University of Toronto Scarborough
Department of Computer and Mathematical Sciences
STAD29 (K. Butler), Midterm Exam
March 4, 2023

Aids allowed (on paper, no computers):

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

This exam has 7 numbered pages of questions plus this cover page.
In addition, you have an additional booklet of Figures to refer to during the exam.
The maximum marks available for each part of each question are shown next to the question part.
If you need more space, use the last page of the exam. Anything written on the back of the page will not be graded.
You may assume throughout this exam that the code shown in Figure 1 of the booklet of Figures has already been run.
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.

1. A study was done to investigated the effectiveness of a new teaching method, called PSI, in economics. This was a comparative study with a control group. Four variables were measured:

   - `grade_improved`: whether the student's exam grades were improved during the study period (yes or no), as compared to before the study period, response
   - `psi` whether the student had been exposed to PSI during the study period (yes or no). The `yes` group is the treatment group, and the `no` group is the control group.
   - `tuce` measure of student's academic ability at the start of the study period
   - `gpa` student's grade point average at the start of the study period.

   Some of the data are shown in Figure 2. One of the authors on this study was named Spector, hence my use of the name `spector` for the dataframe and models.

   (a) [2] Why would logistic regression be suitable here? Explain briefly.

   My answer:

   The response variable `grade_improved` is categorical with two categories (`yes` and `no`). That is to say, all we know about each student is that their exam grades either improved over the course of the study or they did not.

   (b) [2] Some analysis is shown in Figures 3 and 4. Why do you think I used `update` to fit the model `spector.2`? Explain briefly.

   My answer:

   My actual reasoning was that it was a small change from the first model `spector.1`, that was rather long to write, so I decide to use `update` because it would be much easier to say "take out just (the non-significant) `tuce` from the first model", as opposed to typing it out again. The other alternative, copying, pasting and editing, I always find a bit dangerous, because here I'd have to remember to take out `tuce`, change the model number and *not change anything else.*

   Say something here about `update` being more concise because we are making a small change (taking out one of the three explanatory variables) to a longish model that also has the `family` stuff in it. Or, if you prefer, that it is more reliable than copying, pasting, and then editing to physically remove `tuce` from the code, while using `glm` again.

   The point of the question is "why did I use `update` rather than `glm` again?", so show that you know what was driving my decision. Taking out a non-significant explanatory variable was part of the story, but the *way* I chose to do it was the other part of the story.

(c) [2] The variable `tuce` that was removed from the logistic regression was a measure of the student's academic ability. You might expect such an explanatory variable to have an impact on whether or not the student's exam grades improved during the study period. Why do you think it would have been removed from this logistic regression?

My answer:

`tuce` is a measure of academic ability; something else that is also a measure of academic ability is `gpa`, and so you might expect them to be correlated, and we therefore do not need both of them in the model, because they are (more or less) saying the same thing.

Aside: We didn't talk about multicollinearity in connection with logistic regression, but the same issues apply here as they do in regular regression: there are the same problems when explanatory variables are correlated, in that you do not get a good sense of which of them are important. End of aside.

A second hint of this is that when both explanatory variables are in the model, the P-value of `gpa` is 0.025, but when we take `tuce` out, the P-value of `gpa` drops to 0.012, half of what it was before. (Not as dramatic as some we have seen, but dramatic enough, and worth looking out for.)

There are other arguments that I could be swayed by: for example, `tuce` was measured at the beginning, so it might not have had much to do with improvement (eg. if a student already had high academic ability, there is not much room to improve).

Extra 1: I have the data, so I can check the correlation between `gpa` and `tuce`:

```
spector
```

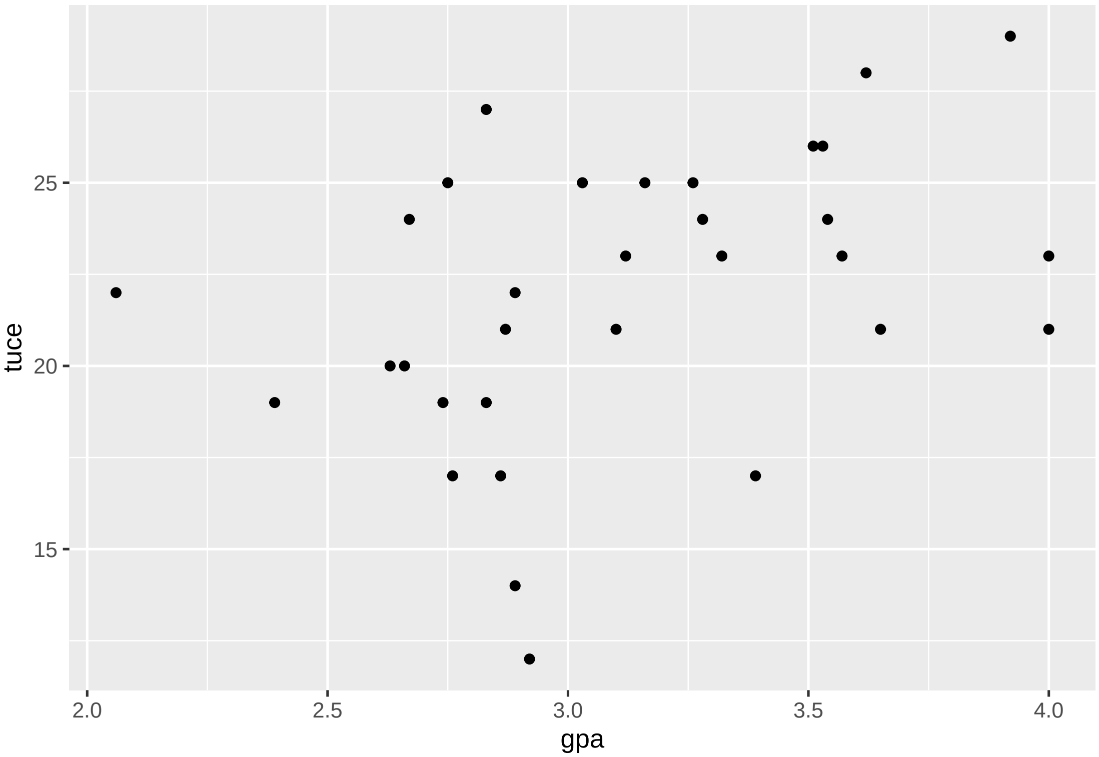
```
## # A tibble: 32 x 4
##    grade_improved psi    tuce   gpa
##    <chr>          <chr> <dbl> <dbl>
##  1 no             no       20  2.66
##  2 no             no       22  2.89
##  3 no             no       24  3.28
##  4 no             no       12  2.92
##  5 yes            no       21  4
##  6 no             no       17  2.86
##  7 no             no       17  2.76
##  8 no             no       21  2.87
##  9 no             no       25  3.03
## 10 yes            no       29  3.92
## # i 22 more rows
```

```
with(spector, cor(gpa, tuce))
```

```
## [1] 0.3869863
```

