out the answer.)

Call the function with the right values in the right order (r first):

```
cylinder(3, 10)
```

```
[1] 282.7433 245.0442
```

This is meant to be easy (and gettable if you know nothing else about functions).

Points:

- 2: as above, or with the inputs named (in which case they can be in either order)
- 1: an error, for example the unnamed inputs in the wrong order
- 0: running the code contained within your function. The point is to run *the function*.

Extra: amusingly, you can work out the volumes and surface areas of several cylinders at once. Feed your function vectors for `r` and `h` (of the same length):

```
cylinder(c(3,5), c(20, 10))
```

```
[1] 565.4867 785.3982 433.5398 471.2389
```

These are (respectively) the volumes of the two cylinders, and surface areas of the two cylinders, one of radius 3 and height 20, and one of radius 5 and height 10. (Yes, it also surprised me that they came out in that order.) To check that:

```
cylinder(3, 20)
```

```
[1] 565.4867 433.5398
```

```
cylinder(5, 10)
```

```
[1] 785.3982 471.2389
```

The volume and surface area both depend on r^2 but just h, so the "squatter" cylinder with a smaller height but bigger radius actually turns out to have both a bigger volume and a bigger surface area.

If you were going to use the function like this, it would be better to at least name the outputs so that it would be clearer what they are. One way to do this (beyond the scope of this question) is to return a *dataframe* with columns called `volume` and `surface`, for example. That would mean modifying the function like this:

```
cylinder_v <- function(r, h) {
  volume <- pi * r^2 * h
  surface_area <- 2 * pi * r * (r+h)
  tibble(volume = volume, surface = surface_area)
}
```

which would give this:

```
cylinder_v(3, 20)
```

volume	surface
565.4867	433.5398

```
cylinder_v(5, 10)
```

volume	surface
785.3982	471.2389

or, to do both at once, which actually works smoothly and puts the right numbers in the right places:

```
cylinder_v(c(3,5), c(20, 10))
```

volume	surface
565.4867	433.5398
785.3982	471.2389

(40) (2 points) Suppose you run `cylinder(h = 10, r = 3)`. Would you get the same answer as in the previous question, or a different one? Explain briefly.

The inputs are the wrong way around, but they are *named*, so the values will get matched up with the right inputs and so you will get the same answer as in the previous question:

```
cylinder(h = 10, r = 3)
```

```
[1] 282.7433 245.0442
```

If you didn't remember this, you can still infer that the function knows that 10 is `h` and 3 is `r`, and will use them as such. This is also a good explanation.

Points:

- 2: "You will get the same answer because the inputs are named" at least.

- 1: "The same answer" but an incomplete explanation of why
- 0: Any answer without explanation, or an assertion that the answer would be different.

If you claim that this is your answer to the previous question, you need to explain here why it works.

(41) (2 points) I changed part of the top line of my function to read `(r, h = 20)`. What will happen if I run `cylinder(5)`? Will it give an error, or will it run? Explain briefly.

Here's my new function:

```r
cylinder <- function(r, h = 20) {
  volume <- pi * r^2 * h
  surface_area <- 2 * pi * r * (r+h)
  c(volume, surface_area)
}
```

This uses the value 20 for `h` as a default, to be used if you do not specify a value for `h`.

Hence, calling it like this:

```r
cylinder(5)
```

```
[1] 1570.7963  785.3982
```

gives the same answer as

```r
cylinder(5, 20)
```

```
[1] 1570.7963  785.3982
```

So it will work, and the above shows what it will do.

Points:

- 2: it will run, using the value 20 for `h`
- 1: it will run, but with incomplete explanation of why (this includes using `h = 20` as your explanation, because this does not show understanding of what the code means)
- 0: no explanation, or assertion that it will give an error.

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

Figures

```r
library(tidyverse)
library(smmr)
library(broom)
```

Figure 1: Packages loaded

```
name                                csd_type        population
Armstrong                           township              1199
Haldimand                           city                 49216
North_Stormont                      township              7400
Temagami                            municipality           862
Oliver_Paipoonge                    municipality          6035
Central_Huron                       municipality          7799
Cornwall                            city                 47845
Head-Clara-Maria                    township               267
Nipigon                             township              1473
Highlands_East                      municipality          3830
James                               township               348
Richmond_Hill                       city                202022
Mattawa                             town                  1881
Blandford-Blenheim                  township              7565
Gauthier                            township               151
Edwardsburgh/Cardinal               township              7505
Pelee                               township               230
Enniskillen                         township              2825
North_Shore                         township               531
Brooke-Alvinston                    municipality          2359
```

