University of Toronto Scarborough
Department of Computer and Mathematical Sciences
STAD29 / STA 1007 (K. Butler), Midterm Exam
February 29, 2019

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 has 34 numbered pages of questions. Check to see that you have all the pages.
In addition, you should have an additional booklet of output to refer to during the exam. Contact an invigilator if you do not have this.
Answer each question in the space provided (under the question).
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 code and output has already been run.

Question 1 (13 marks)

According to Wikipedia, Intensive Care Units in hospitals "…cater to patients with severe or life-threatening illnesses and injuries, which require constant care, close supervision from life support equipment and medication in order to ensure normal bodily functions". A number of patients do not live long enough to be discharged from the Intensive Care Unit (ICU).

A study was carried out to investigate factors associated with a patient living long enough to be discharged from the ICU (or of dying before then). The study collected a large amount of information; we will look only at a few explanatory variables. Our data set is a sample of 200 patients from a much larger number in the original study.

- `id`: identification code for patient
- `sta`: patient's vital status: Lived (until being discharged from the ICU), Died.
- `age`: in years
- `can`: Cancer is part of the patient's present problem: No, Yes
- `cpr`: patient had CPR (cardio-pulmonary resuscitation) before admission to the ICU: No, Yes
- `inf`: probable infection at admission to ICU: No, Yes
- `race`: patient's race: White, Black, Other

The variables above that are actually categorical have been turned into R `factor`s so that they will properly be treated as categorical. The categories are ordered as shown above. Some of the data set is shown in Figure 2.

(a) (2 marks) The response variable is `sta`. How do we know that the logistic regression models that we fit will predict the probability of dying rather than the probability of living? Explain briefly.

My answer: `sta` is categorical with two categories, Lived and Died in that order. The first one is treated as the baseline, and so we predict the probability of the second one, dying.

Saying that dying is the first one alphabetically is wrong for two reasons: (i) the categories are not here ordered alphabetically (see question), and in any case, (ii) it's the *second* category that you need. No points for saying those two things. If you really thought they would be in alphabetical order, you need to say the second one is *living* to be consistent. (That would be one point.)

Your thinking needs to be flexible enough to deal with what is actually happening here. Not everything is going to be exactly the same as your notes.

Also, this is not a survival analysis, since we are not measuring the *length of time* each patient lived for, just whether they lived or died. (The survival analysis question is coming up later, but it is not this one.)

(b) (2 marks) A logistic regression is shown in Figure 3. Based on the output shown there, which explanatory variable would you consider removing first, if any? Explain briefly.

My answer: One of the explanatory variables, `race`, is categorical with three categories, so we should look at the output of `drop1` rather than the output of `summary`. From there, the best explanatory variable to remove first is `can`, with the highest P-value, 0.697.

Question 1 continues… This page: 4 marks.

If you look at `summary` instead, you might be tempted to say that `raceOther` comes out first, but in any kind of regression a categorical variable stays *as a whole* if any of its levels make a difference to the response, and is removed *as a whole* if none of them do. Think about how you would fit a regression wihout `raceOther`: do you include `race` in the model formula or not? Thus, you need to consider `race` as a whole, which the `drop1` output lets you do (see how `race` has 2 df in that table to go with its three categories).

You might suspect that `race` as a whole is going to come out pretty soon (see next part), but the story here is to remove `can` first, re-fit, and *then* think about `race`.

You might think that the high P-value on `raceOther` says to remove `race` as a whole, but it doesn't (necessarily) because `raceBlack` could still be very different from the baseline `raceWhite` and you would want to keep `race` on that basis. (As it turns out, it's not, but the way to discover that is the non-significant `race` on `drop1`, which suggests that after you've dropped `can` there are no significant differences among races and that `race` will be the next thing to come out. You could tell this also by looking at *both* `raceBlack` and `raceOther` and seeing that neither is anywhere near significant. I'm good with that, but you have to look at both.)

All that high P-value on `raceOther` is saying is that there is no significant difference in probability of dying between people of the baseline race White and Other. The biggest race difference is between Black and Other, and the `summary` output doesn't allow us to test that. (It turns out that this difference isn't significant either; we know this because the `drop1` output says that *none* of the races differ in probability of dying, all else equal.)

Extra: the P-values on the `summary` output and the ones in the `drop1` output are similar but not quite the same. `summary` uses the Wald test, and `drop1` (coded this way) uses the likelihood ratio test. For ordinary regression, these are the same, but for generalized linear models such as logistic regression, they are close but not identical. (They are closer the more data you have.)

(c) (3 marks) I used `step` to remove unimportant explanatory variables from my model, obtaining the output shown in Figure 4. I decided to keep `inf` even though its P-value is slightly greater than 0.05. The remaining categorical variables all have two levels. Explain briefly, based on what you know or can guess about patients entering the ICU, why it makes sense that each of the last three Estimates in the Figure (that is, except for the Intercept) is positive.

My answer: We are predicting the probability of dying, so a positive Estimate means that a patient with a larger or non-baseline value of the explanatory variable concerned is more likely to die. Thus:

- An older patient (with larger `age`) is more likely to die than a younger one, which makes sense because an older patient is more likely to have other health problems or to be less healthy generally.
- A patient who had CPR prior to admission to the ICU is (much) more likely to die. This makes sense because CPR is an emergency life-saving procedure, and someone who needs it may not be very healthy (it may only keep them alive for a while).
- A patient who had probable infection at admission to the ICU is more likely to die. This makes sense because if a patient has an infection, it will make it harder to treat whatever else they have, or the infection may compromise that treatment.

Two points for the "more likely to die" in each case, and the third point for a sensible discussion of why, based for example on an older patient, or one who has CPR before, or one who has an infection, being in a worse state of health compared to a younger patient, or one who does not have either of the other risk factors.

(d) (4 marks) We want to obtain predicted probabilities of dying for all combinations of representative values of the explanatory variables in the model shown in Figure 4. Using Figure 5 to help you, give code that will do this. You have to decide what "representative values" means for you. In your code, end by displaying the predictions with the values they are predictions for.

My answer: There are several steps:

- obtain the representative values for each of `age`, `cpr` and `inf` from Figure 5.
- Use `crossing` to make a data frame containing all combinations of these.
- Use `predict` to obtain the predictions. (They will already be probabilities of dying from (a).)

My habit is to use quartiles for quantitative variables (here `age`) and all the categories of a categorical one (the others), but I am willing to entertain anything that looks "representative" in some fashion (for example, if you take the lowest and highest ages):

```
ages <- c(46.75, 72)
cprs <- c("No", "Yes") # capital letters important!
infs <- c("No", "Yes") # order not important provided you get them both
new <- crossing(age=ages, cpr=cprs, inf=infs)
p <- predict(icu.2, new, type="response")
cbind(new, p)
```

```
##     age cpr inf          p
## 1 46.75  No  No 0.09357968
## 2 46.75  No Yes 0.17170188
## 3 46.75 Yes  No 0.34524029
## 4 46.75 Yes Yes 0.51425833
## 5 72.00  No  No 0.17283618
## 6 72.00  No Yes 0.29554942
## 7 72.00 Yes  No 0.51624519
## 8 72.00 Yes Yes 0.68180517
```

One point for each of:

- obtaining representative values for each of the three remaining explanatory variables;
- making all combinations of them;
- getting the predictions;
- gluing them onto the data frame of combinations so that we can see what they are predictions for.

`type="probs"` is for a multi-category response (`polr` or `multinom`), not for a logistic regression where you need `typwe="response"`.

If you do something like this:

```
p <- predict(icu.2, new, type="response")
cbind(new, p)
```

```
##     age cpr inf          p
## 1 46.75  No  No 0.09357968
## 2 46.75  No Yes 0.17170188
## 3 46.75 Yes  No 0.34524029
## 4 46.75 Yes Yes 0.51425833
## 5 72.00  No  No 0.17283618
## 6 72.00  No Yes 0.29554942
## 7 72.00 Yes  No 0.51624519
## 8 72.00 Yes Yes 0.68180517
```

you'll get predictions all right, but they'll be predictions for *all the observations in the original data frame*, not the representative values in the Figure. 2 points.

I realize I could have given you more space to write here!

(e) (2 marks) The predictions I made are shown in Figure 6. Describe the effect of an increase in age on probability of dying, all else equal. Explain briefly.

My answer: Compare two rows of Figure 6 that have the *same* values for `cpr` and `inf` but *different* values for `age`. It doesn't matter which "same" values you choose.

For example, I picked rows 1 and 5 of the predictions (with the other variables being `No`), and the probability of dying went up from 0.09 to 0.17. A similar kind of increase will show up whichever values you choose for `cpr` and `inf`.

I want to see how you made "all else equal": that is, that you specifically compared something with `cpr` and `inf` the same and `age` different. If it seemed clear to me that you were doing this (eg. by telling me which rows you were looking at), I was happy.

This is the same issue as the `age` Estimate being positive, and the conclusion is an exact repeat of what we saw earlier, with the Estimate for `age` being positive: that is to say, two ways of looking at the same thing.

Extra: of course, you can see what representative ages I chose, and then reverse-engineer what I did (for (d)).

Extra 2: some people called this predicted probabilities "P-values". I reserve P-value for the thing coming out of a test, where you look to see whether it's less than 0.05. These are not that. Call them "values of p" if you must, or, better, "predicted probabilities" (which is, admittedly, a bit long to write on an exam).

