University of Toronto Scarborough
Department of Computer and Mathematical Sciences
STAC33 (K. Butler), Final Exam
April 18, 2022

Aids allowed: 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 is open book, as above.
This exam has 31 numbered pages of questions.
In addition, you have an additional booklet of Figures to refer to during the exam. Contact an invigilator if you do not have this.
The maximum marks available for each part of each question are shown next to the question part.
You may assume throughout this exam that the code shown in Figure 1 of the booklet of Figures has already been run.
The University of Toronto's Code of Behaviour on Academic Matters applies to all University of Toronto Scarborough students. The Code prohibits all forms of academic dishonesty including, but not limited to, cheating, plagiarism, and the use of unauthorized aids. Students violating the Code may be subject to penalties up to and including suspension or expulsion from the University.

Question 1 (6 marks)

In a chemical procedure called differential pulse polarography, a chemist measured the maximum `Current` generated (in millionths of an ampere) in a solution containing a given amount of nickel (`Ni`), measured in parts per billion. The machine outputs the data in the format shown in Figure 2. This is in a file called `nickel.txt` in the same folder where you are currently running R Studio.

(a) (3 marks) What R code would read the data shown in Figure 2 into a dataframe called `nickel`?

My answer:

Two (evidently) quantitative variables, separated by a colon rather than the usual space. So use `read_delim` with second input `:` as shown:

```
my_url <- "nickel.txt"
nickel <- read_delim(my_url, ":")
```

```
## Rows: 9 Columns: 2
## -- Column specification --------------------------------------------------
## Delimiter: ":"
## dbl (2): Ni, Current
##
## 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.
```

```
nickel
```

```
## # A tibble: 9 x 2
##      Ni Current
##   <dbl>   <dbl>
## 1  19.1   0.095
## 2  38.2   0.174
## 3  57.3   0.256
## 4  76.2   0.348
## 5  95     0.429
## 6 114     0.5
## 7 131     0.58
## 8 150     0.651
## 9 170     0.722
```

You need the first two lines, or you can put the filename directly into the `read_delim`. The points are for (i) including the right file name in the right place, (ii) using `read_delim`, (iii) using the colon as the second input.

It will work if you have *no* second input to `read_delim`, but if you do it that way, you need to explain *why* it works for the third point:

```
read_delim(my_url)
```

```
## Rows: 9 Columns: 2
## -- Column specification --------------------------------------------------
## Delimiter: ":"
## dbl (2): Ni, Current
##
## 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.
```

Question 1 continues… This page: 3 marks.

```
## # A tibble: 9 x 2
##      Ni Current
##   <dbl>   <dbl>
## 1  19.1   0.095
## 2  38.2   0.174
## 3  57.3   0.256
## 4  76.2   0.348
## 5  95     0.429
## 6 114     0.5
## 7 131     0.58
## 8 150     0.651
## 9 170     0.722
```

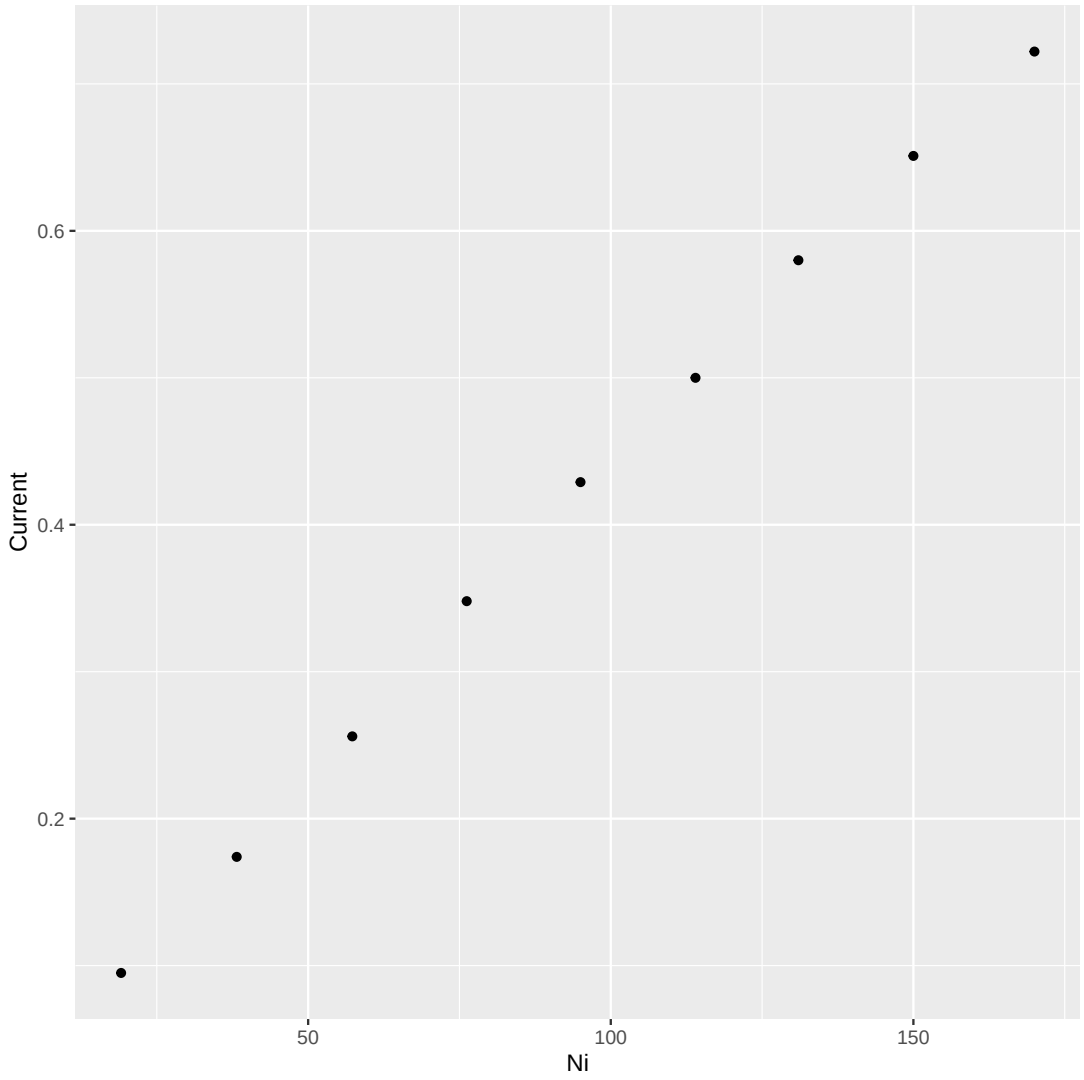
The indication in the message of "delimiter :" shows that `read_delim` successfully guessed what the delimiter was. This is what you need to say: that the delimiter will be guessed as a colon, and the guess will be successful because the (numeric) data really is separated by a single colon all the way down.

(b) (3 marks) What would be an appropriate graph for these data? Explain briefly. What `ggplot` R code would draw this graph?