Figure 2: Ontario municipalities, 20 randomly chosen rows of data file

```
     One-sample t test power calculation 

              n = 16
          delta = 0.1
             sd = 0.2
      sig.level = 0.05
          power = 0.4648089
    alternative = two.sided
```

Figure 3: Power of t-test for silicon dioxide percentage

```
     One-sample t test power calculation 

              n = 66.39753
          delta = 0.1
             sd = 0.2
      sig.level = 0.05
          power = 0.98
    alternative = two.sided
```

Figure 4: Output for estimating sample size for silicon dioxide experiment

```
cars
```

diagnostic_time
30.6
30.1
15.6
26.7
27.1
25.4
35.0
30.8
31.9
53.2
12.5
23.2
8.8
24.9
30.2

Figure 5: Mechanic diagnostic times

```
ggplot(cars, aes(x = diagnostic_time)) + geom_histogram(bins = 7)
```

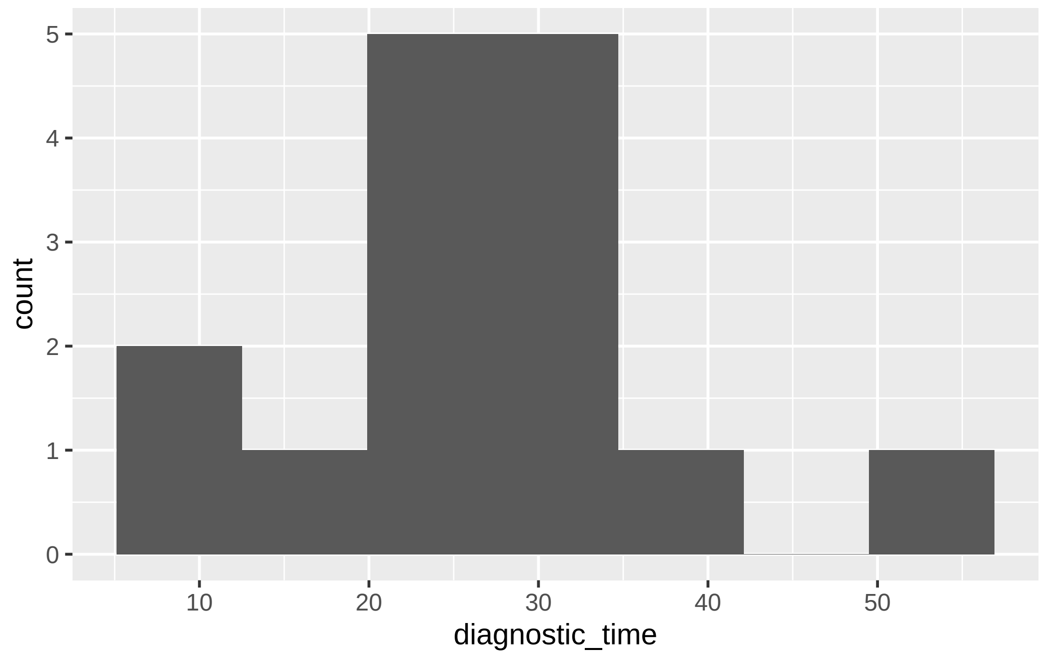


Figure 6: Mechanic diagnostic times: graph

```
$above_below
below above
   13     1

$p_values
  alternative      p_value
1       lower 0.0009155273
2       upper 0.9999389648
3   two-sided 0.0018310547
```