Question 2 (16 marks)

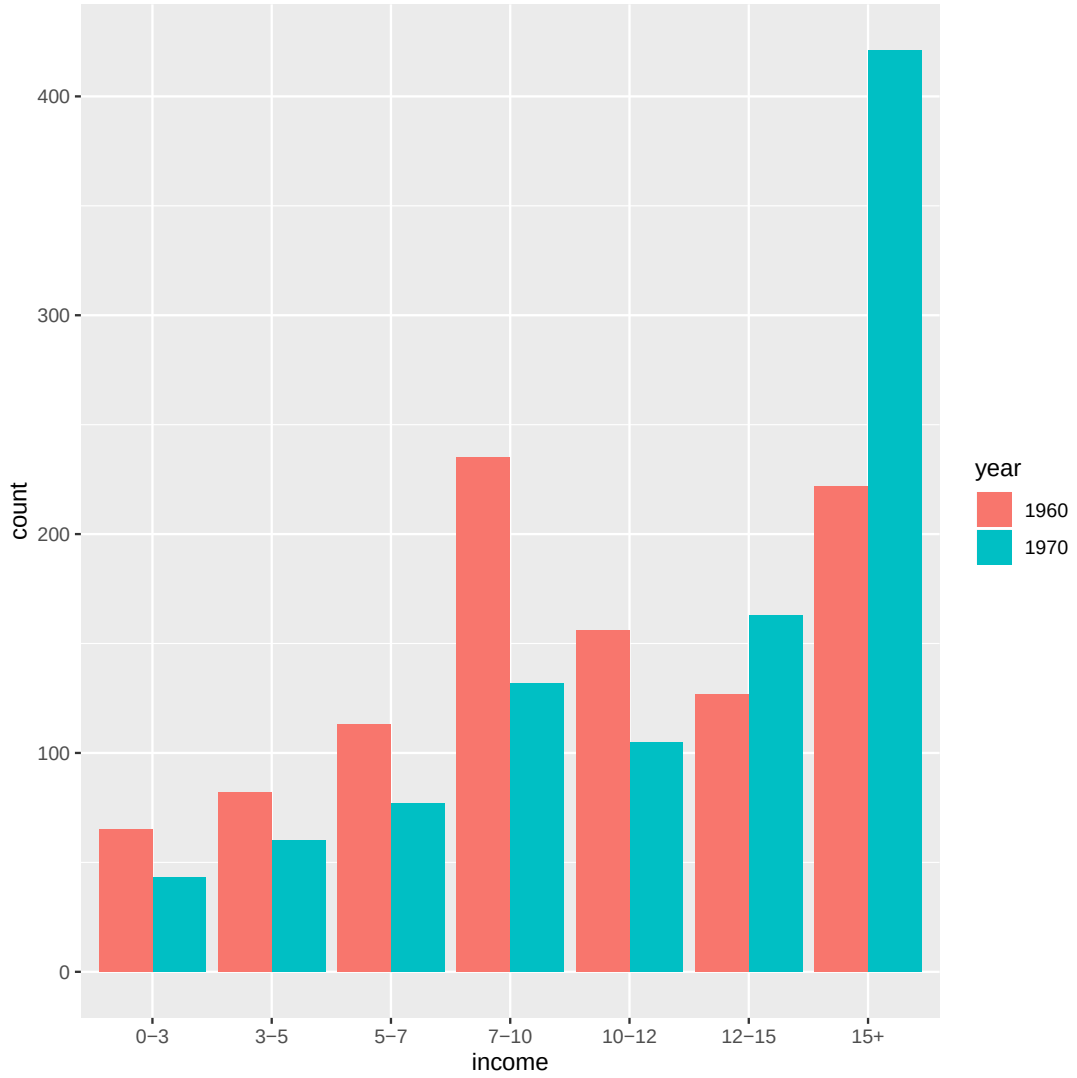
Two surveys were taken, one in 1960 and one in 1970, of different randomly selected groups of 1000 people. One of the questions in the survey asked respondents to say in which of a number of categories their annual income fell. The incomes are in thousands of dollars; in 1960, \$15,000 was a good income! Our question of interest is to say what happened to income over that time period. The data are shown in Figure 7. Note the types of the variables. For example, `ord` means "ordered factor".

(a) (2 marks) Describe briefly an appropriate graph for these data. (I want a description, not code.)

My answer: There are two categorical variables, income and year, so a grouped bar chart would be appropriate, with the counts as the heights of the bars.

Extra: the description is all I need; the code would look like this:

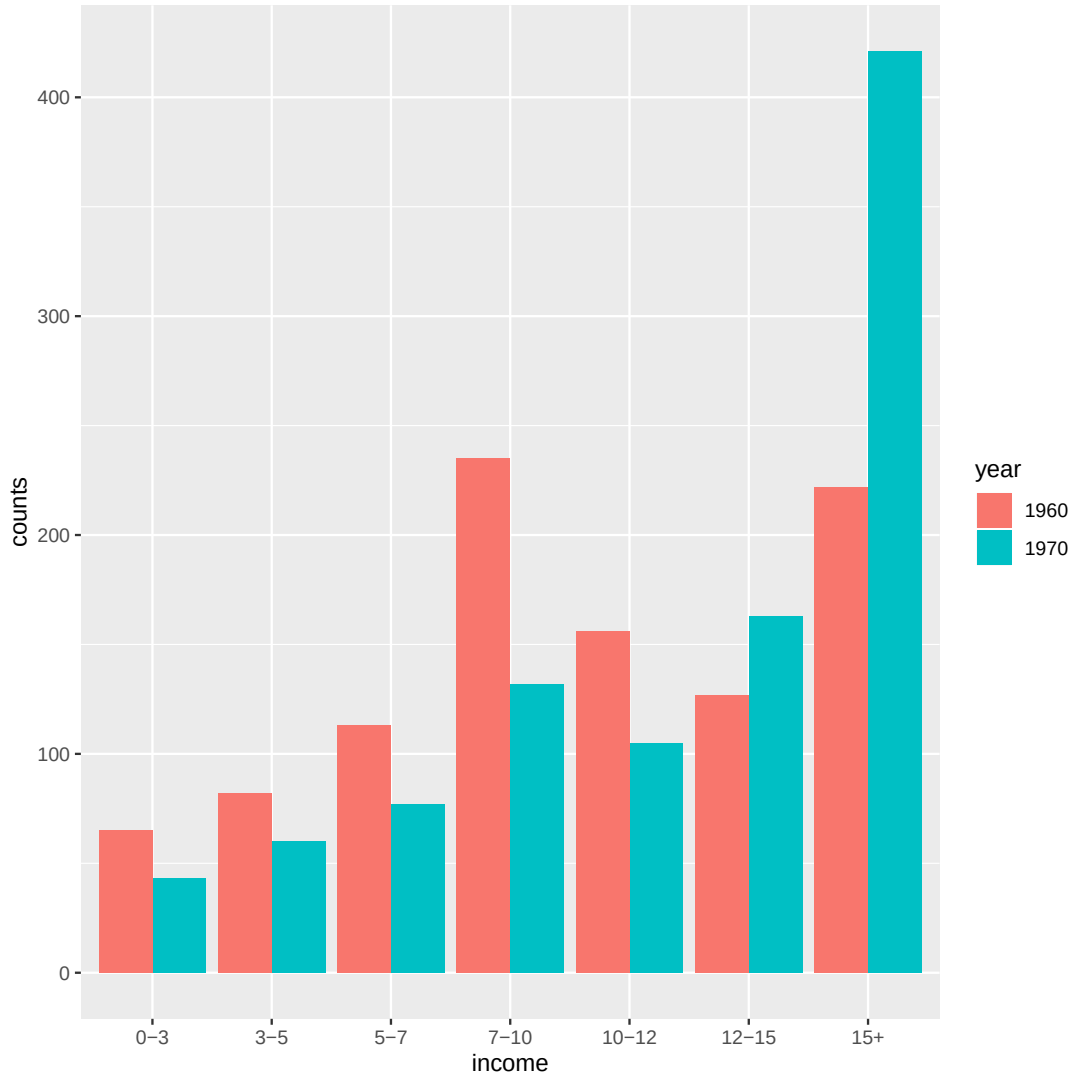
```
ggplot(incomes, aes(x=income, fill=year, weight=counts)) +
    geom_bar(position="dodge")
```



There is one new thing here: the `weight` aesthetic will add up all the frequencies for each income-year combination (only one for each here) and make the bar that height. If I had omitted that, all the bars would have been of height 1, since there is only one *row of the data frame* for each combination. `geom_bar` doesn't know about multiple observations per row of the data frame unless you tell it.

Another way to make the same graph is to use `geom_col` instead of `geom_bar`. This takes a `y` which is the heights of the bars you want to use:

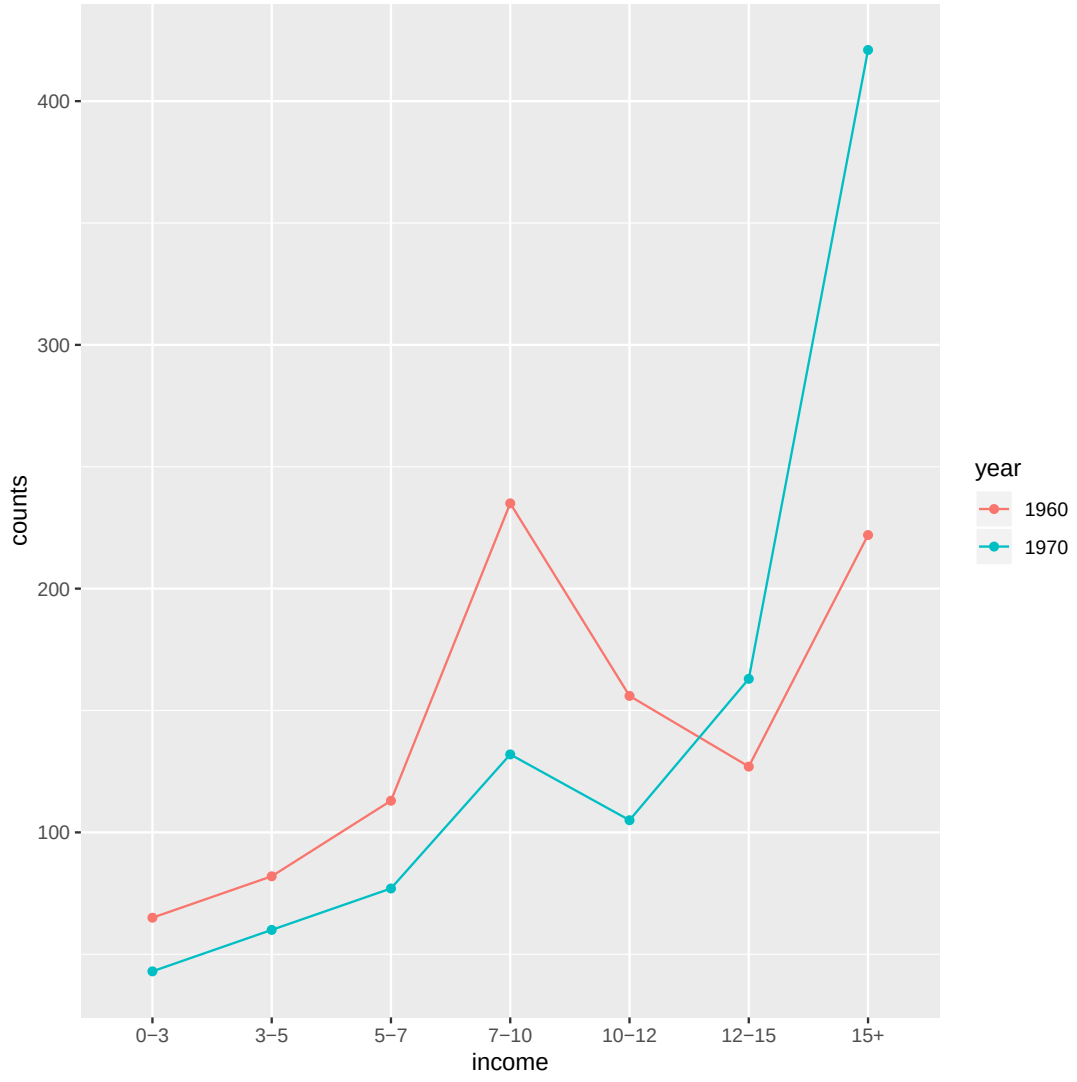
```
ggplot(incomes, aes(x=income, fill=year, y=counts)) +
    geom_col(position="dodge")
```



This is all equivalent, since we have categorical variables, to the plot we did for the miners. We could work out the proportions of people in each income category for each year, and then plot those. But as long as we work out "out of" each year (rather than out of each income category), the graph we get will be the same: both years have the same number of people, so you can look at proportions out of 1000 or counts and it will look (relatively) the same.