Actually not that large:

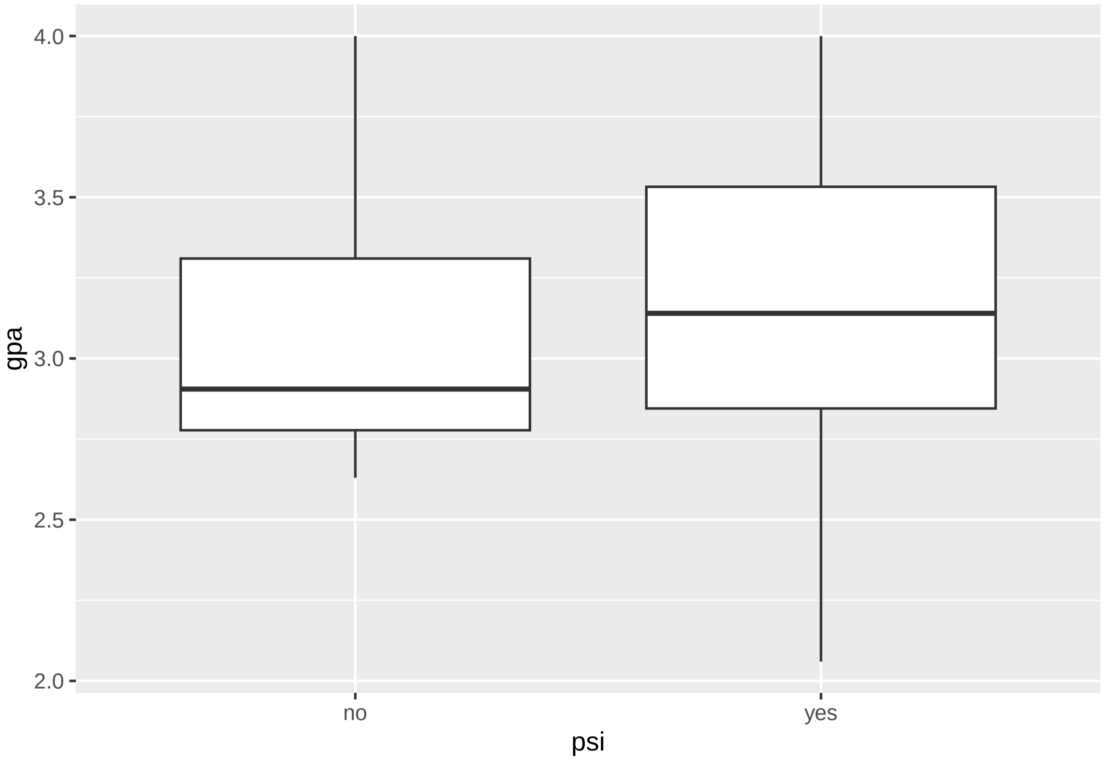
```
ggplot(spector, aes(x = gpa, y = tuce)) + geom_point()
```



Some correlation, but not as much as I was expecting, though it does seem to be true that high `gpa` goes with high `tuce` at least.

Extra 2: We would *not* expect `gpa` to be correlated with `psi`. This looks like a designed experiment, so we would expect the students who were exposed to the PSI instruction to be chosen at random, and thus whether or not they did would be unrelated to `gpa`. We can't really talk about a correlation with a categorical variable, but we can draw a graph (a boxplot) and see if there is any relationship:

```
ggplot(spector, aes(x = psi, y = gpa)) + geom_boxplot()
```



The students who did the PSI instruction had a slighter higher median GPA, but the amount of overlap suggests that this difference was just the way the randomization came out, rather than being any indication of bias in choosing who got the PSI instruction and who did not.

(d) [3] From the model `spector.2` in Figure 4, interpret the numbers in the Estimate column (not including the intercept). Hint: unless you are very careful, you will have to make sure not to over-interpret these numbers.

My answer:

The easier answer is this:

Both estimates are positive, so:

- a student who was exposed to PSI had a higher probability of having improved exam grades, compared to a student who was not exposed to PSI (all else equal, that is, for the same GPA). Not being exposed to PSI is the baseline.
- a student with a higher GPA had a higher probability of improved exam grades compared to one with a lower GPA (all else equal, that is, if `psi` was the same).

That is entirely sufficient as an answer.

If you want to interpret the numbers themselves, you have to say what the numbers represent changes in. This is not probability, as you might have guessed, but *log-odds*, which we discussed briefly in the snow day lecture. So:

- a student who is exposed to PSI, as opposed to one who is not, has a 2.338 higher log-odds of improving on exams.
- a one-point increase in GPA is associated with a 3.063 increase in log-odds of improving exam grades.

An additional clue here is that numbers like 2.338 and 3.063 cannot possibly be changes in probabilities for one-unit changes in anything. That ought to be enough to dissuade you from saying more than you can actually justify. Don't expect to get full marks if you say the "easy answer" but then continue with something that is not correct. It is better to stop, rather than continue with something that is not true.