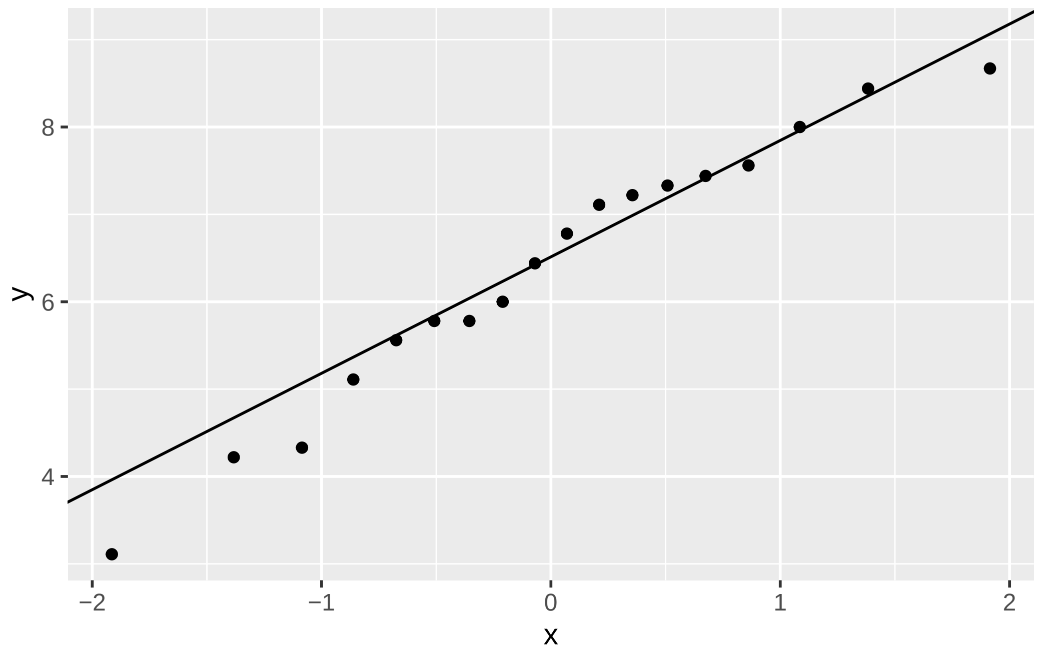
Figure 7: Mechanic diagnostic times: test

```
cooling
```

pig	ph	tunnel	rapid	delta
73	high	8.44	8.44	0.00
74	high	7.11	6.00	1.11
75	high	6.00	5.78	0.22
76	high	7.56	7.67	-0.11
77	low	7.22	5.56	1.66
78	high	5.11	4.56	0.55
79	low	3.11	3.33	-0.22
80	high	8.67	8.00	0.67
81	low	7.44	7.00	0.44
82	low	4.33	4.89	-0.56
83	low	6.78	6.56	0.22
84	low	5.56	5.67	-0.11
85	low	7.33	6.33	1.00
86	low	4.22	5.67	-1.45
87	high	5.78	7.67	-1.89
94	low	5.78	5.56	0.22
95	low	6.44	5.67	0.77
96	low	8.00	5.33	2.67

Figure 8: Pork cooling data

```
ggplot(cooling, aes(sample = tunnel)) + stat_qq() + stat_qq_line()
```



```
ggplot(cooling, aes(sample = rapid)) + stat_qq() + stat_qq_line()
```

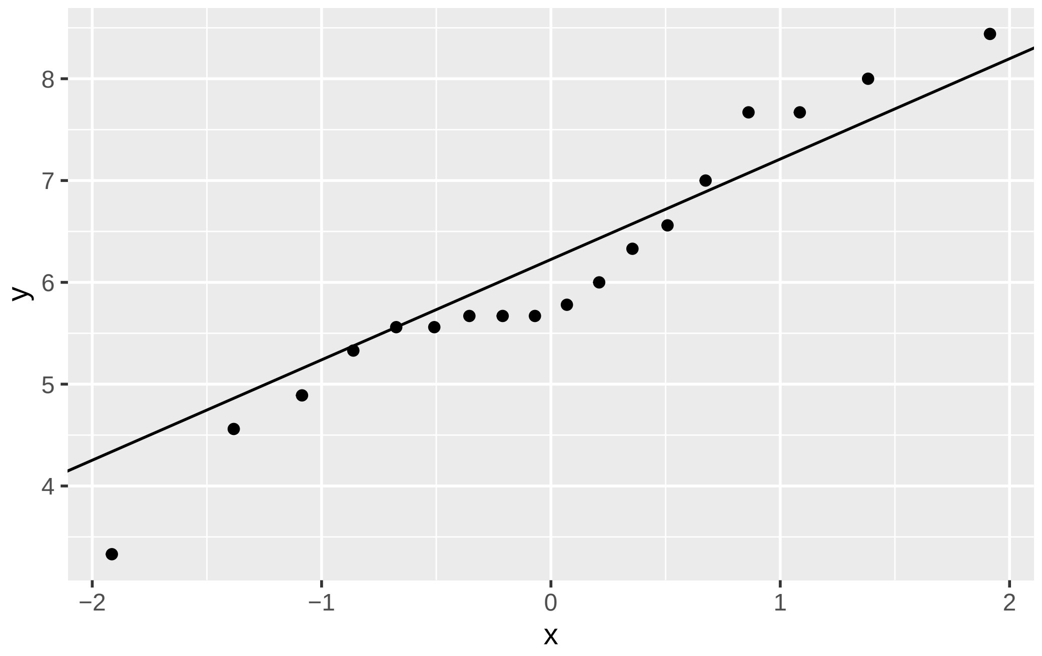


Figure 9: Pork cooling: normal quantile plots (i)

```
ggplot(cooling, aes(sample = delta)) + stat_qq() + stat_qq_line()
```