My answer:

There are two quantitative variables, so a scatterplot would make sense. In the description of the experiment, the concentration of nickel was controlled, so this is explanatory (on x-axis of plot), and the current was observed in response to that, so this is the response (y-axis of plot):

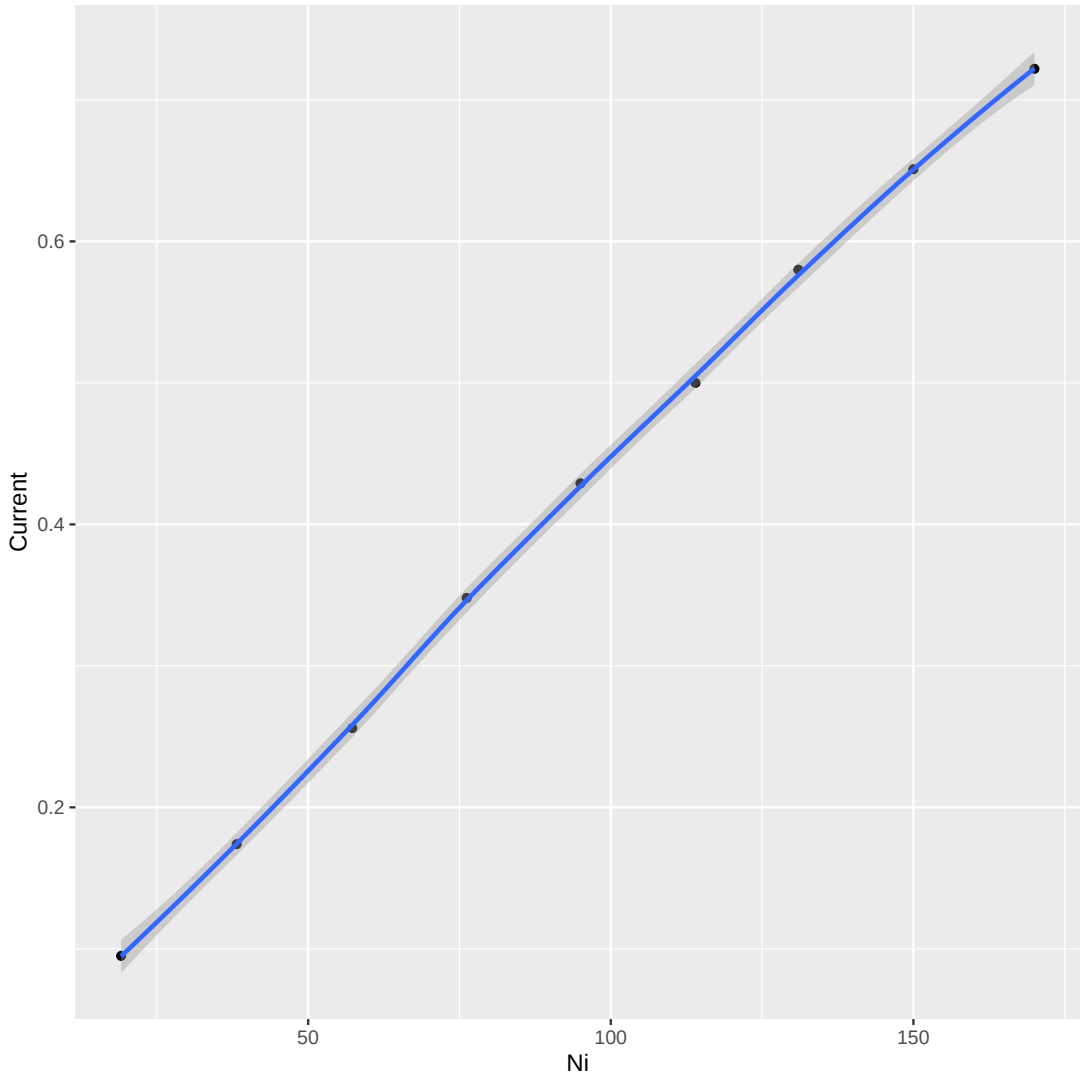
```
ggplot(nickel, aes(x = Ni, y = Current)) + geom_point()
```



Optionally, add a smooth trend to the plot (this is better than a regression line, because you don't know yet whether the trend is linear):

```
ggplot(nickel, aes(x = Ni, y = Current)) + geom_point() + geom_smooth()
```

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



It actually is very nearly straight, but you (writing this exam without access to the actual data) don't know that.

One point for naming the plot, one point for saying why, and one point for coding it with the right variables on the right axes.

Question 2 (16 marks)

Acquired immunodeficiency syndrome (AIDS) is a chronic, potentially life-threatening condition caused by the human immunodeficiency virus (HIV). By damaging your immune system, HIV interferes with your body's ability to fight infection and disease. There's no cure for HIV/AIDS, but medications

Question 2 continues... This page: 0 marks.

can dramatically slow the progression of the disease. These drugs have reduced AIDS deaths in many developed nations.

Data was collected on 2843 patients diagnosed with AIDS in Australia before July 1, 1991. At that time, there were no good medications for AIDS.

The variables measured were, in order:

- the `state` of Australia that the patient lives in. Some of the states are combined.
- the `sex` of the patient
- the date of `diag`nosis with AIDS
- the date of `death` or of last observation (if the patient was still alive at last observation)
- the `status` of the patient at the last observation (A = alive, D = dead)
- `T.categ`: the transmission category (how the patient got AIDS in the first place)
- the `age` at diagnosis in years.

Some of the dataframe, called `aids`, is shown in Figure 3.

For each of the parts below, give (Tidyverse) code that would accomplish the task shown.

(a) (2 marks) Display the median and first and third quartiles of age at diagnosis for patients from each state.

My answer:

Group-by and summarize:

```
aids %>%
  group_by(state) %>%
  summarize(median_age = median(age),
            q1_age = quantile(age, 0.25),
            q3_age = quantile(age, 0.75))
```

```
## # A tibble: 4 x 4
##   state median_age q1_age q3_age
##   <fct>      <dbl>  <dbl>  <dbl>
## 1 NSW           37     30     43
## 2 Other         37     31     43
## 3 QLD           36     30     45
## 4 VIC           36     30     43
```

Extra: this is what will happen if you try to get all three quantiles at once:

```
aids %>%
  group_by(state) %>%
  summarize(stats = quantile(age, c(0.25, 0.50, 0.75)))
```

```
## `summarise()` has grouped output by 'state'. You can override using
## the `.groups` argument.
## # A tibble: 12 x 2
## # Groups:   state [4]
##    state stats
##    <fct> <dbl>
```

```
##  1 NSW      30
##  2 NSW      37
##  3 NSW      43
##  4 Other    31
##  5 Other    37
##  6 Other    43
##  7 QLD      30
##  8 QLD      36
##  9 QLD      45
## 10 VIC      30
## 11 VIC      36
## 12 VIC      43
```

This, I guess, works: all three percentiles are shown for each state, so that all the information is there, if not in the nicest format (and not labelled by which quantile is which, but of course you can work that out). So if you do it this way, two points.