Extra: gpa is playing the role of a "covariate" or variable you are adjusting for. It is not something you can control, but is something that makes a difference to the outcome (whether or not the student's exam grades improve), and you want to account for it along with assessing the effect of the PSI intervention (which is something you *can* control). What the Figure is saying is that there is a positive effect on exam grade improvement of the intervention, *once you have accounted for the effect of gpa* on exam grade improvement. This is a little awkward to say, because the response is a categorical "did exam grades improve or not". Not accounting for gpa would be a mistake, for one because it is significant, but also for two because if you take it out, the relationship between improvement on exams and the intervention will be a good deal less clear than it is now:

```
spector.3 <- update(spector.2, .~. - gpa)
summary(spector.3)
```

```
##
## Call:
## glm(formula = factor(grade_improved) ~ psi, family = "binomial",
##     data = spector)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.6094     0.6324  -2.545   0.0109 *
## psiyes        1.8971     0.8317   2.281   0.0225 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 41.183  on 31  degrees of freedom
## Residual deviance: 35.342  on 30  degrees of freedom
## AIC: 39.342
##
```

```
## Number of Fisher Scoring iterations: 3
```

(e) [3] Some predictions are shown in Figure 5. Explain briefly how these predictions are consistent with your interpretations of each of the two Estimates in the model `spector.2`.

My answer:

Previously, we said that students with higher GPA would have a higher probability of showing improvement on their exams, compared to students with lower GPA, for a fixed value of `psi`. So, hold `psi` constant and change `gpa`. For example, compare rows 1, 2, and 3 of the predictions: the predicted probabilities do indeed go up as `gpa` goes up.

To see the effect of `psi` on predictions, hold `gpa` constant and let `psi` change (eg compare rows 1 and 4). The predicted probability is (substantially) larger when `psi` is `yes` compared to `no`. This is completely consistent with the Estimate of `psi` being positive.

Telling me which rows of the Figure you are comparing is a quick way for me to check that you are comparing sensible things.

(f) [3] Comment briefly on the width of the confidence intervals in Figure 5. Given what you know about the data, does your comment make sense? Explain briefly.

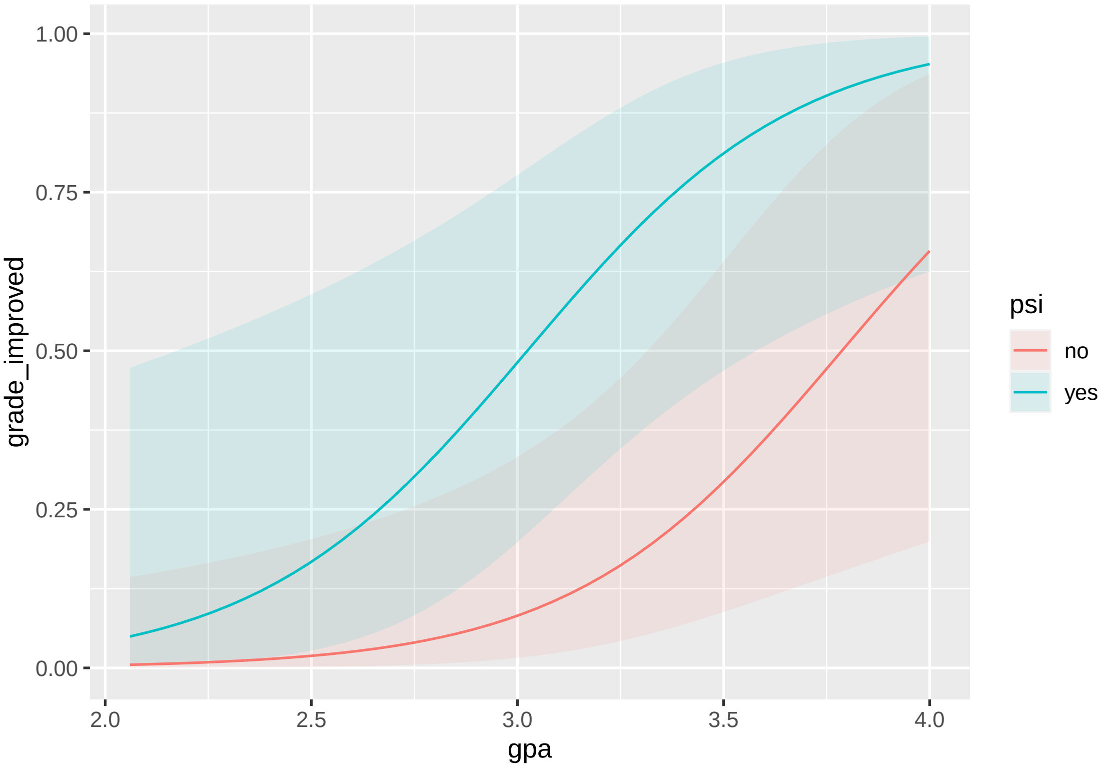
My answer:

The confidence intervals are (very) wide: they go up and down a long way from the prediction in each case.

The usual reason for a wide confidence interval is a small sample size. Here there are 32 observations, as you see in Figure 2. If we were estimating a mean, this would not be especially small (we could in that case expect a good bit of help from the Central Limit Theorem), but for these data, each observation only gives us a `yes` or a `no` for the response, so the 32 observations don't actually contain much information, and we would need a much larger sample size than we had to get those intervals to be reasonably narrow.

Extra: the graph made by `plot_cap` offers a little insight:

```r
plot_cap(spector.2, condition = c("gpa", "psi"))
```



Once again, this graph shows that grades are more likely to be improved if GPA is larger, and students exposed to PSI are more likely to have improved grades than students not so exposed, at any GPA. This graph comes with "envelopes" that express the accuracy of the predictions (as `geom_smooth` does): a blue envelope about the predictions for `psi yes` and a red one about the predictions for `psi no`. These envelopes overlap all the way up (the grey piece in the middle is part of both envelopes). Thus the significance of these effects is not completely convincing, which shows up in their P-values being small but not *very* small.

2. Boxes each containing approximately 100 trout eggs were buried at five different locations in a river, and retrieved (removed from the river) at four different times. (There were thus $5 \times 4 = 20$ boxes of eggs altogether, and each box was retrieved only once.) After each box was retrieved, the number of surviving eggs was recorded. The data are shown in Figure 6. The columns are, respectively, the number of eggs in the box that survived, the total number of eggs in the box to begin with, the location in the river (coded A through E), and the number of days the box was left in the river.