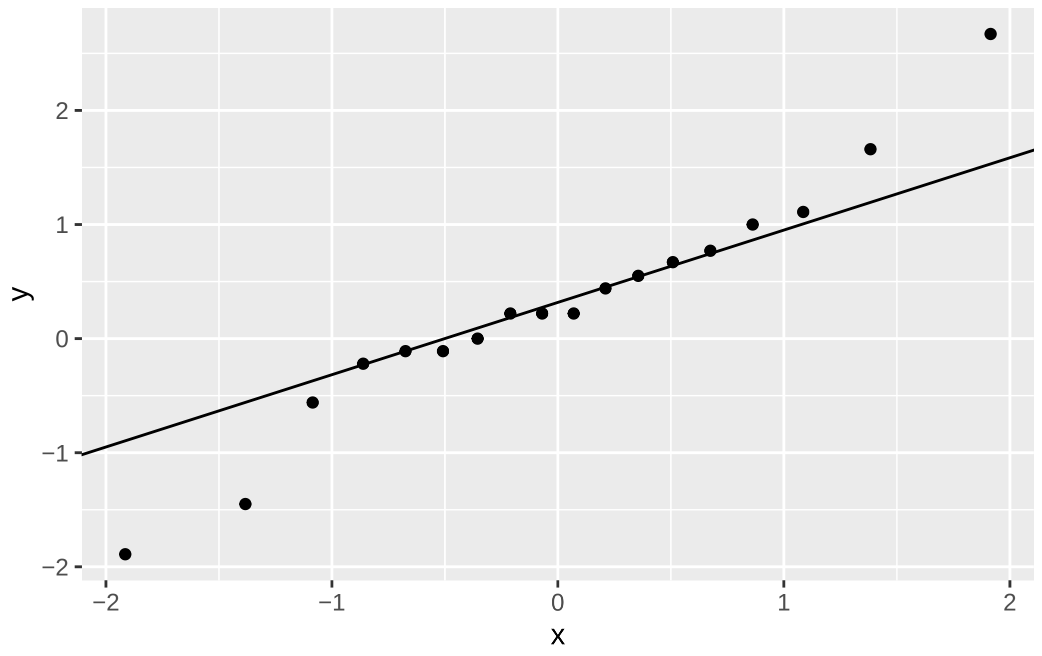


Figure 10: Pork cooling: normal quantile plots (ii)

```
    Paired t-test

data:  tunnel and rapid
t = 1.177, df = 17, p-value = 0.2554
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -0.2285208  0.8051875
sample estimates:
mean difference
      0.2883333
```

Figure 11: Pork cooling: t-test

adhesive	strength
adhesv.1	229
adhesv.1	286
adhesv.1	245
adhesv.1	299
adhesv.1	250
adhesv.2	213
adhesv.2	179
adhesv.2	163
adhesv.2	247
adhesv.2	225

Figure 12: Adhesives data

```
$grand_median
[1] 237

$table
          above
group       above below
  adhesv.1     4     1
  adhesv.2     1     4

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

Figure 13: Adhesives Mood median test

```
carbon
```

Monoxide	Site
0.106	Site1
0.127	Site1
0.132	Site1
0.105	Site1
0.117	Site1
0.109	Site1
0.107	Site1
0.109	Site1
0.122	Site2
0.119	Site2
0.115	Site2
0.120	Site2
0.117	Site2
0.135	Site2
0.118	Site2
0.142	Site2
0.119	Site3
0.110	Site3
0.106	Site3
0.108	Site3
0.105	Site3
0.121	Site3
0.109	Site3
0.134	Site3

Figure 14: Carbon monoxide data

```
            Df    Sum Sq   Mean Sq F value Pr(>F)
Site         2 0.0004813 2.407e-04   2.411  0.114
Residuals   21 0.0020960 9.981e-05
```

Figure 15: Carbon monoxide ANOVA