I guess this way will also make sense:

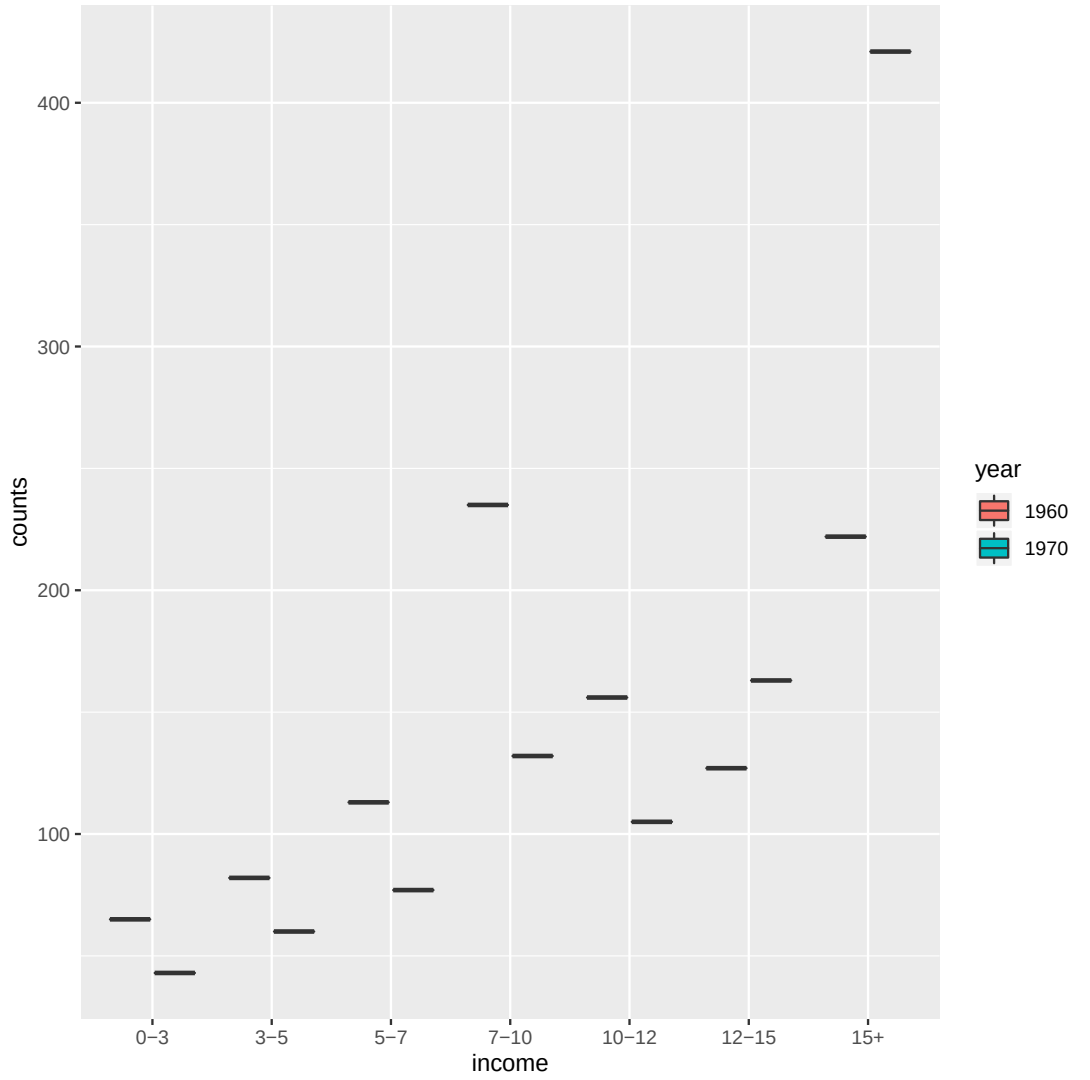
```
ggplot(incomes, aes(x=income, colour=year, y=counts, group=year)) +
    geom_point() + geom_line()
```



I'm not even sure what you would call this: a "point-and-line graph"? It looks to me like the kind of thing a businessperson would like. I'm not wild keen on it myself: I'd rather see the frequencies as bars, since they really are counts of observations. But, as a suggestion for a graph here, no problem: two points.

If you take the variable types literally, you'll see one quantitative variable (the count) and two categorical ones, and that will lead you to grouped *boxplots*. What does that look like?

```
ggplot(incomes, aes(x=income, fill=year, y=counts)) +
    geom_boxplot()
```



This doesn't work because there is only one observation (a count of a number of people) per year-income group combination, and boxplots work when you have a *number* of values per group, so that you have some kind of distribution to display. You don't have that here.

I didn't ask you for code because of the extra complications that you haven't seen. So I asked you for a description: a grouped bar chart (with, best, income group as `x` and year as `fill`, since there are more income categories), or a plot of proportions in each income category separately for each year, which you could also do by facets.

Extra: here's something rather amusing:

```
incomes %>% uncount(counts)
```

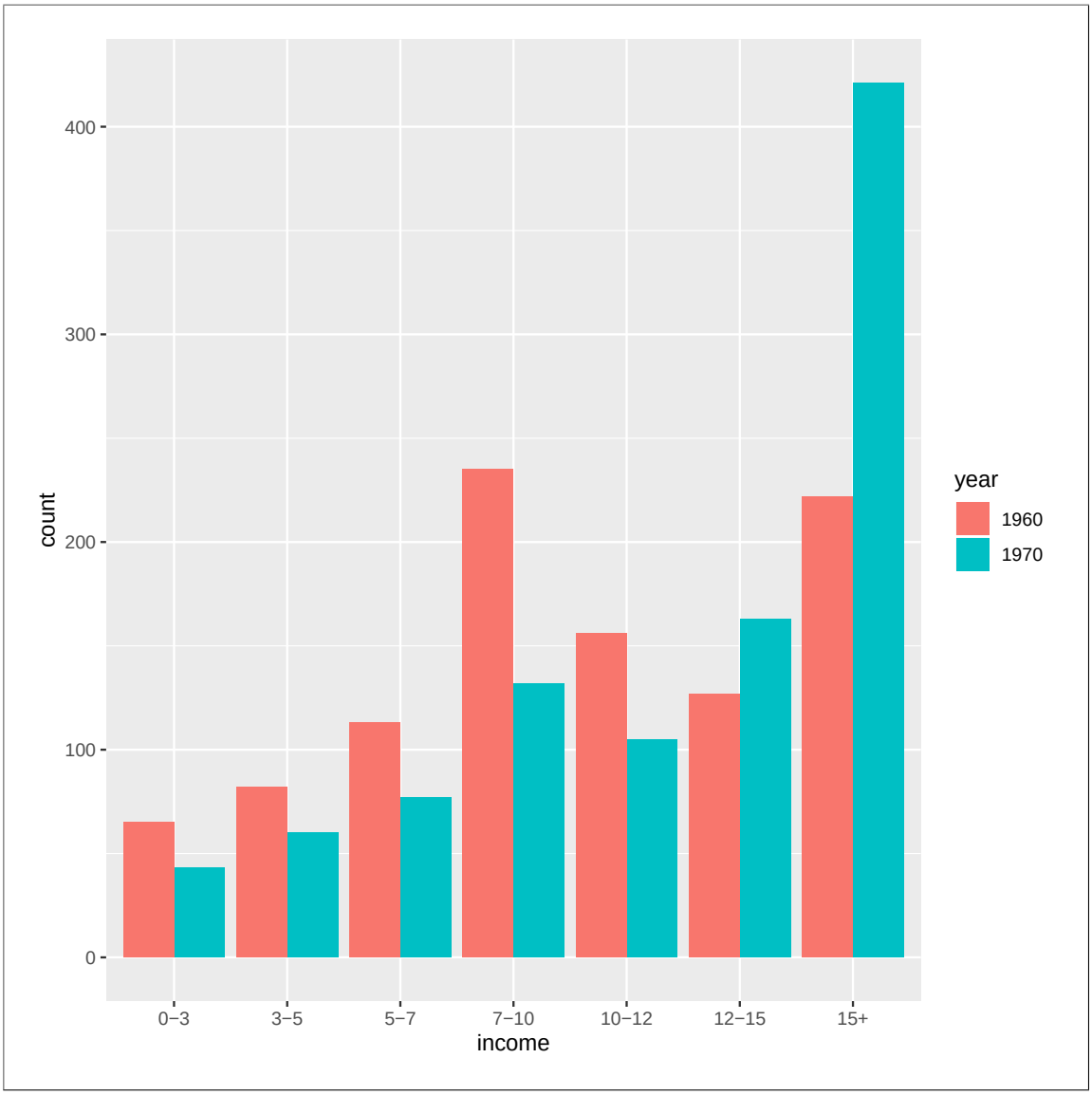
```
## # A tibble: 2,001 x 2
```

```
##    year  income
##    <fct> <ord>
##  1 1960  0-3
##  2 1960  0-3
##  3 1960  0-3
##  4 1960  0-3
##  5 1960  0-3
##  6 1960  0-3
##  7 1960  0-3
##  8 1960  0-3
##  9 1960  0-3
## 10 1960  0-3
## # ... with 1,991 more rows
```

The produces a data frame with one row per observation and therefore 2,000 rows altogether (well, actually, 2001; not quite sure what happened there). Since you would take *this* data frame and `count` income to get back to where we started, the inverse operation is rather amusingly called "uncount".