(b) (2 marks) Display the columns of the dataframe whose names begin with "d", without explicitly naming any columns.

My answer:

`select`, using `starts_with` to get the ones you want:

```
aids %>%
  select(starts_with("d"))
```

If you actually run that, you get lots of rows, but to show that it will work:

```
aids %>%
  select(starts_with("d")) %>%
  slice(1:10)
```

```
##          diag      death
## 1  1989-11-09 1990-05-04
## 2  1990-03-13 1990-05-19
## 3  1986-02-24 1987-05-02
## 4  1986-03-22 1986-06-07
## 5  1987-06-03 1988-03-04
## 6  1987-04-20 1988-04-27
## 7  1989-06-03 1990-06-27
## 8  1987-06-30 1990-04-22
## 9  1988-08-25 1989-12-30
## 10 1988-07-31 1989-10-08
```

(c) (3 marks) Display the patients who are either from the state of Queensland (QLD) or are aged over 50 (or both).

My answer:

Choosing rows, so `filter`. Use `|` for "or":

```
aids %>%
  filter(state == "QLD" | age > 50)
```

and to show that it will work:

```
aids %>%
  filter(state == "QLD" | age > 50) %>%
  sample_n(20)
```

```
##    state sex       diag      death status T.categ age
## 1    QLD   M 1988-10-23 1991-07-01      A      hs  41
## 2  Other   M 1989-07-21 1989-07-24      D      hs  60
## 3    QLD   M 1989-12-19 1989-12-27      D      hs  57
## 4    QLD   M 1991-01-13 1991-04-30      D      hs  57
## 5    NSW   M 1991-02-05 1991-06-20      A      hs  56
## 6    NSW   M 1990-06-17 1991-04-29      D     het  58
## 7    NSW   M 1990-04-27 1990-05-10      D      hs  58
## 8    NSW   M 1985-11-21 1986-01-29      D      hs  64
## 9    VIC   M 1989-04-24 1990-08-12      D      hs  57
## 10   NSW   M 1991-06-19 1991-07-01      A      hs  63
```

```
## 11   QLD   M 1990-04-23 1991-07-01      A     het  25
## 12   QLD   M 1984-07-16 1984-08-01      D   blood   0
## 13   QLD   F 1991-05-13 1991-07-01      A      hs  23
## 14   QLD   M 1990-03-08 1991-07-01      A      hs  32
## 15   NSW   M 1985-03-19 1991-07-01      A   blood  73
## 16   NSW   M 1989-06-17 1990-07-25      D      hs  53
## 17   QLD   M 1985-03-10 1985-06-26      D      hs  52
## 18   QLD   M 1989-11-16 1991-07-01      A      hs  44
## 19   NSW   M 1990-08-21 1990-08-22      D      hs  56
## 20   QLD   M 1988-12-20 1989-06-05      D      hs  51
```

As before, there are lots of patients meeting this criterion, so I only displayed (a random sample of) 20 of them. You can see that the ones displayed either really are from Queensland, or, if they are not, their age is over 50. There are a few patients who are from Queensland and are also aged over 50.

If you forgot "or" but remembered De Morgan's laws, you can also do it this way:

```
aids %>%
  filter(!(state != "QLD" & age <= 50)) %>%
  sample_n(20)
```

```
##    state sex       diag      death status T.categ age
## 1    QLD   M 1988-09-17 1989-11-07      D      hs  29
## 2    QLD   M 1988-04-12 1990-03-28      D    hsid  30
## 3    QLD   M 1988-04-13 1990-07-29      D      hs  31
## 4    QLD   M 1990-11-14 1990-11-24      D      hs  47
## 5  Other   M 1989-06-24 1989-11-17      D      hs  56
## 6    NSW   M 1990-09-08 1991-05-21      D      hs  53
## 7    VIC   M 1991-02-16 1991-07-01      A      hs  59
## 8    NSW   M 1990-10-24 1991-05-12      D      hs  51
## 9    NSW   M 1990-04-27 1990-05-10      D      hs  58
## 10   NSW   M 1987-11-04 1988-08-23      D   blood  65
## 11   VIC   M 1989-11-25 1991-07-01      A      hs  52
## 12   QLD   M 1986-11-20 1986-11-20      D      hs  34
## 13   NSW   M 1988-09-11 1989-04-08      D    haem  52
## 14   NSW   M 1986-10-09 1987-05-26      D      hs  57
## 15   NSW   M 1991-05-09 1991-07-01      A      hs  54
## 16 Other   M 1990-05-29 1991-07-01      A      hs  53
## 17   VIC   M 1987-07-21 1988-09-10      D   other  66
## 18   VIC   M 1988-06-12 1988-07-15      D      hs  51
## 19   QLD   M 1990-10-10 1991-07-01      A      hs  44
## 20   QLD   M 1985-03-18 1985-04-03      D      hs  28
```

Negate both conditions, change OR to AND, then negate the whole thing. This also seems to have worked: the patients under 50 are all from Queensland.

(d) (3 marks) Find the earliest and latest last-observation dates for the patients who were still alive at last observation. I don't want to know about patients who died.

My answer:

The last sentence indicates that we should get rid of the patients who died, by filtering only the patients who were still alive at last observation (in column `status`, labelled A), or by filtering the patients whose status is *not* equal to D. Then, having done that, we should find the min and max of `death` (which are the last-observation dates for the patients who were still alive):

```
aids %>%
  filter(status == "A") %>%
  summarize(min_last = min(death), max_last = max(death))
```

```
##     min_last   max_last
## 1 1989-09-27 1991-07-01
```

or this:

```
aids %>%
  filter(status != "D") %>%
  summarize(min_last = min(death), max_last = max(death))
```

```
##     min_last   max_last
## 1 1989-09-27 1991-07-01
```

The latest date should not be a surprise, since that was (as in the description of the data) the end of the study, so nothing was observed after that. (The data set comes from the MASS package, which is based on a book called "Modern Applied Statistics in S", which is not especially modern any more. In the detailed description in the book, the study actually continued until the end of 1991. But the experimenters might not have found out that the patients who died late in 1991 had actually died: there was a delay in finding out about deaths. Thus the authors took the endpoint of the study to be July 1, 1991, to be sure that they knew about all the patients who died.)

The earliest last-followup date was back in 1989. This was a patient who was never known to die, but the experimenters had no idea what happened to this patient after September 1989. This is what survival analysis people call "censored data":

```
aids %>%
  filter(status == "A" & death < as.Date("1989-10-01"))
```

```
##   state sex       diag      death status T.categ age
## 1   QLD   M 1987-06-27 1989-09-27      A      hs  20
```

This was a male from Queensland who was diagnosed with AIDS at the age of 20 in June 1987, and who was known to have lived for a bit over two years after diagnosis. There is some information here about survival times, if not much; we know that their time from diagnosis to death was at least two years and three months. One of the fun things about dealing with survival data is that you might get a lot of data like this. For people that die, it is in some sense easier; you have a model for survival times (times from diagnosis until death), say a gamma distribution, that might depend on other things (like age, or state of residence, here), and you can write down a likelihood for these people. But for people like the one we're considering here who were never observed to die, all you can do is to write down the probability of living at least 2 years and 3 months after diagnosis and incorporating *that* into the likelihood. This makes the likelihood very messy, and it is impossible to maximize the likelihood other than by numerical methods (to estimate the effect of state on survival rate, say, or to test whether there is a significant effect of state at all).