```
ggplot(carbon, aes(sample = Monoxide)) + stat_qq() +
  stat_qq_line() + facet_wrap(~ Site)
```

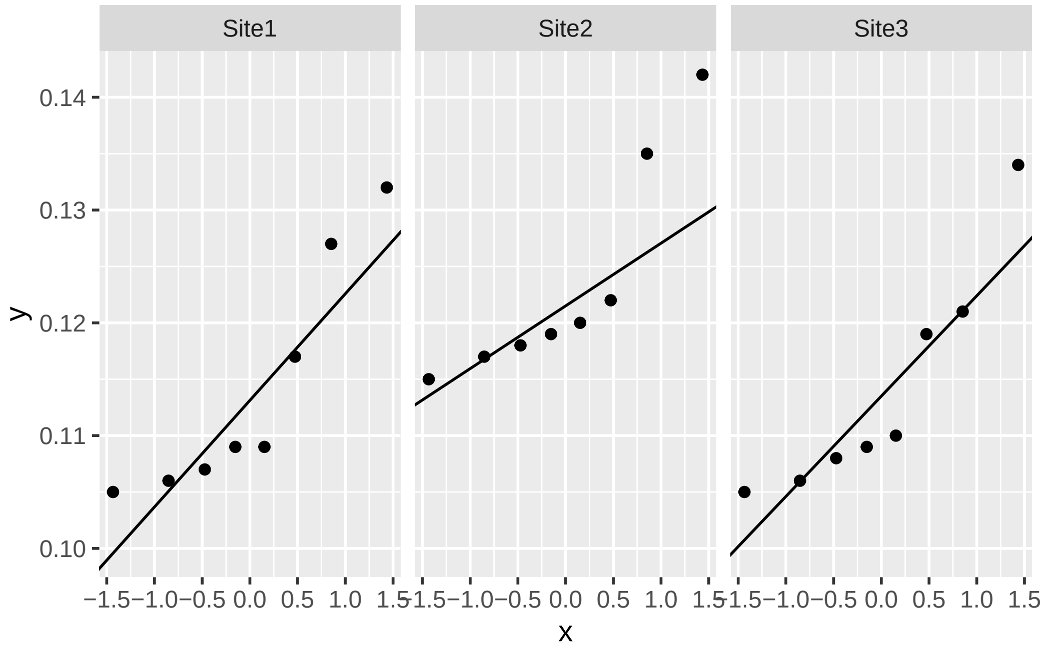


Figure 16: Carbon monoxide graph

```
$grand_median
[1] 0.117

$table
       above
group   above below
  Site1     2     5
  Site2     6     1
  Site3     3     5

$test
       what      value
1 statistic 5.35714286
2        df 2.00000000
3   P-value 0.06866117
```

Figure 17: Carbon monoxide alternative analysis

Subject	Hugger	Conventional
1	0.43	0.75
2	0.08	0.55
3	0.18	0.08
4	0.78	0.75
5	0.03	0.05
6	0.23	1.60
7	0.20	0.65
8	0.00	0.13
9	0.05	0.83
10	0.30	0.58
11	0.33	0.38
12	0.00	0.63
13	0.90	0.25
14	0.24	1.03
15	0.15	1.58
16	0.10	0.20
17	0.33	1.88
18	0.33	2.00
19	0.53	0.25
20	0.43	0.18
21	0.65	0.85
22	0.20	1.15
23	0.25	0.93
24	0.15	1.05
25	0.05	0.85
26	0.25	0.88

Figure 18: Toothbrush data: dataframe `tb1`

Subject	Hugger:F	Hugger:M	Conventional:F	Conventional:M
1	0.43	0.15	0.75	1.58
2	0.08	0.10	0.55	0.20
3	0.18	0.33	0.08	1.88
4	0.78	0.33	0.75	2.00
5	0.03	0.53	0.05	0.25
6	0.23	0.43	1.60	0.18
7	0.20	0.65	0.65	0.85
8	0.00	0.20	0.13	1.15
9	0.05	0.25	0.83	0.93
10	0.30	0.15	0.58	1.05
11	0.33	0.05	0.38	0.85
12	0.00	0.25	0.63	0.88
13	0.90	NA	0.25	NA
14	0.24	NA	1.03	NA

Figure 19: Toothbrush data: dataframe `tb2`

gender	ht:13	wt:13	ht:14	wt:14	ht:15	wt:15
female	101	61.7	105	62.5	115.0	62.9
male	100	61.5	112	64.5	123.5	67.0

Figure 20: Dataframe `hw1` of heights and weights

gender	age	ht	wt
female	13	101.0	61.7
female	14	105.0	62.5
female	15	115.0	62.9
male	13	100.0	61.5
male	14	112.0	64.5
male	15	123.5	67.0

Figure 21: Dataframe `hw2` of heights and weights

Year	Yield	Rainfall
1890	24.5	9.6
1891	33.7	12.9
1892	27.9	9.9
1893	27.5	8.7
1894	21.7	6.8
1895	31.9	12.5
1896	36.8	13.0
1897	29.9	10.1
1898	30.2	10.1
1899	32.0	10.1
1900	34.0	10.8
1901	19.4	7.8
1902	36.0	16.2
1903	30.2	14.1
1904	32.4	10.6
1905	36.4	10.0
1906	36.9	11.5
1907	31.5	13.6
1908	30.5	12.1
1909	32.3	12.0
1910	34.9	9.3
1911	30.1	7.7
1912	36.9	11.0
1913	26.8	6.9
1914	30.5	9.5
1915	33.3	16.5
1916	29.7	9.3
1917	35.0	9.4
1918	29.9	8.7
1919	35.2	9.5
1920	38.3	11.6
1921	35.2	12.1
1922	35.5	8.0
1923	36.7	10.7
1924	26.8	13.9
1925	38.0	11.3
1926	31.7	11.6
1927	32.6	10.4

Figure 22: Corn yield data