Having gotten to here, we are in exactly the situation where we would normally use a grouped bar chart, which we can now make without extra difficulty:

```
incomes %>% uncount(counts) %>%
    ggplot(aes(x=income, fill=year)) +  geom_bar(position="dodge")
```



(b) (3 marks) Give code for fitting an appropriate model, named `income.1`, for predicting income category from year. (You may assume any necessary packages are loaded with `library()`.)

My answer: This is a categorical response, so some kind of logistic regression is called for. Note that income has seven categories (more than two) and they come in order, so we need `polr` rather than anything else.

There is one more thing: we need to specify how many observations each row of the data frame summarizes. This is the same idea as the coal miners example from class. Note that the response variable `income` is already an ordered factor (`ord` at the top of the column in

Figure 7), so you don't have to do anything with it. In fact, wrapping it in `factor` is *an error*, because it shows a lack of understanding: why would you make a factor out of something that is *already* a factor?

Hence:

```
income.1 <- polr(income~year, data=incomes, weight=counts)
```

and not

```
income.1a <- polr(factor(income)~year, data=incomes, weight=counts)
```

If you're going to copy code from somewhere in your notes, *make sure you change the variable name* from what is there. The column of frequencies is called `counts` in this data set, so that is what you need with `weight`. Using something else rather betrays that you are copying without thinking.

(c) (2 marks) In your code of the previous part, did you have a `weight`? Explain briefly why you need it (if you had it) or why you don't need it (if you didn't have it).

My answer: This is how we specify that each row of the data frame represents more than one person (in fact, the number of people in `counts`), so I needed to have it. Otherwise, `polr` would have treated each row of the data frame as *one* observation, and we would have had a very wrong answer.

Points are for *answer plus explanation.* Don't expect any points if you don't have an explanation.

If you didn't get the previous part, that makes this one harder. But you can still get two points here: figure out what `weight` does (in any context where you've seen it), and then try to explain how that does or does not apply to the code you just wrote. I would work with `weight` as you have seen it in the various flavours of logistic regression, or even as in a weighted linear regression. Show me that you know what it does, and how you think it does or does not apply here.

(d) (2 marks) Some output from your fitted model is shown in Figure 8. The `tidy` output at the top is basically the same as the `summary` output for the model. Is the distribution of incomes the same or different for the two years? Explain briefly.

My answer: Look at the `drop1` output: the P-value for `year` is very small, so `year` definitely has an effect on income: that is to say, the distribution of incomes is different for the different years.

I was looking for a P-value here, to answer the question "are they different?". The issue of how they're different is coming up. (If the years were not significantly different, there would be no differences worth talking about that would generalize to a population.)

Extra: I gave you the `tidy` output rather than the `summary` output, because the `summary` output contains exactly the code you were asked to supply in an earlier part!

As I said in class, the `summary` (or `tidy`) output is not very illuminating, so you don't actually need to look at it at all. The only thing that's of any interest here is the top line, for `year1970` (since `year` is categorical): the `statistic` is a z, so its value being 9.81 is unusually large, which suggests that 1970 is different from 1960, as confirmed by the `drop1` table, which is really the informative one.

(e) (3 marks) Give code to obtain predicted probabilities of a person falling in each income category, for each year in the data set, and to display your predictions next to the values they are predictions for.

My answer: Make a data frame of years to predict for, and call it `new`:

```
new <- tibble(year=c("1960", "1970"))
```

then feed that into `predict`:

```
p <- predict(income.1, new, type="probs")
```

then display that beside the years these are predictions for:

```
cbind(new, p)
```

```
##   year        0-3        3-5        5-7      7-10     10-12     12-15       15+
## 1 1960 0.07319675 0.09254759 0.11750129 0.2093851 0.1351699 0.1345620 0.2376374
## 2 1970 0.03442580 0.04788022 0.06908339 0.1533551 0.1275407 0.1592583 0.4084564
```

One point for each of these, correctly done.

This is a bit of a repeat of the prediction you did earlier, but this is a simpler one (with a much simpler `new`), so I figured it was OK to have you do this one. Plus, this one is `probs`, so you'll need to recognize why it's that and the other one was `response`.

If you include `income` in your `new`, it'll still work, though you get some extra output, so I make that a half-point deduction. These are the same years that are in the original data frame, so failing to define `new` at all will still kind of work, but if you are to save yourself from a one-point deduction for that, you'll need to convince me that you know what you're doing.

Likewise, if you include the whole `years` column in your `new`, you'll get several copies of 1960 and of 1970, so you'll get several copies of your predictions. I prefer several copies of the right thing to not getting the right thing at all, but I like it better if you get the right thing just once:

```
new <- tibble(year=incomes$year)
```

```
p <- predict(income.1, new, type="probs")
cbind(new, p)
##    year        0-3        3-5        5-7      7-10     10-12     12-15
## 1  1960 0.07319675 0.09254759 0.11750129 0.2093851 0.1351699 0.1345620
## 2  1960 0.07319675 0.09254759 0.11750129 0.2093851 0.1351699 0.1345620
## 3  1960 0.07319675 0.09254759 0.11750129 0.2093851 0.1351699 0.1345620
## 4  1960 0.07319675 0.09254759 0.11750129 0.2093851 0.1351699 0.1345620
## 5  1960 0.07319675 0.09254759 0.11750129 0.2093851 0.1351699 0.1345620
## 6  1960 0.07319675 0.09254759 0.11750129 0.2093851 0.1351699 0.1345620
## 7  1960 0.07319675 0.09254759 0.11750129 0.2093851 0.1351699 0.1345620
## 8  1970 0.03442580 0.04788022 0.06908339 0.1533551 0.1275407 0.1592583
## 9  1970 0.03442580 0.04788022 0.06908339 0.1533551 0.1275407 0.1592583
## 10 1970 0.03442580 0.04788022 0.06908339 0.1533551 0.1275407 0.1592583
## 11 1970 0.03442580 0.04788022 0.06908339 0.1533551 0.1275407 0.1592583
## 12 1970 0.03442580 0.04788022 0.06908339 0.1533551 0.1275407 0.1592583
## 13 1970 0.03442580 0.04788022 0.06908339 0.1533551 0.1275407 0.1592583
## 14 1970 0.03442580 0.04788022 0.06908339 0.1533551 0.1275407 0.1592583
##          15+
## 1  0.2376374
## 2  0.2376374
## 3  0.2376374
## 4  0.2376374
## 5  0.2376374
## 6  0.2376374
## 7  0.2376374
## 8  0.4084564
## 9  0.4084564
## 10 0.4084564
## 11 0.4084564
## 12 0.4084564
## 13 0.4084564
## 14 0.4084564
```

(f) (2 marks) The predictions from your code are shown in Figure 9. Describe briefly how the probabilities are changing between 1960 and 1970.

My answer: The probabilities of the lower incomes are going down and the probability of (especially) the highest income is going way up. Something like that. Try to summarize rather than comparing all 7 one by one (that's why I said "briefly").

Say something about what's going up *and* what's going down. The probabilities for each year add up to 1, so if any of them are going up, there must be some going down as well.

(g) (2 marks) Would you say, according to the fitted model, that incomes between 1960 and 1970 are typically going down, staying the same or going up? Explain briefly.

My answer: They're typically going up, for two reasons: the probability of an income being small is going down, and the probability of an income being large is going up. These are both pointing to an overall increase.

You can also talk about the "estimate" from the `tidy` (`summary`) output if you want; this is positive for 1970 compared to 1960 which means, if I have it right, that the whole distribution is shifted towards the higher end (more people at the high end and fewer at the low end). I wasn't completely convinced by this answer, especially if you tried to make it into a slope (of what?), but you were probably good with it unless you said something that I disagreed with.

There's a certain overlap between parts (f) and (g). If you wanted to say in (f) that incomes are overall going up, and you said somewhere how you know that, then you are good for the points here as well. I am checking back to see that you said something sensible somewhere. (This is particularly true if I wanted to give you less than 2 in (g); in that case, I looked back at (f) as well to see if there was anything in there that would count for (g).)

Question 3 (14 marks)

100 patients with a certain disease are randomly allocated to one of four treatments, labelled A, B, C, and D. Each patient is monitored periodically until the disease comes back (labelled "recurrence" in the data set), or until the study ended and no recurrence had been seen until that point. Some of the data are shown in Figure 10. Our intention is to do a survival analysis with these data, to see which treatment is most effective at delaying recurrence, but with the data as they were given to us, we have some work to do first. For each patient, we have: the treatment they were on, whether or not recurrence occurred (the disease came back), the month, day and year that they were enrolled into the study, and the month, day and year of the last followup (the last time they were seen by a doctor). The data frame as read in from the file is called `disease`.

(a) (1 mark) When did the study end? Explain (very) briefly.

My answer: Look at the patients for whom no recurrence occurred: their last followup was always on February 29, 2020: today!

Or you can eyeball what seems to be the latest date of any observation.

I wanted something to say how you know, but if you just gave me a date I couldn't very well give you nothing, so I gave you 0.5. I meant to add something like "this is the data set you are given, which the researchers say is complete", since you could otherwise argue that the study will continue. Asking for the study to continue until recurrence has been seen for everyone doesn't work, however, since for some people the disease might never recur (and you can't know that for certain unless you follow such people for the rest of their lives). Studies of this kind always end on some date, with (in this case) the recurrence of the disease "never seen so far"; after all, the researchers involved have to write a paper at some stage! This is why Cox had to handle censored observations in his proportional-hazards model; they make the computation a lot messier (which fortunately we don't have to worry about), but real-life studies have them, and they contain information.

Extra: real studies are not as simple as this one. Patients can be censored for any number of reasons, only one of which is "the study ended and the event never happened". Patients can drop out of a study because, say, they move house and can no longer visit the hospital to be observed, or they no longer want to take part in the study (a standard ingredient of consent

forms is "you can withdraw from the study at any time"), or they might even *die of something else* unrelated to the disease that we are interested in here.

(b) (3 marks) Give code to create the date of enrollment as an R date.