(e) (3 marks) Display only the age and how the patient first got AIDS for only the patients from the state of Victoria (VIC).

My answer:

This means choosing some columns and some rows, but the rows you choose depend on a column you are *not* choosing. So do the filter *first*, and *then* do the select to grab the columns you want. "How the patient first got AIDS" is in the column `T.categ`:

```
aids %>%
  filter(state == "VIC") %>%
  select(age, T.categ)
```

and to show that this works:

```
aids %>%
  filter(state == "VIC") %>%
  select(age, T.categ) %>%
  slice(1:10)
```

```
##    age T.categ
## 1   43      hs
## 2   38      hs
## 3   47      hs
## 4   32      hs
## 5   51      hs
## 6   43      hs
## 7   27      hs
## 8   35      hs
## 9   29      hs
## 10  43      hs
```

Once again, the `slice` is to make sure I don't display too many rows.

(f) (3 marks) Display the age and status of each of the 12 oldest patients who come from New South Wales (NSW).

My answer:

Get all the patients from New South Wales, then sort them in order by age (in descending order), then select the age and status columns, then display the first 12 of those:

```
aids %>%
  filter(state == "NSW") %>%
  arrange(desc(age)) %>%
  select(age, status) %>%
  slice(1:12)
```

```
##    age status
## 1   82      D
## 2   80      D
## 3   78      D
## 4   78      D
## 5   77      A
```

```
## 6   74      D
## 7   73      D
## 8   73      A
## 9   72      D
## 10  71      D
## 11  70      D
## 12  70      D
```

This time, evidently, you *do* need the `slice`. The `filter` can be later, as long as it comes before the `select`: that is, you could sort by age first, and when you pull out the patients from New South Wales, they will still be sorted in order by age. The `arrange` could also be right before the `slice`. In summary:

- there must be a `filter`, an `arrange`, a `select`, and a `slice`.
- the `filter` must be before the `select` (since otherwise you won't have a `state` to filter by)
- the `arrange` must be before the `slice` (since otherwise you'll be displaying the *first* twelve patients rather than the oldest twelve, even if you then sort them by age)
- the `filter` must be before the `slice` (since otherwise you could end up displaying patients not from New South Wales).

Thus, any of these orders of the four elements are good:

```
filter arrange select slice
filter arrange slice select
filter select arrange slice
arrange filter select slice
arrange filter slice select
```

This, I anticipate, is going to be a pain to grade.

Question 3 (12 marks)

An environmentalist heard reports that a community was discharging untreated sewage into a river, and feared that this might have an impact on the ability of the river to support aquatic life. To investigate, the environmentalist chose at random five locations upstream from the community and five locations downstream from the community. At each one, they took a sample of the river water and measured the dissolved oxygen content, in parts per million. The higher the dissolved oxygen content is, the better the water is for supporting aquatic life, and dumping sewage into a river will decrease the dissolved oxygen content.

(Hint: a river flows downhill, so that water flows from upstream of a point to downstream of that point, and not the other way around.)

The data are shown in Figure 4, in the form that the environmentalist collected it. The data frame is called `river_water`.

(a) (3 marks) Describe briefly why the data as shown in Figure 4 are not suitable for an appropriate analysis, as we did it in lecture. (If you find it helpful, as part of your answer you can describe what an appropriate format would be.)

My answer:

There are two things to say: (i) what the appropriate analysis is (one point), and (ii) what the data should look like for that analysis (two points).

These are 10 independent observations (there is no matching up of, say, observation 1 upstream of the community and observation 1 downstream of it). Thus the appropriate analysis should be something like a two-sample t-test rather than matched pairs. (Mood's median test is also reasonable based on what you have learned about the data so far.) At a minimum, say that this is laid out like paired data when it's actually two independent samples (there is no pairing of an upstream site with a downstream one).

The way we learned a two-sample t-test (or a Mood's median test), we need all 10 dissolved oxygen content values in *one* column, with a second column saying whether it was upstream or downstream. If you like, write out what it should look like:

```
## # A tibble: 10 x 2
##    where      oxygen
##    <chr>       <dbl>
##  1 upstream      4.8
##  2 downstream    5
##  3 upstream      5.2
##  4 downstream    4.7
##  5 upstream      5
##  6 downstream    4.9
##  7 upstream      4.9
##  8 downstream    4.8
##  9 upstream      5.1
## 10 downstream    4.9
```

or some reasonable fraction of that, enough to give the grader the idea. Saying that the data is "untidy" is too vague, and copying words from my lecture notes is not any help to you in answering the question. You need to say something specific to *these* data as they are laid out.

Note that if this had been paired data, the format in the Figure would have been *exactly correct* (to use with `paired = TRUE` in `t.test`, for example, or to calculate differences between paired observations). Insisting that the the data should be tidy without thinking about this issue is, therefore, missing the point. Or, at least, failing to appreciate the whole point.

Extra: I got these data from a textbook, and they were laid out as in Figure 4, probably to save space on the page. You often see people arrange data this way in a spreadsheet also, but that rather betrays that they are not thinking very much about the analysis and what the data needs to look like for that.

This is in some sense a "toy" problem, in that in real life, a lot more data would be collected, but I wanted one where you could see exactly what was happening (where I could easily show you all the data).

(b) (2 marks) What code would transform the data in Figure 4 into a suitable layout? Save your results into a dataframe `river2`.

My answer:

This is a classic pivot-longer:

```
river_water %>%
  pivot_longer(everything(), names_to = "where", values_to = "oxygen") -> river2
```

and to confirm that all is good, which I can do since I am sitting in front of a computer:

```
river2
```

```
## # A tibble: 10 x 2
##    where      oxygen
##    <chr>       <dbl>
##  1 upstream      4.8
##  2 downstream    5
##  3 upstream      5.2
##  4 downstream    4.7
##  5 upstream      5
##  6 downstream    4.9
##  7 upstream      4.9
##  8 downstream    4.8
##  9 upstream      5.1
## 10 downstream    4.9
```

I don't mind much what you call the new columns, except that if I think the names are confusing, don't expect to get full marks.

If you messed up the previous part, you might get 1 here for something consistent enough with your previous part.

(c) (4 marks) The environmentalist knows from previous experience that dissolved oxygen content values have very close to a normal distribution, though not necessarily with equal spreads at different locations. Some possible analyses are shown in Figure 5 through Figure 8. One of these analyses is the most appropriate, and the other three are less appropriate. For each Figure in turn, briefly discuss whether or not it is the most appropriate and why.