```
ggplot(corn, aes(x = Rainfall, y = Yield)) + geom_point()
```

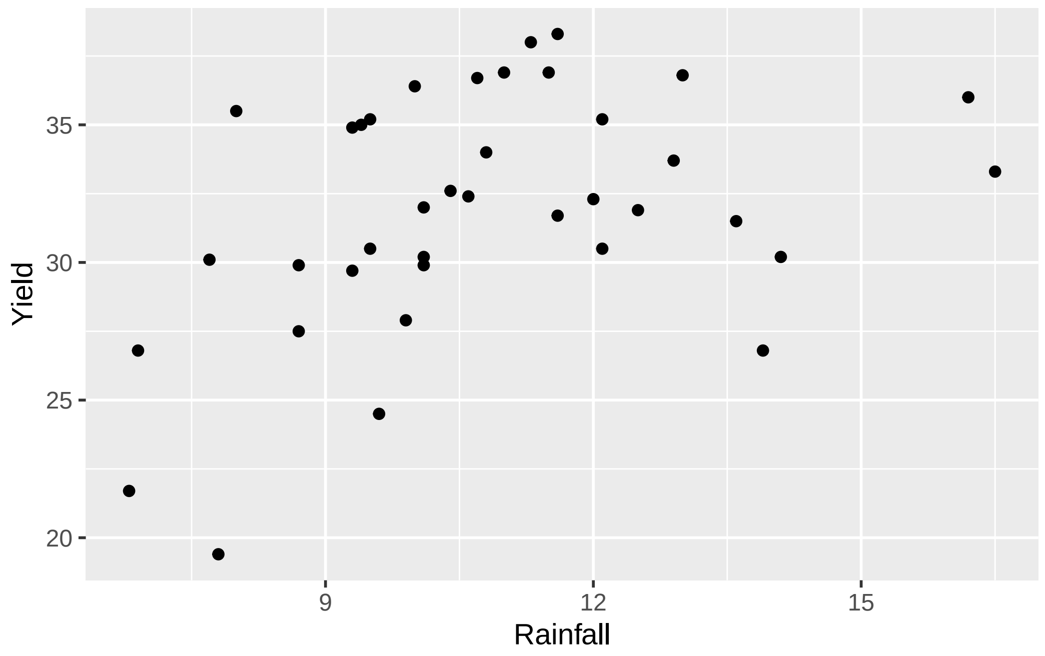


Figure 23: Scatterplot of corn yield against rainfall

```r
corn.1 <- lm(Yield ~ Rainfall + I(Rainfall^2), data = corn)
summary(corn.1)
```

```

Call:
lm(formula = Yield ~ Rainfall + I(Rainfall^2), data = corn)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.4642 -2.3236 -0.1265  3.5151  7.1597 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)   
(Intercept)   -5.01467   11.44158  -0.438  0.66387   
Rainfall       6.00428    2.03895   2.945  0.00571 **
I(Rainfall^2) -0.22936    0.08864  -2.588  0.01397 * 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.763 on 35 degrees of freedom
Multiple R-squared:  0.2967,	Adjusted R-squared:  0.2565 
F-statistic: 7.382 on 2 and 35 DF,  p-value: 0.002115
```

Figure 24: Regression 1, predicting corn yield as a parabola relationship with rainfall