   (a) [3] Some code is shown in Figure 7. Why is it necessary to run this code? Explain briefly.

My answer:

Our data has more than one individual (egg) per row, along with counts of how many eggs survived. This means that we need to create the response matrix shown in order to fit a logistic regression. (This is different from having each *egg* on one line of the data file, along with a record of whether that particular egg survived or not. Make it clear that you know the difference.)

One point for saying that we needed to work out how many eggs did not survive, without making any further relevant points. Two points if you also say that we need a two-column response without making it clear enough *why* that is.

(This is an analysis of survival, but curiously it is not a “survival analysis”. To be a survival analysis, we would need to record the exact time that each egg was living before it died, and we don’t have that, or at least not accurately, because the boxes were removed from the river only at certain times rather than being monitored constantly.)

(b) [2] A logistic regression is shown in Figure 8. Does this logistic regression predict the probability that an egg survives, or the probability that the egg does not survive? How do you know? Explain briefly.

My answer:

When each row refers to multiple individuals, the probability being predicted is the thing in the *first* column of the response matrix. In Figure 7, the first column of `response` is `survive`, so we are predicting the probability that an egg *survives* (as it depends on location and the time it was left there).

You could also rationalize it by saying that in Figure 8, the slope of `period` is negative, and the longer you look, the more eggs you would expect to have died by that point, so the probability of *survival* must be the thing that’s going down. (This is not quite right, because there is also a period-squared term, but the relationship is only a little curved, so the logic is actually sound.) You can also do this by looking at the locations in the data and seeing that location A has the best survival and E the worst, and, given the signs of the Estimates, it must be survival that’s being predicted. Doing this gives you some help for the next part.

One point if you say that `dead` is the baseline and it’s therefore probability of survival. That would be true if there were *one* egg per row of the dataframe, rather than a whole box of eggs.

No reason is no points.

(c) [2] From Figure 8, at which location is the probability of an egg surviving predicted to be highest, all else equal? (That is, you may assume that the comparison of locations is done for the same value of `period`.) Explain briefly.

My answer:

There are five locations, A through E, with location being (evidently) categorical. Location A is the baseline, so its Estimate is zero. The Estimates for the other `location`s are all negative, so the highest predicted probability of survival, all else equal, is at location A.

You can also note that there is not a significant difference in probability between location A and location B, so a reasonable answer *if you make this point* is also “either location A or B”. Apart from this consideration, this question has nothing to do with P-values.

If you thought earlier that we were predicting the probability that an egg *did not* survive, the right answer here is “location E”, because it has the most *negative* coefficient. (If that was your answer earlier, an answer of “location A” here is *wrong* because you are being inconsistent.)

I need a reason. An answer without one is zero.

3. In 1993, a survey was carried out in western Germany about attitudes towards science. We focus on one of the survey questions, here labelled `D`. The responses to question `D` are labelled 1, “strongly disagree”, through 5, “strongly agree”. A person who answered 5 to this question has a strong (positive) belief in the value of science. For each survey respondent, the `sex` they identified as, and their level `edu` of education, were also recorded. The column `edu` was actually recorded in categories, but we will treat this as quantitative, with a higher value of `edu` corresponding with more education. Some of the data is shown in Figure 9.

(a) [3] A model was fitted, as shown in Figure 10. Even though the responses to item D are given as numbers, I used `polr` instead of an ordinary linear regression. Why was that? Explain briefly.

My answer:

There are two things to say here:

- why not a linear regression?
- why `polr` as opposed to something else?

The numerical values in column `D` are actually labels for the five categories from “strongly disagree” through “strongly agree”, and don’t have any numerical meaning here at all. (1 point.) `polr` is used for models where the response is categorical with more than two categories (1 point) and those categories are *in order* (the last point).

I wanted you to show me that the numbers correspond to categories from strongly disagree to strongly agree, and this is where the ordering comes from. Otherwise, the numbers 1 to 5 could be labels for something that actually does *not* have an order (like favourite colours in boxes of Smarties, or M&Ms if you prefer those).

A hint is the use of `factor(D)` in Figure 10 to turn `D` into something categorical.

(b) [2] Some predictions are shown in Figure 11. Why was the `pivot_wider` a good idea? Explain briefly.

My answer:

`predictions` will put *one* predicted probability on each row of the output dataframe: that is to say, there will be one row for each `sex`, `edu` *and* `D` combination, for a total of $2 \times 2 \times 5 = 20$ rows. This will make it difficult to assess the effect of `sex` and `edu` on a person's attitudes towards science. The `pivot_wider` will display all five probabilities for the same values of `sex` and `edu` on *one* line, which will make the assessment of effects much easier.

The people that lost out here did so by not being clear enough. The issue here is that it is the `pivot_wider` that puts all five of the predictions (for the probabilities of the responses to `D`) on the same line. "Tidying the data" is not specific enough.

(c) [3] Is it true based on Figure 11 that someone with more education has a stronger belief in the value of science overall, all else equal? Explain briefly.

My answer:

In the table at the bottom of Figure 11, compare two rows that differ in `edu` but have the same `sex`. This could be rows 1 and 2, or you could compare rows 3 and 4.

For a person with more education (a higher value of `edu`), the probability of responding 1 or 2 or 3 is less, and the probability of responding 4 or 5 is substantially more. That is to say, a person with more education is more likely to have a stronger belief in the value of science, because they are more likely to respond at the upper end of the scale and less likely to respond at the lower end.

For this kind of question, a comparison of the probabilities of each response category is a reasonable starting point, but you need to go beyond that to say whether there is an overall tendency towards the upper or lower end, and (in this case) what that means in terms of attitudes towards science. I would also like to see a comparison of *two* or more categories, preferably one at the low end going down and one at the upper end going up as education increases, to show that it's not just the very highest category that happens to increase, but an overall trend of a greater belief in science.

No explanation might mean one point, if you are lucky.