My answer:
One point each:

- Figure 5: this is a matched pairs analysis, which is no good since we already said that this is two independent samples. In addition, it uses the dataframe in Figure 4, which we already said is not appropriate (and you said why earlier). Alternatively (but not quite so good), this test assumes the differences are normal, which we don't know about.
- Figure 6: this is a two-sample (Welch) t-test. The normality assumption is reasonable (from what the environmentalist knows about these kinds of measurements), and it does not assume equal spreads (which was an assumption the environmentalist was not prepared to make). So this one is appropriate, and after scanning the others you can see that it is the most appropriate.
- Figure 7: this is a two-sample pooled t-test. But a pooled test assumes equal spreads (of the dissolved oxygen content values at the two different values of `where`, upstream and downstream of the community), and the environmentalist was not prepared to make this assumption. So, not appropriate.
- Figure 8: this is a Mood's median test. The P-value given in the table is two-sided, but the values above and below are "correct side" (more of the upstream values are above the overall median and more of the downstream ones are below), so we can halve the P-value to get the one-sided one (the calculation below the output). This is meant to dissuade you from claiming that this one is no good because it should be one-sided. The Mood's median test is appropriate here, but it is not as much so as the right t-test (Figure 6) when the assumptions for that t-test are satisfied, which they are here. (I was careful to say that the observations are very close to normal so that the small sample sizes are not a problem.)

With all that said, the best analysis is the one in Figure 6, and the others could be better for the reasons given.

In your answer, use the Figure numbers that I had (or clearly describe which test you are looking at in each case), or else you make me work harder to sort out whether you know what you are saying (and, in that case, I will look more carefully for reasons to deduct marks).

Last: *do not ever* choose between tests on the basis of their P-values. This is called "P-hacking" and is dishonest science, because you could get a dishonestly small P-value by doing lots of potentially relevant tests and choosing the smallest of their P-values. Indeed, by doing enough tests, you could get a P-value as small as you like *even if there is actually no effect.* I don't think you want to get a reputation as a dishonest scientist. So don't do this.

(d) (3 marks) From your best analysis (of the previous part), what conclusion do you draw, in the context of the data and bearing in mind what the environmentalist would like to know?

My answer:
The four P-values were 0.8598, 0.07536, 0.07348, 0.13515. So whichever analysis you preferred, the P-value was not small enough to reject the null hypothesis at the usual α of 0.05. One

point. This means that there is not enough evidence to declare that the mean (median) oxygen content is lower downstream of the community than it is upstream. The second point. That is to say, the environmentalist does not have evidence to prove that untreated sewage was being discharged into the river (because if there had been, the dissolved oxygen content values would have been significantly lower on average downstream of the community.) The third point. If you get to this last point without going through the previous two things, that's fine as long as it looks as if you know what you're doing. The environmentalist's concern is whether the sewage discharge was happening or not; the business with the dissolved oxygen content was a step on the way to answering *that* question, so if you didn't get all the way there, you might have lost a half point.

The question says that we are entitled to take as true that dumping sewage into a river *will* reduce the dissolved oxygen content (without fail, we assume). Since we have found no evidence of a decrease in oxygen content, we have therefore also found no evidence of sewage dumping. We could be wrong in a couple of ways: maybe sewage was dumped and we didn't detect it (a type II error), or sewage was dumped and somehow this didn't show up in the dissolved oxygen content, a possibility I am saying we can ignore.

Remember how to handle one-sided hypotheses: the *null has an equals sign in it* always, so if here μ_1 is the population mean oxygen content downstream of the community, and if μ_2 is the same thing upstream of the community, our hypotheses here are $H_0 : \mu_1 \geq \mu_2$ and $H_a : \mu_1 < \mu_2$. (I find it easier to write the alternative first and then match the null to it.) Here, we failed to reject the null, so the mean dissolved oxygen content downstream is *greater than or equal to* the mean upstream. We have certainly not proved that it was higher downstream, first because that's not what the null says, and second because failing to reject the null does not entitle you to say that you have proved *anything*. The type I error probability comes from what happens when the two means are equal, because that's the part of the null nearest to the alternative.

It turned out that the mean dissolved oxygen content was a little lower downstream of the community, but our P-value indicates that whether it was lower or higher is just chance. If you wanted to quantify the difference between upstream and downstream, you would obtain a confidence interval for the difference in means, which would mean running the test again, but this time two-sided, and you would get a confidence interval that included both positive and negative values, indicating that the difference could go either way. The (negative) difference in means that was observed is statistically meaningless, in that it is not reproducibly negative: if the environmentalist did this whole study again, it is quite possible that things could come out the other way around.

Extra 1: you might reasonably argue that the environmentalist should have taken a lot more than five water samples from upstream and downstream of the community. Had they done so, a difference like the one observed might have come out significant. As it was, there was too much variability to find a significant difference between the means (medians).

Extra 2: note, as often, the P-values for the Welch and pooled t-tests are almost identical. That means that it didn't actually matter which of those two tests was done in the final analysis, but logically we should respect the environmentalist's judgement and prefer the Welch test to the pooled one. In this case it made no difference, but it other situations it would, and it pays to be careful.

Extra 3: often, in a situation where the (Welch) t-test applies, its P-value is smaller than the

one for Mood's test. This is the case here, and is because the Welch test is using the data better (or is more powerful) than Mood's median test, and so is to be preferred. We don't "need non-normality" to be doing Mood's test; it is *valid* any time, but it is going to be less powerful than a t-test when the latter is valid, and so a t-test should be preferred when the data are normal enough.

Question 4 (13 marks)

This question uses the dataframes shown in Figure 9 through Figure 12. In each case, I show you some code that starts from one of these dataframes, and I ask you to write down what the output is. No explanation is necessary, but if you are wrong, an explanation might get you some partial credit.

(a) (3 marks) Suppose the following code is run:

```
d1 %>% pivot_longer(y:z, names_to = "name", values_to = "value")
```

What would be the output from this code?

My answer:

Here it is:

```
d1
```

```
## # A tibble: 2 x 3
##       x     y     z
##   <dbl> <dbl> <dbl>
## 1    10    11    13
## 2     9    12    14
```

```
d1 %>% pivot_longer(y:z, names_to = "name", values_to = "value")
```

```
## # A tibble: 4 x 3
##       x name  value
##   <dbl> <chr> <dbl>
## 1    10 y        11
## 2    10 z        13
## 3     9 y        12
## 4     9 z        14
```

The two columns `y` and `z` are pivoted longer, so the column `name` contains the name of the column from which the value came, and the column `value` contains the value. The values in `x` are repeated as necessary to match up with the `y` and `z` they originally went with.

Here and elsewhere in this question, the order of the rows is not important, as long as the values match up (which I appreciate is a pain for the grader). In this one, `y` in `name` goes with 11 and 12 in `value`; one of the `y` values goes with an `x` of 10 and the other with 9.

(b) (3 marks) Suppose the following code is run:

```
d2 %>%
  pivot_longer(starts_with("y"), names_to = c(".value", "when"), names_sep = "_")
```

What would be the output from this code?