```
corn.1a <- augment(corn.1, data = corn)
ggplot(corn.1a, aes(x = .fitted, y = .resid)) + geom_point()
```

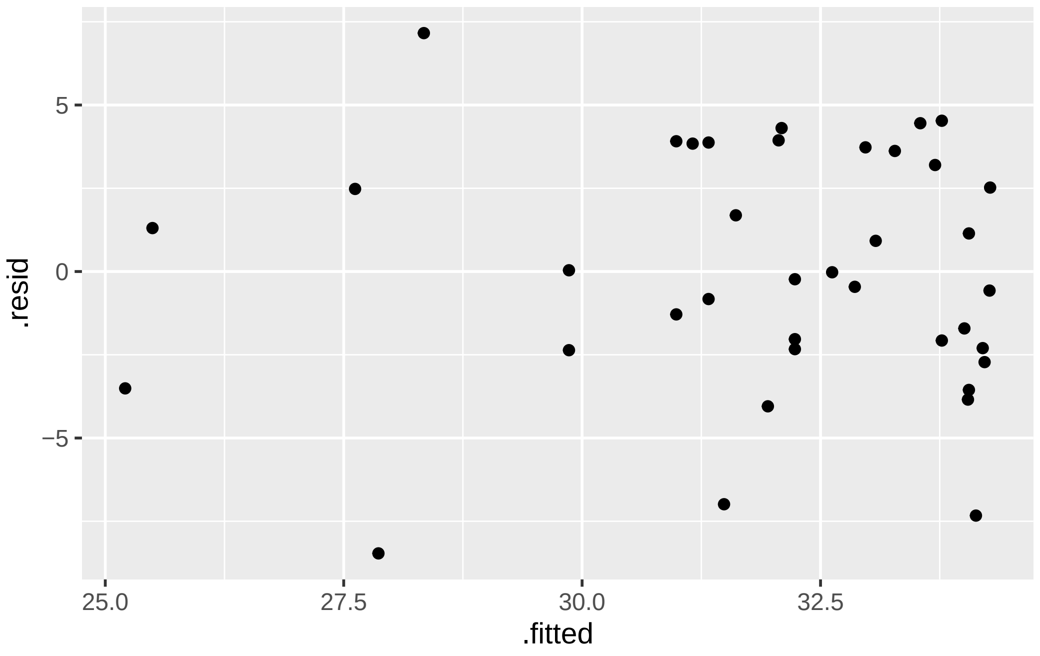


Figure 25: Residuals against fitted values for regression 1

```
ggplot(corn.1a, aes(x = Year, y = .resid)) + geom_point() + geom_smooth()
```

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

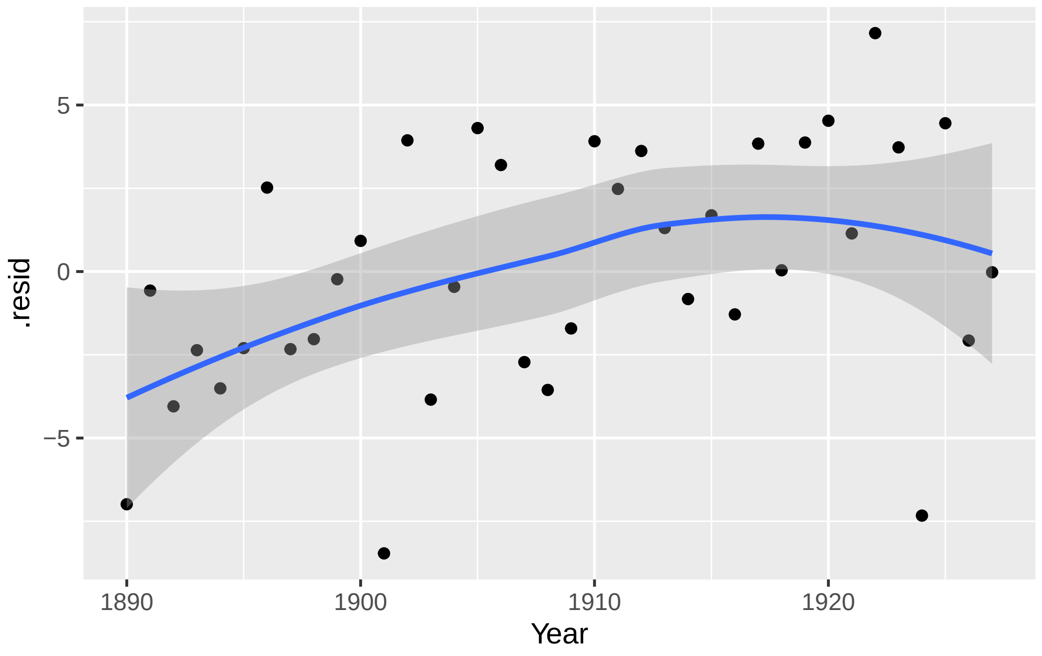


Figure 26: Residuals against year for regression 1, with smooth trend