My answer: I like using `unite` to combine the three parts of the date into one (as a piece of text), and then using something like `mdy` from `lubridate` to turn the text into a date:

```
disease0 %>% unite(enrolled_text, enrolled.m, enrolled.d, enrolled.y) %>%
    mutate(enrolled=mdy(enrolled_text)) %>%
    select(enrolled_text, enrolled)
```

```
## # A tibble: 100 x 2
##    enrolled_text enrolled
##    <chr>         <date>
##  1 Apr_30_2018   2018-04-30
##  2 Nov_18_2018   2018-11-18
##  3 Oct_27_2018   2018-10-27
##  4 May_14_2019   2019-05-14
##  5 May_9_2018    2018-05-09
##  6 Aug_3_2019    2019-08-03
##  7 Mar_22_2019   2019-03-22
##  8 Jul_17_2019   2019-07-17
##  9 Feb_1_2019    2019-02-01
## 10 Aug_23_2018   2018-08-23
## # ... with 90 more rows
```

You don't need the `select`; that is for me since I have a lot of columns.

You can combine the columns in whichever order you like, as long as you use a function like `mdy` that is consistent with the order you used. The first input to `unite` is the name of the new column, and the others are the names of the columns to be united, which will then disappear. Thus, for example, you can say

```
disease0 %>% unite(enrolled_text, enrolled.y, enrolled.m, enrolled.d) %>%
    select(enrolled_text)
```

```
## # A tibble: 100 x 1
##    enrolled_text
##    <chr>
##  1 2018_Apr_30
##  2 2018_Nov_18
##  3 2018_Oct_27
##  4 2019_May_14
##  5 2018_May_9
##  6 2019_Aug_3
##  7 2019_Mar_22
##  8 2019_Jul_17
##  9 2019_Feb_1
## 10 2018_Aug_23
## # ... with 90 more rows
```

and then use `ymd` since that is now the order of the pieces in the date.

There is no such function as `combine`. There is `paste` and there is `str_c`, either of which will work to combine the month, day and year into one piece of text, on which you can then use

`mdy` to make actual dates. I was more sympathetic if you showed me what you were trying to do, as opposed to asserting that there was a function called `combine`, or that `+` could be used with text (that is Python, not R). Telling me that you were trying to combine the month, day and year into a piece of text, then you would use `mdy` on that text, might even get you two points if you explained yourself clearly enough.

(c) (3 marks) After running your code of the previous part, repeating the process on the followup dates, and removing columns no longer needed, the data is as shown in Figure 11. A survival analysis needs "survival times": that is, the number of days between enrolment into the study and last follow up. Give code to obtain these from the data shown in Figure 11.

My answer:

There are several ways to do this. The easiest is to remember that dates are stored as numbers of days since Jan 1, 1970, so that you can obtain the number of days between two days simply by subtracting them:

```
disease1 %>% mutate(days=last_follow-enrolled)
```

```
## # A tibble: 100 x 5
##    trt   status        enrolled   last_follow days
##    <chr> <chr>         <date>     <date>      <drtn>
##  1 C     recurrence    2018-04-30 2019-02-02  278 days
##  2 B     recurrence    2018-11-18 2019-04-20  153 days
##  3 D     recurrence    2018-10-27 2019-02-03   99 days
##  4 C     no recurrence 2019-05-14 2020-02-29  291 days
##  5 B     recurrence    2018-05-09 2019-03-05  300 days
##  6 B     recurrence    2019-08-03 2020-01-24  174 days
##  7 A     recurrence    2019-03-22 2019-08-19  150 days
##  8 C     recurrence    2019-07-17 2019-09-27   72 days
##  9 D     recurrence    2019-02-01 2019-05-03   91 days
## 10 C     recurrence    2018-08-23 2019-01-22  152 days
## # ... with 90 more rows
```

The units are uncertain, but since this gives me days, you would also get days and the coding is thus correct.

Or you can borrow the idea from class of turning the time between enrolment and last followup into a period and dividing it by `days(1)`, which, in principle, will get a fractional number of days (but these are all whole numbers of days):

```
disease1 %>% mutate(days=as.period(enrolled %--% last_follow)/days(1))
```

```
## estimate only: convert to intervals for accuracy
```

```
## # A tibble: 100 x 5
##    trt   status        enrolled   last_follow  days
##    <chr> <chr>         <date>     <date>      <dbl>
##  1 C     recurrence    2018-04-30 2019-02-02  277.
##  2 B     recurrence    2018-11-18 2019-04-20  154.
##  3 D     recurrence    2018-10-27 2019-02-03   98.3
##  4 C     no recurrence 2019-05-14 2020-02-29  289.
```

```
##  5 B     recurrence    2018-05-09 2019-03-05  298.
##  6 B     recurrence    2019-08-03 2020-01-24  173.
##  7 A     recurrence    2019-03-22 2019-08-19  150.
##  8 C     recurrence    2019-07-17 2019-09-27   70.9
##  9 D     recurrence    2019-02-01 2019-05-03   93.3
## 10 C     recurrence    2018-08-23 2019-01-22  152.
## # ... with 90 more rows
```

I'm actually not sure about this, since the number of days should not be a fraction. But it's what the notes say, so it will do.

I think this way is better: turn them into `durations`:

```
disease1 %>% mutate(days=as.duration(enrolled %--% last_follow)/ddays(1))
```

```
## # A tibble: 100 x 5
##    trt   status        enrolled   last_follow  days
##    <chr> <chr>         <date>     <date>      <dbl>
##  1 C     recurrence    2018-04-30 2019-02-02    278
##  2 B     recurrence    2018-11-18 2019-04-20    153
##  3 D     recurrence    2018-10-27 2019-02-03     99
##  4 C     no recurrence 2019-05-14 2020-02-29    291
##  5 B     recurrence    2018-05-09 2019-03-05    300
##  6 B     recurrence    2019-08-03 2020-01-24    174
##  7 A     recurrence    2019-03-22 2019-08-19    150
##  8 C     recurrence    2019-07-17 2019-09-27     72
##  9 D     recurrence    2019-02-01 2019-05-03     91
## 10 C     recurrence    2018-08-23 2019-01-22    152
## # ... with 90 more rows
```

This gives the same as the subtraction, so I think it is right.

Keep in mind: if you do it the subtraction way, the later date (last followup) goes first, but if you use `as.period` or `as.duration`, it's "from the earlier date to the later one", so the enrolment date goes first.

Extracting the number of completed days will not work. Displaying the period (or duration) shows why:

```
disease1 %>% mutate(days=as.period(enrolled %--% last_follow))
```

```
## # A tibble: 100 x 5
##    trt   status        enrolled   last_follow days
##    <chr> <chr>         <date>     <date>      <Period>
##  1 C     recurrence    2018-04-30 2019-02-02  9m 3d 0H 0M 0S
##  2 B     recurrence    2018-11-18 2019-04-20  5m 2d 0H 0M 0S
##  3 D     recurrence    2018-10-27 2019-02-03  3m 7d 0H 0M 0S
##  4 C     no recurrence 2019-05-14 2020-02-29  9m 15d 0H 0M 0S
##  5 B     recurrence    2018-05-09 2019-03-05  9m 24d 0H 0M 0S
##  6 B     recurrence    2019-08-03 2020-01-24  5m 21d 0H 0M 0S
##  7 A     recurrence    2019-03-22 2019-08-19  4m 28d 0H 0M 0S
##  8 C     recurrence    2019-07-17 2019-09-27  2m 10d 0H 0M 0S
##  9 D     recurrence    2019-02-01 2019-05-03  3m 2d 0H 0M 0S
```

```
## 10 C     recurrence    2018-08-23 2019-01-22  4m 30d 0H 0M 0S
## # ... with 90 more rows
```

There is a number of months also. Extracting `day` of the period gives you how many days it is *in addition to* the number of months.

(d) (2 marks) The data after we have finished tidying it is shown in Figure 12. What code will create a suitable response variable `y` for a survival analysis of these data?

My answer: This needs `Surv` with the survival times (that is, the number of days until recurrence or censoring), and something that will be `TRUE` if recurrence occurred (that is, the event occurred):

```
y <- with(disease, Surv(days, status=="recurrence"))
y
##   [1] 278  153   99  291+ 300  174  150   72   91  152  300   93  105  277  152
##  [16]  70  285  154  300  164  291  284    4  147  300  300  159  277+ 152  296
##  [31] 163  291  300  109  292  108  159+ 154  300+ 293  289  257  148  296  156+
##  [46]  18  300   76  295  284  300  264  101  133  300  300    6  281  137  164
##  [61] 300  276  295  296  152   80   96  300   80  300  300  291  287  290+ 300
##  [76]  72  285  220   75  105  298  148  300  300  255+   6  284  154   73  300
##  [91] 162    7    2  300  151  300  300  281  155   90
```

The dollar sign way works but is uglier:

```
y1 <- Surv(disease$days, disease$status=="recurrence")
y1
##   [1] 278  153   99  291+ 300  174  150   72   91  152  300   93  105  277  152
##  [16]  70  285  154  300  164  291  284    4  147  300  300  159  277+ 152  296
##  [31] 163  291  300  109  292  108  159+ 154  300+ 293  289  257  148  296  156+
##  [46]  18  300   76  295  284  300  264  101  133  300  300    6  281  137  164
##  [61] 300  276  295  296  152   80   96  300   80  300  300  291  287  290+ 300
##  [76]  72  285  220   75  105  298  148  300  300  255+   6  284  154   73  300
##  [91] 162    7    2  300  151  300  300  281  155   90
```

The second input to `Surv` is something that is TRUE if the event occurred and FALSE otherwise, so you need to say what value of `status` goes with recurrence. `status` by itself is neither true nor false, so it will not do as a second input to `Surv`. (This caught a lot of people.)

Since this one was only two points, you get 1 if you made one error (I felt bad giving you only 1 if you wrote `=` rather than `==`, so that is 1.5), and 0.5 if you had `Surv` somewhere but more than one error.

(e) (2 marks) In Figure 13, a Cox model is fitted and some output shown. Is there evidence of any difference among the treatments? Explain briefly.

My answer: The best way is to look at the `drop1` output at the bottom of the Figure. This gives a test specifically for treatment. Since the P-value of 0.007183 is small, there are differences in survival (that is, times to recurrence) among the treatments.