My answer:

```
d2
```

```
## # A tibble: 2 x 4
##   x     y_first y_second y_third
##   <chr>   <dbl>    <dbl>   <dbl>
## 1 a          10       11      13
## 2 b          12       15      16
```

```
d2 %>%
```

```
  pivot_longer(starts_with("y"), names_to = c(".value", "when"), names_sep = "_")
## # A tibble: 6 x 3
##   x     when      y
##   <chr> <chr>  <dbl>
## 1 a     first     10
## 2 a     second    11
## 3 a     third     13
## 4 b     first     12
## 5 b     second    15
## 6 b     third     16
```

The `.value` says to create a column named `y` (the first bit is always `y`) filled with the values in the dataframe, and then to create a column called `when` filled with the second part of the names of the columns in `d4`. The column `x` not mentioned is repeated as needed to match up with the other values.

(c) (3 marks) Suppose the following code is run:

```
d3 %>% pivot_wider(names_from = x, values_from = y)
```

What would be the output from this code?

My answer:

This is the "easy" version of `pivot_wider` that is literally the flip side of pivoting longer:

```
d3
## # A tibble: 4 x 3
##      id x         y
##   <dbl> <chr> <dbl>
## 1     1 a        10
## 2     2 a        11
## 3     1 b        12
## 4     2 b        13
d3 %>% pivot_wider(names_from = x, values_from = y)
## # A tibble: 2 x 3
##      id     a     b
##   <dbl> <dbl> <dbl>
## 1     1    10    12
## 2     2    11    13
```

The values in `x` are `a` and `b`, so `x` and `y` are replaced by new columns called `a` and `b`; the values in these are the ones in `y`. The rows into which the values of `y` go are actually the ones in `id`, but you can (here) successfully guess that they are in the same order as the original data, because the values in `id` also are.

(d) (4 marks) Suppose the following code is run:

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

What would be the output from this code?

My answer:

I get to try it and see:

```
d4
```

```
## # A tibble: 4 x 3
##   x     y         z
##   <chr> <chr> <dbl>
## 1 a     b         9
## 2 c     d        10
## 3 a     d        11
## 4 c     b        12
```

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

```
## # A tibble: 2 x 3
##   x         b     d
##   <chr> <dbl> <dbl>
## 1 a         9    11
## 2 c        12    10
```

To work out why that:

The names of the new columns are coming from `y`, so the new columns will be called `b` and `d`. The values to go into them will be the ones in `z`, so the 9 and 12 will go into the `b` column and the 10 and 11 will go into `d`. The remaining column, `x`, will determine which rows each numeric value will go into. Thus the 9 and 11 will go into the row where `x` is `a`, and the 10 and 12 will go into the row where `x` is `b`. Thus the result will be as shown. An answer with rows or columns in a different order is fine, since the point of the question was to see whether you know what `pivot_wider` actually does (so that the values are in the right places relative to each other), rather than the nitty-gritty details of how things are ordered.

Question 5 (9 marks)

A realtor in Taiwan collected some information about houses for sale in the city of Taipei, as follows:

- `sale_date`: the date on which the house was sold (in fractional years; for example 2013.5 means June of 2015)
- `age` of the house when it was sold (in years)
- `mrt`: distance from the house to the nearest MRT (rapid transit) station (in metres)
- `conv`: number of convenience stores within walking distance of the house (count)
- `price`: selling price per unit area, in suitable units (tens of thousands of Taiwan dollars per "ping", where a ping is 3.3 square metres)

The realtor's aim was to predict selling price from the other variables, so as to be able to set a good price for houses in Taipei that come on the market in the future. Some of the data is shown in Figure 13.

(a) (2 marks) A Box-Cox analysis is shown in Figure 14. Why do you think the realtor decided to predict the log of the selling price (per unit area), rather than some other function of the selling price?

My answer:

Ordinarily, we would look for defensible values of the transformation parameter λ that are inside the confidence interval, like 0 or 0.5 or 1 (log or square root or no transformation). But there are none of those inside the interval. It looks as if the realtor chose a log transformation because $\lambda = 0$ is at least close to the interval, and a power 0.25 (fourth root) is something they would really have to make a case for, whereas a log-transformation is easy to justify.

Another possible answer would be that a unit change in log-price is a percentage change in actual price, which is easy to interpret (a feature of the house might be related to a 10% increase in selling price, say). But I think it also needs to be said that $\lambda = 0$ is close to the confidence interval for λ.

Extra: there were 414 houses in the data set, a large sample size, so that the confidence interval for λ is going to be short, and there might not be any useful values of λ inside the confidence interval, so we have to get creative.

(b) (2 marks) A regression analysis, predicting log price from the other variables, is shown in Figure 15. Which, if any, explanatory variables should the realtor remove from the regression? Explain briefly.

My answer:

None of them, because the t-tests for their slopes are all strongly significant.

(c) (2 marks) Suppose the realtor decided to remove the explanatory variable `mrt` from the regression. What would happen to R-squared?

My answer:

`mrt` is very strongly significant, with a tiny P-value, so that if it is removed from the regression, R-squared would decrease (one point) substantially (another point).

If you really want to see, the R-squared for the regression in Figure 15 is almost 65%. Taking out `mrt` gives this:

```
houses.1a <- update(houses.1, .~. - mrt)
summary(houses.1a)
```

```
##
## Call:
## lm(formula = log(price) ~ sale_date + age + conv, data = houses)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.73418 -0.17350 -0.01235  0.17890  1.37945
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.026e+02  1.061e+02  -1.909    0.057 .
## sale_date    1.023e-01  5.272e-02   1.940    0.053 .
## age         -7.605e-03  1.306e-03  -5.821 1.18e-08 ***
## conv         8.115e-02  5.052e-03  16.063  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.302 on 410 degrees of freedom
## Multiple R-squared:  0.4121,Adjusted R-squared:  0.4077
## F-statistic: 95.78 on 3 and 410 DF,  p-value: < 2.2e-16
```

All the way down to 41%.

I chose the most significant explanatory variable so that the effect would be the most spectacular. Removing other xs gives a similar effect, just not so big.

The reason that you need an adverb like "substantially" or "a lot" is that taking *any* explanatory variable out, even a non-significant one, will always decrease R-squared, if only by a little. So there is no insight in saying that R-squared will decrease, since this will always happen; the key point is *how much* it will decrease by. Here, you can tell that it will decrease a lot.

(d) (3 marks) Residual plots for the regression of log-price on the other explanatory variables are shown in Figures 16 through 18. Do you see any problems with these residual plots? If so, describe the problems you see; if not, explain how you know that the plots are satisfactory.

My answer:

Figure 16 is basically satisfactory (random), apart from the one very negative residual at the bottom right, and possibly the most positive one top right, which seem to be outliers.

Figure 17, a normal quantile plot, seems to have long tails at both ends. This is partly the result of the two extreme residuals, but there are many other residuals that are larger in size than you might expect, so the problem here is long tails, a feature of the whole distribution, rather than a small number of outliers.

In the four plots of Figure 18, I see no additional problems apart from the negative and positive

residuals (that we saw before), showing up in various places along the bottom and top of these plots. The plots otherwise look random.