```r
corn.2 <- lm(Yield ~ Rainfall + I(Rainfall^2) + Year, data = corn)
summary(corn.2)
```

```

Call:
lm(formula = Yield ~ Rainfall + I(Rainfall^2) + Year, data = corn)

Residuals:
    Min      1Q  Median      3Q     Max 
-9.3995 -1.8086 -0.0479  2.4050  5.1839 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)   
(Intercept)   -263.30324   98.24094  -2.680  0.01126 * 
Rainfall         5.67038    1.88824   3.003  0.00499 **
I(Rainfall^2)   -0.21550    0.08207  -2.626  0.01286 * 
Year             0.13634    0.05156   2.644  0.01229 * 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.477 on 34 degrees of freedom
Multiple R-squared:  0.4167,	Adjusted R-squared:  0.3652 
F-statistic: 8.095 on 3 and 34 DF,  p-value: 0.0003339
```

Figure 27: Second regression for predicting corn yield

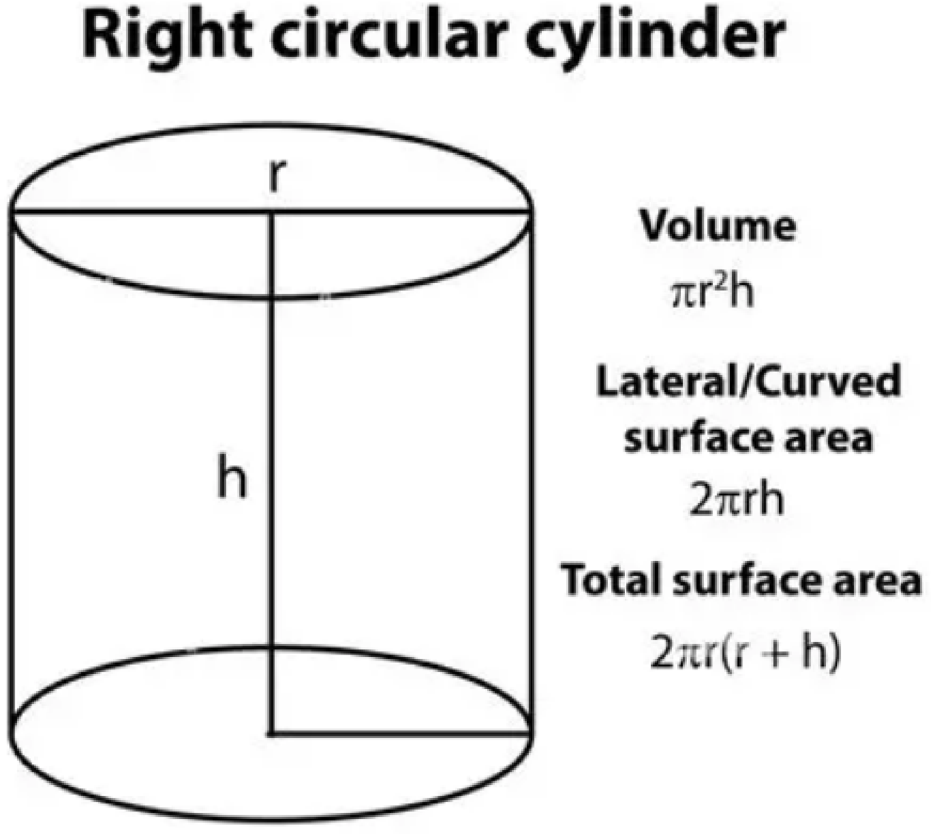


Figure 28: Cylinder of radius r and height h