(d) [3] In Figure 11, would you describe the effect of `sex` as large or small? Explain briefly. From the output in Figure 10, why would you have expected to see the size of effect you did? Explain briefly again.

My answer:

To see the effect of `sex`, compare two rows where `sex` differs but `edu` (education) is the same, such as the first and third rows. (Tell me which rows you're comparing.) The differences between the five predictions in those two rows are all small, with (maybe) the males having a slightly stronger belief than females do (for the same level of education). But the overall effect of `sex` is small. Something of that sort for two points.

In Figure 10, `sex` is actually not significant at $\alpha = 0.05$ (its P-value is 0.063) and so could have been removed from the model. So it makes sense that its effect is small. Kind of a giveaway one point.

4. An important part of any business is making sure that the business gets paid for the work it does. Most businesses issue an invoice stating how much money the customer owes, and giving the customer a certain amount of time to pay the invoice. When the customer pays an invoice, the business marks the invoice as "settled", and no further action is needed. (If the customer does not pay an invoice by the due date, further action will need to be taken to make sure the invoice gets paid.)

A certain company kept track of over two thousand invoices it issued between January 2012 and November 2014. The information we will use is:

- `invoice_date`: when the invoice was issued
- `due_date`: when payment is due
- `invoice_amount`: how much money ($) the invoice is for
- `settled_date`: when the invoice was settled (paid)

The dataset also contains information about the customer who received each invoice, which we will not use in this question. Some of the (relevant) data, in dataframe `receivables`, is shown in Figure 12. You will be writing some code in this question.

(a) [3] All the dates in the data file were recorded as text. What code would re-define the `invoice_date` column to be an R date? (For the rest of the question, assume that the other two dates have also been re-defined as R dates.)

My answer:

Take a look at the form of the dates in Figure 12. It must be month/day/year, separated by slashes, because the first value (in the data you see) is always 12 or less, and the second value can be greater than 12 but is always 31 or less. Hence:

```
receivables %>%
  mutate(invoice_date = mdy(invoice_date))
```

```
## # A tibble: 2,466 x 4
##    invoice_date due_date   invoice_amount settled_date
##    <date>       <chr>               <dbl> <chr>
##  1 2013-01-02   2/1/2013             55.9 1/15/2013
##  2 2013-01-26   2/25/2013            61.7 3/3/2013
##  3 2013-07-03   8/2/2013             65.9 7/8/2013
##  4 2013-02-10   3/12/2013           106.  3/17/2013
##  5 2012-10-25   11/24/2012           72.3 11/28/2012
##  6 2012-01-27   2/26/2012            94   2/22/2012
##  7 2013-08-13   9/12/2013            74.7 9/9/2013
##  8 2012-12-16   1/15/2013            75.1 1/12/2013
##  9 2012-05-14   6/13/2012            80.1 7/1/2012
## 10 2013-07-01   7/31/2013            48.3 7/26/2013
## # i 2,456 more rows
```

Mainly I was checking that you had `mdy(invoice_date)` in there somewhere.

`as.Date` does *not* work, because that only applies to dates in the form 2023-03-13 with the year first and the month second. This is an excellent format for you to keep *your* dates in, but other people are usually not so obliging (as here).

Extra: here's the dataframe that will be used for the rest of the question:

```
receivables %>%
  mutate(invoice_date = mdy(invoice_date),
         due_date = mdy(due_date),
         settled_date = mdy(settled_date)) -> receivables
```

(b) [2] Someone tells you that this business always gives all its customers the same amount of time to pay their invoices. What code would tell you how many days that was?

My answer:

The obvious choice is to work out the fractional number of days between the invoice date and the due date:

```
receivables %>%
  mutate(until_due = (as.period(invoice_date %--% due_date)/ days(1)))
```

```
## # A tibble: 2,466 x 5
##    invoice_date due_date   invoice_amount settled_date until_due
##    <date>       <date>              <dbl> <date>           <dbl>
##  1 2013-01-02   2013-02-01           55.9 2013-01-15        30
##  2 2013-01-26   2013-02-25           61.7 2013-03-03        30
##  3 2013-07-03   2013-08-02           65.9 2013-07-08        30
##  4 2013-02-10   2013-03-12          106.  2013-03-17        32.4
##  5 2012-10-25   2012-11-24           72.3 2012-11-28        30
##  6 2012-01-27   2012-02-26           94   2012-02-22        30
##  7 2013-08-13   2013-09-12           74.7 2013-09-09        30
##  8 2012-12-16   2013-01-15            75.1 2013-01-12        30
##  9 2012-05-14   2012-06-13            80.1 2012-07-01        30
## 10 2013-07-01   2013-07-31            48.3 2013-07-26        30
## # i 2,456 more rows
```

See below for more about that 32.4 days.

Another approach is to reason that dates are stored as a number of days internally, so simply subtracting the dates ought to work, and so it does. This actually comes out the best:

```
receivables %>%
  mutate(until_due = due_date - invoice_date)
```

```
## # A tibble: 2,466 x 5
##    invoice_date due_date   invoice_amount settled_date until_due
##    <date>       <date>              <dbl> <date>       <drtn>
##  1 2013-01-02   2013-02-01           55.9 2013-01-15   30 days
##  2 2013-01-26   2013-02-25           61.7 2013-03-03   30 days
##  3 2013-07-03   2013-08-02           65.9 2013-07-08   30 days
##  4 2013-02-10   2013-03-12          106.  2013-03-17   30 days
##  5 2012-10-25   2012-11-24           72.3 2012-11-28   30 days
##  6 2012-01-27   2012-02-26           94   2012-02-22   30 days
##  7 2013-08-13   2013-09-12           74.7 2013-09-09   30 days
##  8 2012-12-16   2013-01-15           75.1 2013-01-12   30 days
##  9 2012-05-14   2012-06-13           80.1 2012-07-01   30 days
## 10 2013-07-01   2013-07-31           48.3 2013-07-26   30 days
## # i 2,456 more rows
```