In summary, the problem is that there are too many residuals that are too large in size, especially the most negative one and possibly the most positive one. This might be because some of the other houses had features that were not in this data set (extra bedrooms or bathrooms, say, or were in a less desirable neighbourhood) that could have a big impact on selling price.

You have some freedom in a question like this; anything that looks like a reasonable reading of the residual plots is all right, bearing in mind that you don't want to look too long or you will see something that is not really there, that might just be chance. I think the best reading of the first and last plots is that all is OK apart from those one or two large residuals, and the better reading of the second one is "long tails" rather than "outliers", in that the whole distribution of outliers seems to suffer from long tails, rather than a few of the residuals being large in size (though one or two of them are definitely larger than the others).

I hope you know by now that calling for the large-residual observations to be removed is not a sound idea. The only reason to remove observations from the data set is if they are in error. It's certainly fair to look carefully at those two houses with large-sized residuals to see whether there is anything different about them (for example, they might have been the only two houses in a different neighbourhood in which this regression for predicting house prices does not apply, for example because houses in that neighbourhood tend to be different, and then you would be justified in removing them).

Question 6 (18 marks)

In this question, we will be writing and using functions.

(a) (3 marks) Write an R function called `f1` that has one input called `n` which returns the sum of the first `n` integers. For example, `f1(3)` should evaluate to 6 because $1 + 2 + 3 = 6$. For maximum points, your function should be as concise and clear as possible. You may assume that the input to your function is a positive integer.

My answer:

My three-point function is just this:

```
f1 <- function(n) {
  sum(1:n)
}
```

There is no problem with using two R ideas: the `a:b` notation for the integers `a` through `b` inclusive, and the built-in function `sum`. In fact, it's better to build your function on top of well-tested smaller ingredients.

Other ways that work, but are not as clear and concise, and thus are only two points, are:

the Python way, with a for loop:

```
f1a <- function(n) {
  sum <- 0
  for (i in 1:n) {
    sum <- sum + i
  }
  sum
}
```

or the recursive way (say where you learned about this if you do it this way):

```
f1b <- function(n) {
  if (n == 1) return(1)
  n + f1b(n-1)
}
```

or you might even remember that the first n integers sum to $n(n+1)/2$ (this is one of the classic proofs by induction), but your function then needs to have a comment in it:

```
f1c <- function(n) {
  # first n integers add up to n(n+1)/2
  n * (n+1) / 2
}
```

Strictly speaking, this last one will return a decimal number rather than an integer, because of the division (like my `hotpo` in lecture). You can run the answer through `as.integer` as I did there. (Of course, the answer must be an integer because one of n and $n+1$ will be even and thus divisible by 2.) There is also an integer division operator `%/%`.

All of these work:

```
nn <- 3
f1(nn)
## [1] 6
f1a(nn)
```

Question 6 continues…

```
## [1] 6
f1b(nn)
## [1] 6
f1c(nn)
## [1] 6
```

(b) (3 marks) How would you rewrite your function `f1` to calculate the sum of the integers between a second input `lo` and the first input `n`, inclusive? `lo` should be an optional input that is given the value 1 if it is not specified by the user. Your new function should be called `f2`.

My answer:

Like this. The key is the addition of the second input `lo` with a default value of 1 (with the equals sign), and the modification of however you wrote the function to start adding at `lo` rather than 1.

```
f2 <- function(n, lo = 1) {
  sum(lo:n)
}
```

(c) (2 marks) How would you best use your function `f2` to (i) sum the integers between 3 and 6, (ii) sum the integers between 1 and 10? ("Between" means "inclusive" here.)

My answer:

The first one is the obvious

```
f2(6, 3)
## [1] 18
```

but for the second one there is no need to specify that it starts at 1, because it will already start there if you don't say, so that this is better:

```
f2(10)
## [1] 55
```

(d) (3 marks) Suppose you were to run your function `f2` with inputs `lo = 3` and `n = 2`. What do you think should happen in this case? Rewrite your function `f2` to handle cases like this.

My answer:

The obvious answer is that this should be an error, since we are supposed to go from `lo` *up* to `n`. To handle this, add a `stopifnot` first. I called mine `f2a` because I want to run the original `f2` in a moment:

```
f2a <- function(n, lo = 1) {
  stopifnot(lo <= n)
  sum(lo:n)
}
f2a(2, 3)
## Error in f2a(2, 3): lo <= n is not TRUE
```

If you can make a case for some other behaviour, go ahead and do so. The clearer and more convincing your explanation is, the better my chances of being swayed by it. For example, my original `f2` actually still works:

```
f2(2, 3)
```

```
## [1] 5
```

because `3:2` is actually still defined: it counts down rather than up:

```
3:2
```

```
## [1] 3 2
```

and so the answer is $3+2=5$. If this is what you think should happen, then say that you don't need to make any changes to your function, but you will also need to make the case that you know what will happen if you run `f2(2, 3)`. Calling it an error and handling that is probably easier to get marks from.

(e) (3 marks) For the remainder of the question, we go back to using the function `f1` that you defined earlier.

You are given three values of `n` in a vector, like this:

```
ns <- c(4, 6, 10)
```

How would you best run `f1` on all three of these values of `n` at once, arranging the answers back into a vector? Give the code that would do it.

My answer:

The best solution here is a `map`:

```
map_int(ns, ~f1(.))
```

```
## [1] 10 21 55
```

There is, I guess, one point for something like this:

```
c(f1(4), f1(6), f1(10))
```

```
## [1] 10 21 55
```

but this is a "solution" that doesn't scale at all.

(f) (4 marks) Suppose the values of `n` we want to run `f1` on are now in a column of a dataframe like this:

```
d <- tibble(n = ns)
d
```

```
## # A tibble: 3 x 1
##       n
##   <dbl>
## 1     4
## 2     6
## 3    10
```

What are two *different* ways in which we might create a column `first` that contains the results of running `f1` on those three values of `n`?

My answer:

The way you may think of first is `map` inside a `mutate`:

```
d %>% mutate(first = map_int(n, ~f1(.)))
```

```
## # A tibble: 3 x 2
##       n first
##   <dbl> <int>
## 1     4    10
## 2     6    21
## 3    10    55
```

The second way is to recognize that we are running our function once on each row, like we did with taking lots of bootstrap samples, so `rowwise` will also work:

```
d %>%
  rowwise() %>%
  mutate(first = f1(n))
```

```
## # A tibble: 3 x 2
## # Rowwise:
##       n first
##   <dbl> <int>
## 1     4    10
## 2     6    21
## 3    10    55
```

Two points for each of these.

You need the `rowwise`:

```
d %>%
  mutate(first = f1(n))
```

```
## Warning in 1:n: numerical expression has 3 elements: only the first used
```

```
## # A tibble: 3 x 2
##       n first
##   <dbl> <int>
## 1     4    10
## 2     6    10
## 3    10    10
```

The reason is that `f1` only accepts a single number, not a vector of three numbers (as, for example, `sqrt` does). To be more precise, it will accept a vector, but `:` will not (the things on each side of that have to be numbers, not vectors), and so only the *first* number in `n` is used (which is why the three answers come out the same, and we get the warning we did). For the non-rowwise way to work, you would have to write your function `f1` to properly handle more than one value of `n` in the input, and I think it is unlikely that you will manage to do that under exam conditions.