Since treatment is the only explanatory variable, an overall test of the model will also test

for differences among treatments. The three tests at the bottom of the `summary` output will therefore also work here. (If I had had another explanatory variable such as age, however, this would not have worked and the only way would have been to look at the `drop1` output.) Pick one of the likelihood ratio test (P-value 0.007), Wald test (0.007) or score test (0.005) to come to the same conclusion. (In this case, the `drop1` test is one of these, but I forget which.)

Looking at the `summary` is not helpful (but see below) because all it tells you is how the treatments compare with the baseline treatment A; we don't have anything like Tukey that will compare each pair of treatments (properly, allowing for the multiple testing).

All this does not tell us anything about *how* they are different, however.

If you insisted on looking at the `summary` anyway, and you mentioned how treatment A was the baseline and things were being compared with that, and said something sensible, you got 1 point. If you looked at the `summary` and said something like "treatment B is significant" without saying what it was different *from*, you get 0.5. This is the same principle as ANOVA: look at the overall test first, to see if anything is going on (which is what we are doing here). Later, we think about which treatment is best, now that we know they are not all the same.

Looking at both the `drop1` and the `summary`, particularly if you talk about the latter first, might cost you a half point, since you don't need to talk about the `summary` at all.

(f) (3 marks) Figure 22 shows estimated survival curves. Which treatment is best at delaying recurrence? Cite *two* pieces of evidence, using this Figure and Figure 13, to support your assertion.

My answer: The best treatment is D.

To be best at delaying recurrence, the survival curve has to be up and to the right. This is the red one, which is stratum 1. Looking at the data frame I called `new` above the plot, the treatments are listed in *reverse* alphabetical order, so stratum 1 is treatment D.

This is a bit sneaky, I suppose, but I want you to get into the habit of finding the number of the stratum with the best survival, and then looking to see which treatment that stratum actually was. For example, the strata could be *combinations*, such as a treatment and sex combination, and then you would definitely have to look back to see which stratum is which combination. This was a slightly artificial way of getting you beyond "it's the first stratum, so it must be the first treatment", which is much too simple-minded in general and I didn't want you to get away with that here.

The second piece of evidence for this is in the `summary` output in Figure 13. The treatment with the most *negative* estimate (called `coef` in the table) is the one that makes the event (recurrence) more likely to take a long time to happen. The most negative one, -0.15, again goes with treatment D. (Remember that treatment A is the baseline and has a `coef` of zero.)

Question 4 (16 marks)

Arthritis sufferers often feel pain in their joints (knees, elbows, wrists) which interferes with their daily lives. Three competing treatments for joint pain are compared in terms of their mean time to pain relief in patients with osteoarthritis. Because the investigators hypothesize that there may be a difference in time to pain relief in men versus women, they randomly assign each of 15 participating men to one of the three competing treatments and randomly assign each of 15 participating women to one of the three competing treatments. Thus there are 5 men and 5 women assigned to each treatment. Participating men and women do not know to which treatment they are assigned. They are instructed to take the assigned medication when they experience joint pain and to record the time, in minutes, until the pain subsides. A shorter time to pain relief is better. The data are shown in Figure 14 as data frame `joint`.

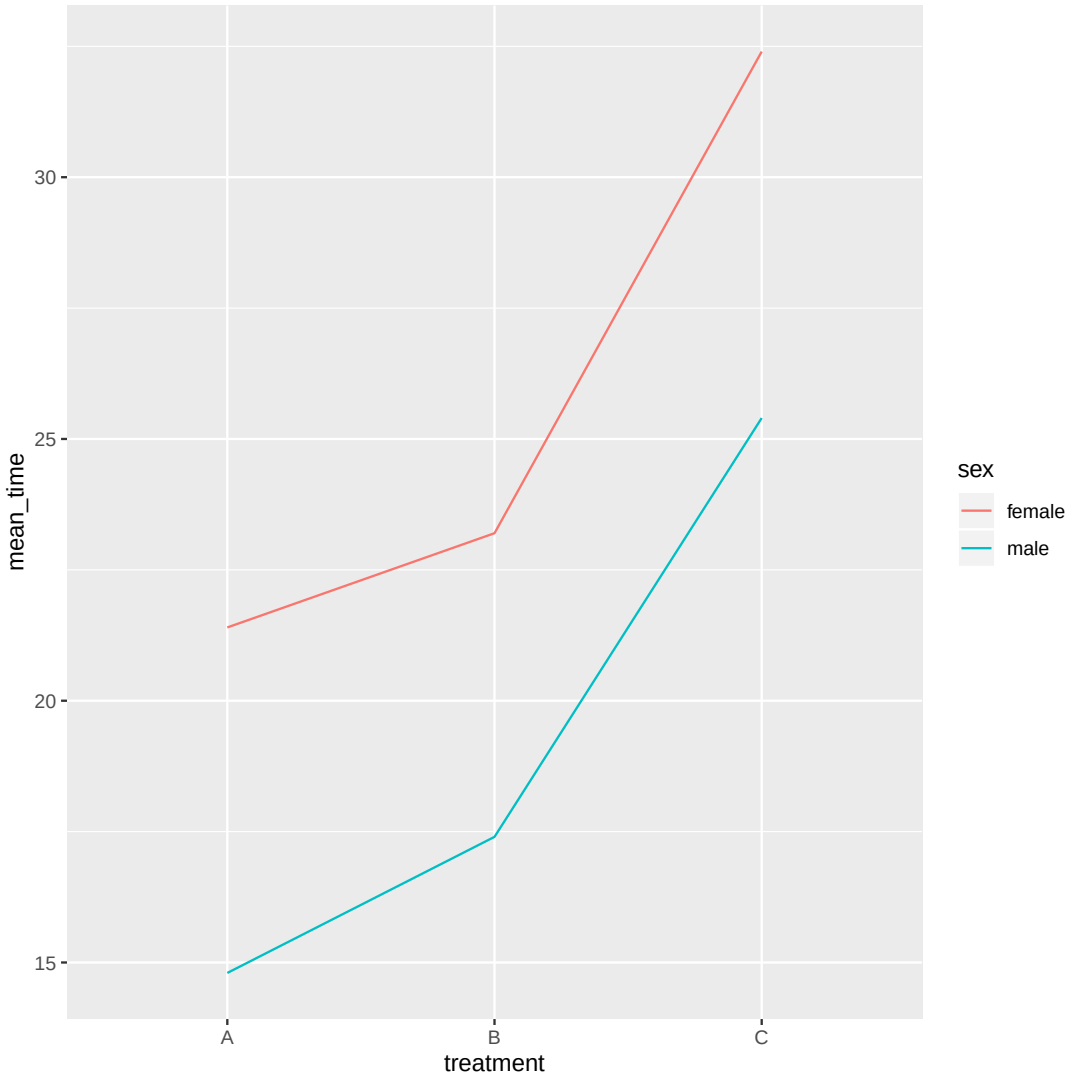
(a) (3 marks) There are two categorical explanatory variables. One way to understand the nature of their effect on time to pain relief is to draw an interaction plot. Give code to do this. Hint: your code could do a calculation first.

My answer: I am trying to guide you towards calculating the mean for each combination first:

```
joint %>% group_by(treatment, sex) %>%
  summarize(mean_time=mean(time)) -> summary_table
```

One point for that (you have known about group-by and summarize since early in C32). Then, to make the plot, use this data frame:

```
ggplot(summary_table, aes(x=treatment, y=mean_time,  colour=sex, group=sex)) +
    geom_line()
```



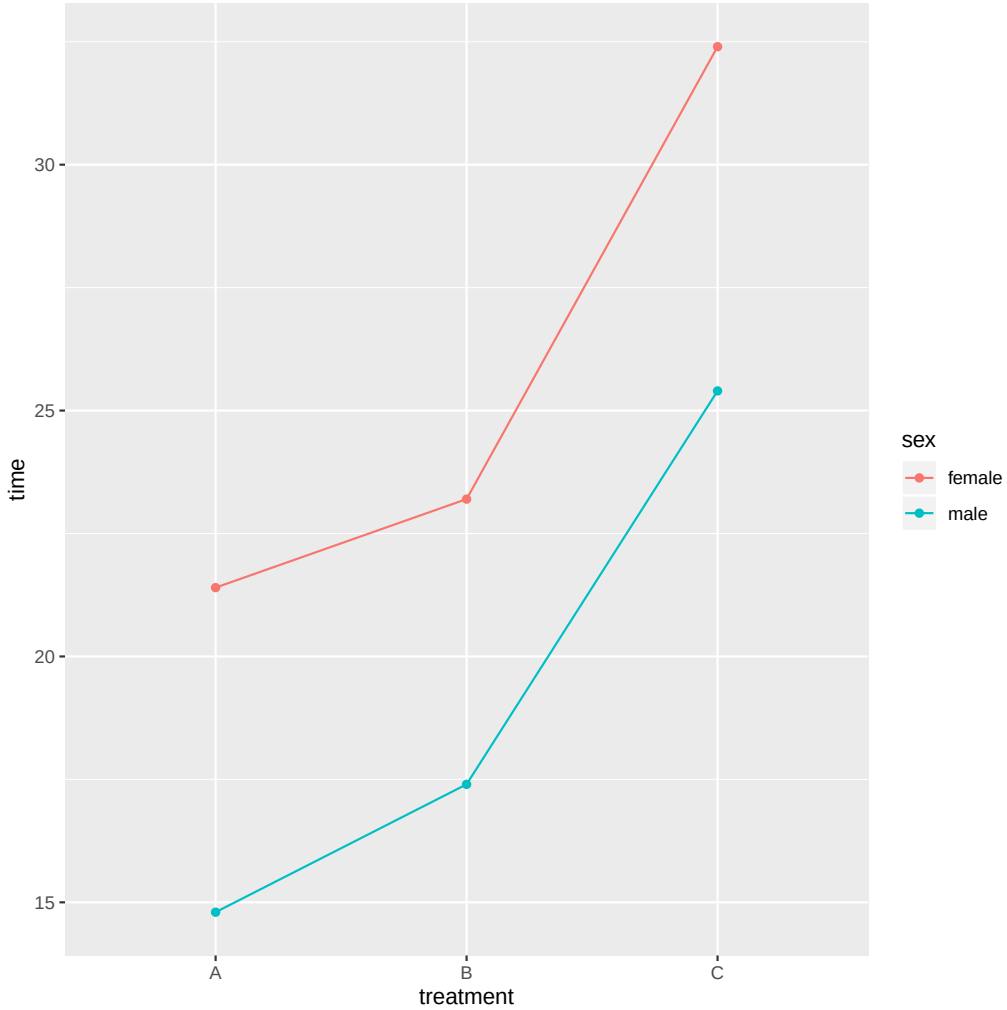
Two points for that. The key is to have both `colour` *and* `group` and for them both to be the same, so that the right things get joined by lines.