Or you could even compute the `period` without converting it into decimal days:

```
receivables %>%
  mutate(until_due = as.period(invoice_date %--% due_date))
```

```
## # A tibble: 2,466 x 5
##    invoice_date due_date   invoice_amount settled_date until_due
##    <date>       <date>              <dbl> <date>       <Period>
##  1 2013-01-02   2013-02-01           55.9 2013-01-15   30d 0H 0M 0S
##  2 2013-01-26   2013-02-25           61.7 2013-03-03   30d 0H 0M 0S
##  3 2013-07-03   2013-08-02           65.9 2013-07-08   30d 0H 0M 0S
##  4 2013-02-10   2013-03-12          106.  2013-03-17   1m 2d 0H 0M 0S
##  5 2012-10-25   2012-11-24           72.3 2012-11-28   30d 0H 0M 0S
##  6 2012-01-27   2012-02-26           94   2012-02-22   30d 0H 0M 0S
##  7 2013-08-13   2013-09-12           74.7 2013-09-09   30d 0H 0M 0S
##  8 2012-12-16   2013-01-15           75.1 2013-01-12   30d 0H 0M 0S
##  9 2012-05-14   2012-06-13           80.1 2012-07-01   30d 0H 0M 0S
## 10 2013-07-01   2013-07-31           48.3 2013-07-26   30d 0H 0M 0S
## # i 2,456 more rows
```

If you went this way, it works here, so you are good, but technically it fails in general for the same reason that `day(until_due)` fails below, though it "fails better" because it shows you the bigger time unit (months) and you can work out how many days it was.

These last two work because days is an obvious choice for a “reasonable” timescale here. (The fourth invoice is “1 month 2 days” because the month in question is a February with 28 days, which shows up in the last one and the fractional days, because it is reckoned to be over a month.)

Using `settled_date` is not helpful: that would tell you how long until the invoice was *actually* paid, not how long until it was *supposed* to be paid.

This variation doesn't quite get there:

```
receivables %>%
  mutate(until_due = (as.period(invoice_date %--% due_date))) %>%
  mutate(days_of = day(until_due))
```

```
## # A tibble: 2,466 x 6
##    invoice_date due_date   invoice_amount settled_date until_due      days_of
##    <date>       <date>              <dbl> <date>       <Period>         <dbl>
##  1 2013-01-02   2013-02-01           55.9 2013-01-15   30d 0H 0M 0S        30
##  2 2013-01-26   2013-02-25           61.7 2013-03-03   30d 0H 0M 0S        30
##  3 2013-07-03   2013-08-02           65.9 2013-07-08   30d 0H 0M 0S        30
##  4 2013-02-10   2013-03-12          106.  2013-03-17   1m 2d 0H 0M 0S       2
##  5 2012-10-25   2012-11-24           72.3 2012-11-28   30d 0H 0M 0S        30
##  6 2012-01-27   2012-02-26           94   2012-02-22   30d 0H 0M 0S        30
##  7 2013-08-13   2013-09-12           74.7 2013-09-09   30d 0H 0M 0S        30
##  8 2012-12-16   2013-01-15           75.1 2013-01-12   30d 0H 0M 0S        30
##  9 2012-05-14   2012-06-13           80.1 2012-07-01   30d 0H 0M 0S        30
## 10 2013-07-01   2013-07-31           48.3 2013-07-26   30d 0H 0M 0S        30
## # i 2,456 more rows
```

If `days_of` is less than a month, this works, but if it's more than a month (see the fourth line) it's the number of *left-over* days beyond however many months there are. I think this was reasonable for you to guess from what we learned; the number of days might have been something like 90, and that would definitely have shown up here as some number of months, and once again this code would have given the number of days left over, beyond however-many months it was. Not a huge deal, but a deal nevertheless.

Extra: it turns out that the payment time is 30 days. Note that the due date is often, but not always, one day earlier in the month than the invoice date, because most (but not all) months have 31 days. This is the kind of calculation that we do better to let `lubridate` handle, because it is better for that package to think and get it right than for us to think and (quite likely) get it wrong.

(c) [2] We are asked to find out how many of the invoices were paid on or before the due date, and how many were paid after the due date. What code would find this out?

My answer:

The idea is one you may have seen from the power simulation stuff in C32: count a logical condition:

```
receivables %>%
  count(settled_date <= due_date)
```

```
## # A tibble: 2 x 2
##   `settled_date <= due_date`     n
##   <lgl>                      <int>
## 1 FALSE                        877
## 2 TRUE                        1589
```

The count by TRUE is how many were paid on time, and the count by FALSE is the number that were paid late. You can compare dates directly (because they are numbers under the hood).

If you prefer, define a new column that says whether each invoice was paid on time or not, as done below, and count that. I like some extra brackets on a logical condition inside a `mutate` so as make clear what is done first (evaluate the logical condition, and *then* assign its result to `paid_on_time`):

```
receivables %>%
  mutate(paid_on_time = (settled_date <= due_date)) %>%
  count(paid_on_time)
```

```
## # A tibble: 2 x 2
##   paid_on_time     n
##   <lgl>        <int>
## 1 FALSE          877
## 2 TRUE          1589
```

The answer is the same both ways: not quite two-thirds of the invoices were paid on time.

A lot of people made this a lot more difficult than it needed to be. A common approach was to subtract (in effect) the settled date and the due date. This doesn't do much by itself (half a point), but if you do something to distinguish the ones of those that are positive from the ones that are negative, you are getting somewhere (one point). The second point is then for counting them, via `count` or via `group_by` (the categorical variable that indicates whether the time difference is positive or negative) and `summarize` with `n()`, as for example like this:

```
receivables %>%
  mutate(days_late = as.period(due_date %--% settled_date)) %>%
  mutate(was_late = (days_late > 0)) %>%
  group_by(was_late) %>%
  summarize(count = n())
```

```
## # A tibble: 2 x 2
##   was_late count
##   <lgl>    <int>
## 1 FALSE     1589
## 2 TRUE       877
```