Question 7 (15 marks)

You observe a process that produces values Y from an exponential distribution with rate β. That is to say, the density function of Y is $f(y) = \beta \exp(-\beta y)$ for $y \geq 0$ and 0 otherwise, where $\beta > 0$. The mean of this distribution is $1/\beta$. You want to estimate β using Bayesian methods.

You believe before looking at any data that β is most likely between 0.1 and 0.5, a fact that is summarized by a gamma prior distribution with shape 4.6 and rate 17.2. The prior density is shown in Figure 19. (95% of this distribution is between 0.1 and 0.5.)

You observe the data shown as `my_y` in Figure 20. The maximum likelihood estimate of β is $\hat{\beta} = 1/\bar{y}$, as shown in Figure 21.

In Stan, the exponential distribution is written `exponential()`, and the gamma distribution as `gamma()`. Inside the brackets go the name or value of the one parameter for the exponential distribution and the names or values for the two parameters for the gamma distribution.

(a) (3 marks) Write the `model` section of a Stan program that will sample from the posterior distribution of β.

My answer:

This means specifying the likelihood (distribution of the data as it depends on the parameter) and the prior (distribution of the parameter as it depends on numbers):

```
model {
  // prior
  beta ~ gamma(4.6, 17.2);
  // likelihood
  y ~ exponential(beta);
}
```

If you want to call the observed thing `x` rather than `y`, that's fine, as long as you're consistent later (in your `data` section). Also, if you want to parametrize your prior, fine, but the two numeric values have to go into the data as well. The parameter we're estimating must be called `beta`, though, since that's the name it's given in the question. Be aware that I do not look kindly on copying things verbatim from your notes without any apparent understanding. If I see `poisson` or `weibull` here, I will know that this is what you did. (In any case, in my questions, there is almost never any value in copying word-for-word from my notes.)

The comments are optional, but will help to guide both your thinking and the grader's. It is probably easier to think about the likelihood first, but it makes more sense (in terms of "define before use") to have the prior first in your answer (even if you leave a space for it and write it in once you realize what it has to be).

Question 7 continues… This page: 3 marks.

In code, you would actually have to spell out the word `beta` rather than using the Greek letter β, but I was willing to forgive that in an answer that was otherwise correct.

Grading note: you will lose a point for missing out semicolons, but you should only lose that point once, probably in this part. Likewise, if you mess up the section title and curly brackets, you should lose something for that once, probably a half point, but only once in this question.

(b) (2 marks) Write the `parameters` section of your Stan program.

My answer:

This:

```
parameters {
  real<lower=0> beta;
}
```

The lower limit for `beta` is important, because the sampler must never be allowed to accept negative values for the exponential rate.

If you called your parameter something else, I checked back to the previous part to make sure that what you had here was playing the role of the parameter in your `model` section.

(c) (2 marks) Write the `data` section of your Stan program.

My answer:

```
data {
  real<lower=0> y[5];
}
```

Make sure you count that there were 5 observations in your data, as in Figure 20. The lower limit on `y` means that Stan will check your input data (when you do the sampling) and immediately stop with an error if any of the data values for `y` are negative. This is a plus, but it will still run without.

If you called your data something different earlier, like `x`, make sure you use the same name here. Also, if you gave the prior distribution named parameters, like `rate` and `shape`, these need to go here, and also into the `data` list in the next part (with their values there).

Expect to lose one point for a major error, and another half point for another error. If you have anything right at all, you should get a half point.

(d) (3 marks) Assuming that your Stan program compiles correctly into an object called `expo`, what R code would run it for the data in Figure 20, to obtain a sampled posterior distribution? You may assume that you already have the data stored as in Figure 20. Save the posterior distribution into `expo_fit`.

My answer:

First arrange the data, which is called `my_y` in Figure 20, but needs to be called `y` for the Stan program. Then

```
my_data <- list(y = my_y)
expo_fit <- expo$sample(data = my_data)
```

I am also OK with the `rstan` version:

```
expo_fit <- sampling(expo, data = my_data)
```

I didn't do this in lecture, but you may have found it somewhere else in my materials, so it is good.

The data input to the sampler has to be a list, so you either have to make that first, or make the list as part of the input to the sampler. You cannot input `my_y` directly to the sampler.

I said that the code was already compiled, so you don't need the `cmdstan_model` thing. If it's there, I'll ignore it, but you don't get any credit for it.

Points: if your input to `sample` is not a list but is something sensible, that's 2; if you made a list but messed up some small part of it, that's 2.5.

(e) (2 marks) A summary of the posterior distribution of `beta` is shown in Figure 22. Why does most of the posterior distribution appear to be less than the maximum likelihood estimate shown in Figure 21?

My answer:

The maximum likelihood estimate of `beta` is about 0.427. Most of the prior distribution shown in Figure 19 is less than this, and because the posterior distribution is a combination of the prior and the likelihood, the posterior distribution is mostly below 0.427 as well, though not as much because the data pulls it upward.

Something close enough to that will get the marks.

(f) (3 marks) Suppose you wanted to modify your Stan program to allow estimation of an exponential rate parameter from a sample of any size, and to use a gamma distribution with any parameters as the prior for `beta`. (Call those parameters `theta` and `lambda`.) What *changes* would you make to the Stan code that you wrote earlier?

My answer:

This is a lot like the last bit of the lecture on this material.

This means introducing a sample size (call it say `n`), and parameters `theta` and `lambda` for the prior. All of these are now data, so they need to be declared in the `data` section (before they are used in the case of `n`). My complete code is below:

```
data {
  int<lower=1> n;
  real<lower=0> theta;
  real<lower=0> lambda;
  real<lower=0> y[n];
}

parameters {
  real<lower=0> beta;
}

model {
  // prior
  beta ~ gamma(theta, lambda);
  // likelihood
  y ~ exponential(beta);
```

```
}
```

The changes are:

- in the data section, to declare `n`, `theta`, and `lambda` as integer, real, and real respectively. (The lower bounds for `theta` and `lambda` are optional, since I didn't tell you that they had to be positive, although you might remember from when you first ran into the gamma distribution that its parameters have to be positive.)
- also in the data section, change the declaration of `y` from `y[5]` to `y[n]`
- in the model section, change the prior for `beta` to `gamma(theta, lambda)`.

I tried not to punish you again for any errors you made earlier; if it looked as if you were making some sensible changes to the code you wrote earlier, I was happy with that.

If you were thoughtful enough to write your code earlier to handle any sample size and to allow the parameters to the gamma prior to be input, you have three free points here! (You needed to say that you had done this before, or of course you could write it out again. If you said you had done this before, I went back and checked.)

This was (by design) a question that required some clear thinking. If you made it to the end, and you still seemed to know what you were doing, a tip of the hat to you!