Variations that are also good:

- adding `geom_point` to plot the means for each combination as well
- switching around treatment and sex. (My take is that with three treatments and two sexes, having the treatment on the x-axis is better, but I don't insist on that here.)
- rather than saving the data frame of means, you can pipe it straight into `ggplot`.

- Doing it the slick way, having `ggplot` find the means for you. I always have to look this up:

```
ggplot(joint, aes(x=treatment, y=time, colour=sex, group=sex)) +
    stat_summary(fun.y = mean, geom = "point") +
    stat_summary(fun.y = mean, geom = "line")
```



(or omit the points). These last two lines can be copied as is from the auto noise example in the lecture notes. (This is a rare example where copying something verbatim from the lecture notes is actually helpful to you.) Three points for the whole thing if you do it this way.

(b) (2 marks) My interaction plot is shown in Figure 23. What do you conclude from it? Explain briefly.

My answer: The two traces are almost exactly parallel, so I expect there to be no significant interaction.

No need to talk about a time effect or a treatment effect yet (though no penalty if you do). The first order of business is to say whether we expect an interaction, since whether or not there is one affects what we do the rest of the way.

A few people said that the traces are not parallel. I think that these people are expecting entirely too much to be looking for the lines to be more parallel than this; as I see this one, there's no way that the lines are *significantly* non-parallel, although of course you'd need to see how much variability there is to be sure. I really don't think you can support non-parallel lines, but if the rest of the discussion is sensible (eg. it follows from a conclusion that the lines are not parallel), I gave you a point.

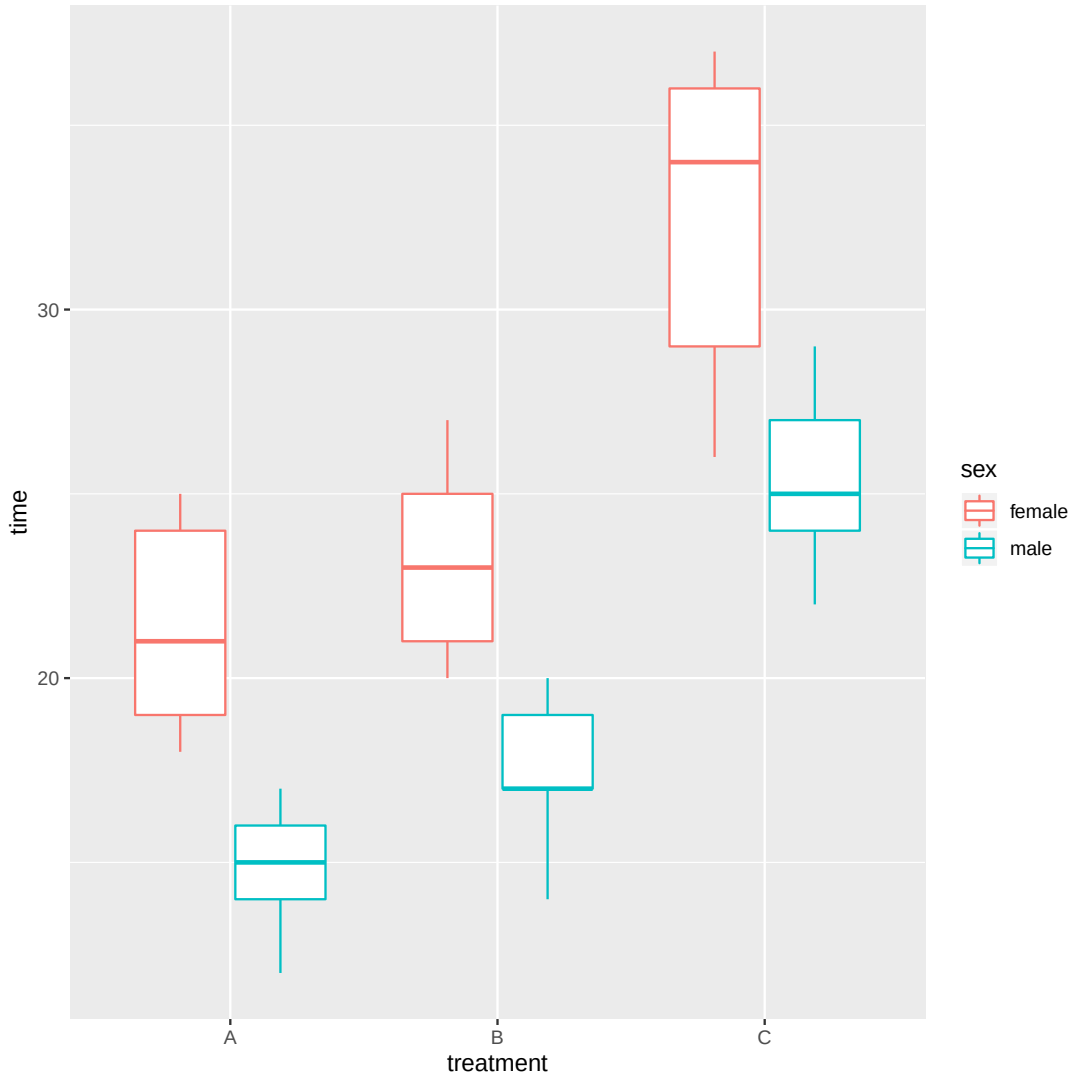
Some people spelled it "parrell". I thought this was a singer.

(c) (3 marks) What other plot could you draw with this data set? Describe and justify briefly one advantage *and* one disadvantage of your proposed plot, compared with the interaction plot.

My answer: With one quantitative variable and two categorical variables, a grouped boxplot (or, possibly, sets of side-by-side boxplots in facets). One point.

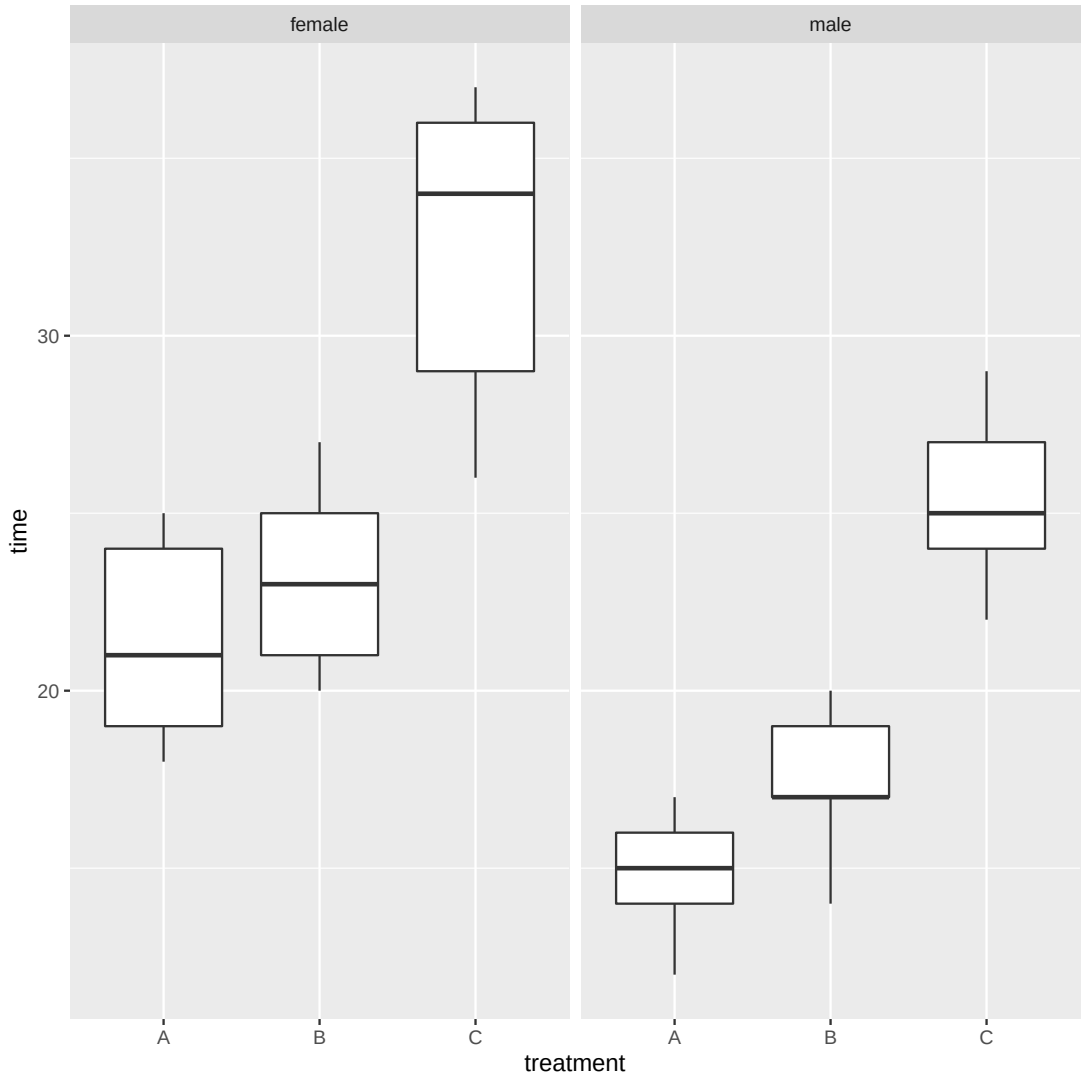
Here's what the plots actually look like:

```
ggplot(joint, aes(x=treatment, y=time, colour=sex)) + geom_boxplot()
```



or

```
ggplot(joint, aes(x=treatment, y=time)) +
    geom_boxplot() + facet_wrap(~sex)
```



An advantage of a grouped boxplot is that you see other features of the distributions apart from just the means (which is all the interaction plot shows), such as the spread or the shape/skewness/outliers. If you are assessing assumptions, not just looking at means, this is the kind of plot you want.

A disadvantage is that grouped boxplots make it difficult to judge whether there is an interaction; the interaction plot includes *only* the information you need to judge interaction. If you work at it a bit, you can get this out of (for example) a grouped boxplot, but you have to look at the medians and try to ignore everything else.

(d) (3 marks) Two analyses of variance are shown in Figure 15. What do you conclude from `joint.1`? Is it a good idea to do the analysis `joint.2`? Explain briefly.

My answer: In `joint.1`, test the interaction and see that it is not significant. (One point.) We should remove the interaction. (One point). Model `joint.2` is appropriate because it no longer contains the interaction. (The last point.) A rather quick three.

(e) (3 marks) Figures 16 and 17 show two possible further analyses. Which one of these is more appropriate, and what do you conclude? Explain briefly. If you think neither of these analyses is appropriate, explain briefly why.

My answer: Figure 16 is simple effects (of treatment for each sex). This would be appropriate if the interaction had been significant. Figure 17 contains two separate Tukey analyses, one for treatment and one for sex. This is appropriate; since there is no interaction, the two explanatory variables act independently and you can talk about "the" effect of treatment regardless of sex and "the" effect of sex regardless of treatment. One point for preferring the Tukey for a good reason, or for *not* doing simple effects for a good reason.

The Tukey analysis, Figure 17, says that treatments A and B are not significantly different, but that C is significantly different from both. Looking at the Tukey confidence intervals (or at the interaction plot), you see that C has a *longer* mean time to pain relief, so it is *worse* than A and B. The bottom table says that the sexes differ (which we already knew since there are only two of them here) but it does tell us that males experience pain relief quicker than females do. (This you could also see from the interaction plot.)

If you had mistakenly looked at the simple effects, you would in fact have come to the same conclusion, twice: for females, treatment C is worst and the others are not significantly different, and for males, exactly the same. This is a hint that the simple effects are not the thing to look at; you would look at them if the interaction had been significant, and then the comparison of treatments would have been *different* for males and females. (That's what an interaction *is*: the effect of treatment is different for males and females.) This wouldn't have given you a comparison between males and females, though; for that, you could do the simple effects the other way around to ask "given the treatment, how do males and females compare?", and you would have found that males consistently experience quicker pain relief than females, regardless of treatment.

(f) (2 marks) Which treatment or treatments would you recommend? Is that different for males and females? Explain briefly.

My answer: Looking at the Tukey, C is significantly worse than A and B, which are not significantly different from each other. Thus recommend "either A or B" for any patient, regardless of sex.

If you thought the simple effects were the way to go, you'll come to the same conclusion here, albeit with a bit more work. It is, as discussed above, the same for males and females because there is no interaction; the treatment and sex effects are independent.

Question 5 (14 marks)

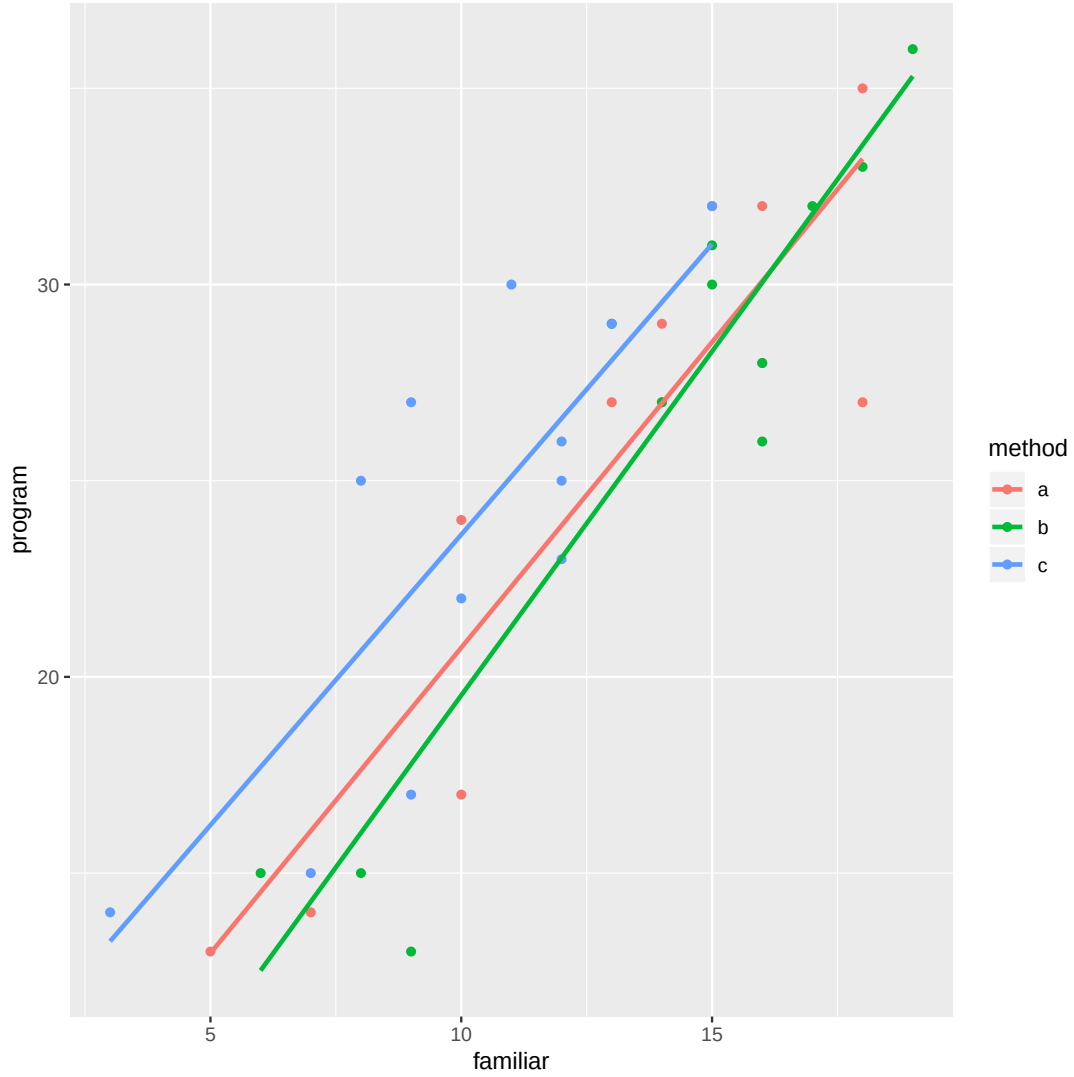
To assess the relative merits of three methods of instruction (labelled `a, b, c`) for elementary computer programming, a curriculum researcher randomly selected 12 fifth graders from each of three elementary schools in a certain school district. Each group, within the setting of its home school, then received a six-week course of instruction. At the end of the six weeks, each student did a test to see how well they had learned the prescribed elements of the subject matter. However, two of the schools (the ones at which methods `a` and `b` were taught) were in more affluent neighbourhoods, and so, before the course began, each student in each of the schools was given a test of basic computer familiarity as well.

The data are shown in Figure 18.

(a) (3 marks) A graph is shown in Figure 24. Give the code that was used to draw the graph.

My answer: This is a scatterplot with familiarity as x-variable, programming test score (response) as y-variable, and with the points distinguished by teaching method. On the plot, we have the points plus the regression lines (without the grey "envelopes"), which leads us to this:

```
ggplot(prog, aes(x=familiar, y=program, colour=method)) +
    geom_point() + geom_smooth(method="lm", se=F)
```



Be careful to include all the pieces, especially the two things inside `geom_smooth`.

(b) (2 marks) An analysis of covariance is shown in Figure 19. What is the one principal thing that this analysis tells you **about the data**?

My answer: The interaction is not significant, so the regression lines on Figure 24 are not significantly different from parallel. (One point.) That means that the predicted increase in programming score per unit increase in familiarity score is the same for all three teaching methods. Or, more simply, the slopes of all three regression relationships are the same, regardless of teaching method. (The second point.)

I bolded "about the data" in the question to remind you to say something about programming scores and familiarity scores and the relationship between them for the different methods. There is very little value in saying "the interaction is not significant" here; to get to a useful conclusion, one that somebody can act upon, you need to get from there to "the lines are parallel", and then from there to something about what was actually observed.

Extra: if you look back at the plot, you'll see that the lines do differ slightly in slope, but not very much given the amount of variability of the points around the lines. This is why the slopes came out not significantly different.

(c) (2 marks) A second analysis of covariance is shown in Figure 20. Is this an appropriate analysis to conduct? Explain briefly. If it is appropriate, what do you conclude from it?

My answer: This is an appropriate analysis; the interaction was not significant, so we have removed it. One point.

Both the level of familiarity with computers and also the teaching method have an impact on the programming score. One more point.

At this point, we cannot say more.

(d) (2 marks) What does Figure 24 suggest about the likely reason for the significance or non-significance of `method` that you observed earlier? Explain briefly.

My answer: There was a significance difference among the methods. On the graph, I think the lines for methods A and B are very close, but the line for method C looks at least a little different from them. So my guess is that method C is significantly better than methods A or B (which are probably not significantly different from each other).

If you somehow failed to find any difference among the methods in the previous part, do your best to rationalize that from the graph. If you do that reasonably, you can get full marks for this part.

(e) (3 marks) Another test is shown in Figure 21. What does this one say? Why do you think this conclusion is different from what you have seen elsewhere in this question? What made it come out different in the way it did? Explain briefly.

My answer: The ANOVA here says that there is *no* difference among the three methods. One point.

This is different from what we have seen elsewhere because familiarity with computers was not included in the analysis, and should have been, because students who were more familiar with computers also did better at programming, regardless of which method they were on. A second point.

The mean and SD table above the ANOVA says why it came out this way. If you just look at programming scores by method, they are all about the same (so the ANOVA was non-significant). However, the students who did method C were *less* familiar with computers than the students who did the other methods. They had, in other words, more to learn; the fact that they ended up doing about as well at programming indicates that method C helped them to learn *more* in the six weeks. These students began in a worse place, but ended up in about the same place, than the students on the other methods. Something from here for the third point.

(f) (2 marks) What do you think would be a statistically better way to design this study? Why do you think it was done the way it was? Explain briefly.

My answer: It would be better to test all three methods in all three schools, since there might be a difference among schools as well as a difference in familiarity with computers among students at different schools. One point.

My guess as to why it was done as it was: available resources in schools. There might have been only one teacher or only one classroom available at each school, so only one of the methods could be used at each school. This, or anything sensible, nets the other point.

Extra: testing each method at each school would have given us a more complicated analysis of covariance, with two factors (school and method) as well as one covariate (familiarity). But there is no additional problem: run it as an `lm` and look at the ANOVA (or at `drop1`). This would separate the effect of school from the effect of method, and could thus be expected to give us a clearer conclusion.

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