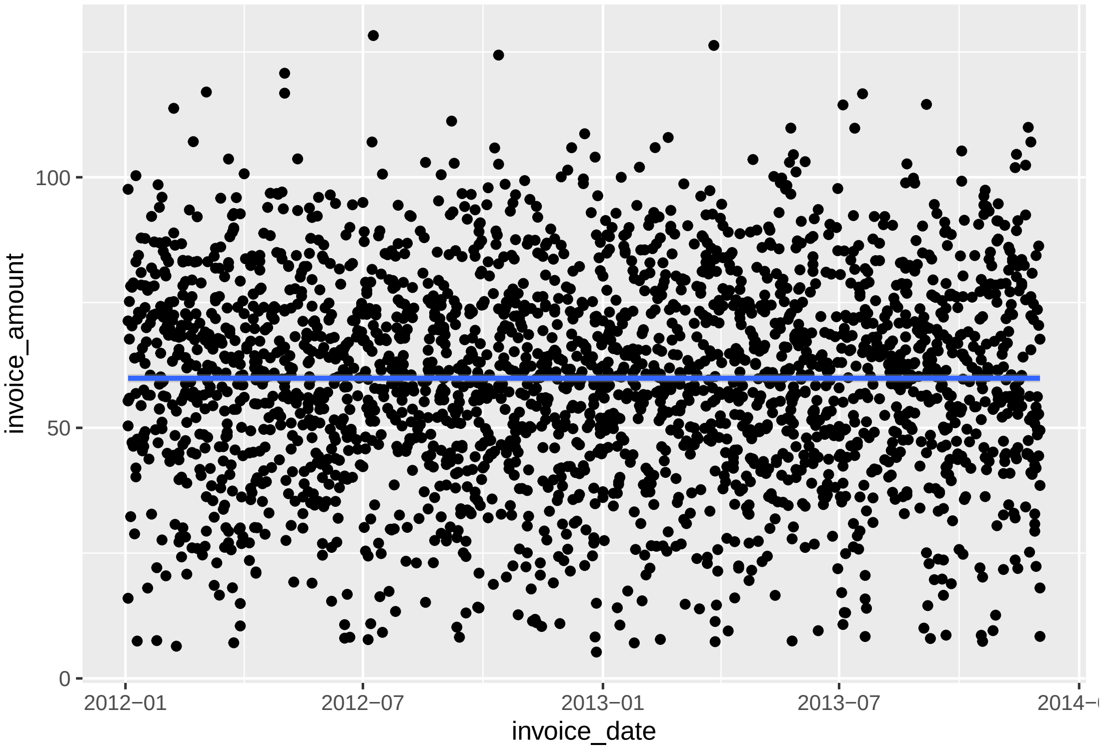
(d) [3] The business management wants to know whether invoice amounts are changing over time. Bear in mind that there are over two thousand invoices of varying sizes, so the management may need some help in seeing any trend. What code will draw a suitable graph to help the management find out what they want to know? Use whichever dates you feel are appropriate.

My answer:

Really the only date-related thing in this part is knowing that dates will plot properly on a `ggplot`. The rest of it is to plot the invoice amounts against one of the dates, and to add a smooth trend to help the management see whether or not it is changing over time. My take is that the invoice date is the best, because that was when the service (whatever it was) was provided, but I have no objection to your using either of the other dates:

```
ggplot(receivables, aes(x = invoice_date, y = invoice_amount)) +
  geom_point() + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

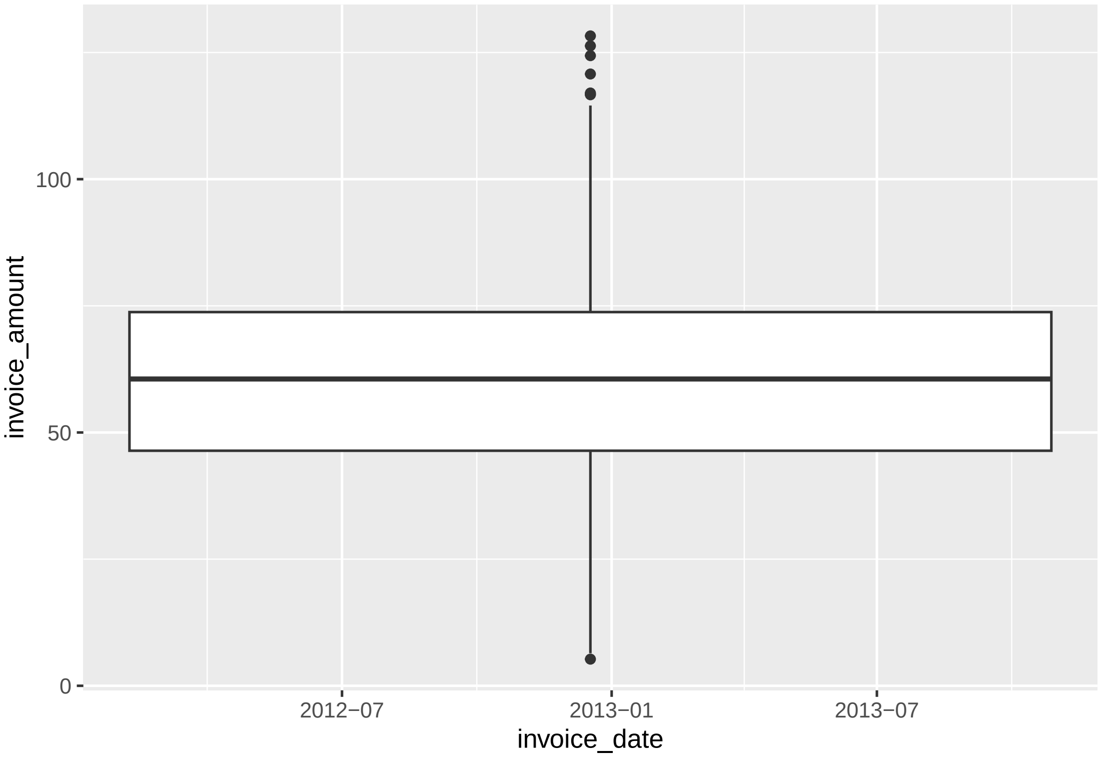


Extra: it turns out that there is no trend at all: the average invoice is about $60 the whole time. The pattern of points makes it very difficult to see whether there is a trend at all, so you need to add a smooth trend. A smooth trend is better than a regression line, because the trend might be something up-and-down or seasonal, which a straight line would miss.

There were quite a few different ideas here, and I think I was able to work out what everyone who made a serious effort was trying to do. Some people wanted histograms of invoice amounts by date; there are a lot of different dates, so you would have a lot of different histograms and it would be hard to see any trend in the invoice amounts. I have more sympathy for trying to do it with boxplots, but you have to discretize time somehow, or else it won't work at all:

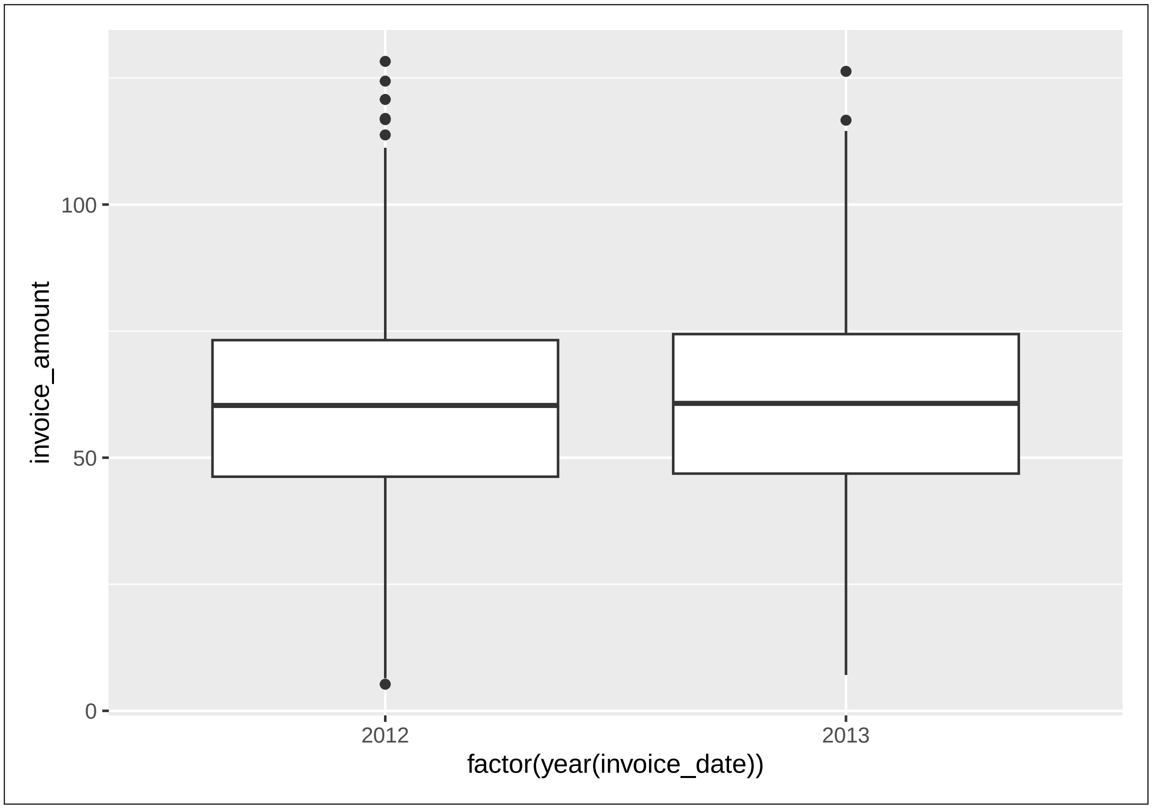
```
ggplot(receivables, aes(x = invoice_date, y = invoice_amount)) +
  geom_boxplot()
```

```
## Warning: Continuous x aesthetic
## i did you forget `aes(group = ...)`?
```



This one is too crude of a discretization, but is at least showing something:

```
ggplot(receivables, aes(x = factor(year(invoice_date)), y = invoice_amount)) +
  geom_boxplot()
```



(e) [4] The management also wants to know what the mean invoice amount is for each month over the time that the data were collected (that is, treating February of 2012 separately from February of 2013). What code would calculate these means?

My answer:

There is also a lot of overlap with C32 here. We will need to group by both year and month (to treat each month separately), which means we first need to extract the year and month from the dates before doing the group-by and summarize (which is the dates connection):

```
receivables %>%
  mutate(year = year(invoice_date), month = month(invoice_date)) %>%
  group_by(year, month) %>%
  summarize(month_mean = mean(invoice_amount))
```

```
## `summarise()` has grouped output by 'year'. You can override using the `.groups` argument.
```

```
## # A tibble: 24 x 3
## # Groups:   year [2]
##     year month month_mean
##    <dbl> <dbl>      <dbl>
##  1  2012     1       62.9
##  2  2012     2       61.1
##  3  2012     3       57.5
##  4  2012     4       61.9
##  5  2012     5       61.1
##  6  2012     6       56.9
##  7  2012     7       60.3
##  8  2012     8       59.9
##  9  2012     9       57.3
## 10  2012    10       61.3
## # i 14 more rows
```

I think it makes sense to use the `invoice_date` for this one, but I have once again no objection if you use one of the other dates.

For me, it is clearer to extract the year and month first, but the below also works, so is also good:

```
receivables %>%
  group_by(year(invoice_date), month(invoice_date)) %>%
  summarize(month_mean = mean(invoice_amount))
```

```
## `summarise()` has grouped output by 'year(invoice_date)'. You can override using the `.grou
```

```
## # A tibble: 24 x 3
## # Groups:   year(invoice_date) [2]
##    `year(invoice_date)` `month(invoice_date)` month_mean
##                   <dbl>                  <dbl>      <dbl>
##  1                 2012                      1       62.9
##  2                 2012                      2       61.1
##  3                 2012                      3       57.5
##  4                 2012                      4       61.9
##  5                 2012                      5       61.1
##  6                 2012                      6       56.9
##  7                 2012                      7       60.3
##  8                 2012                      8       59.9
##  9                 2012                      9       57.3
## 10                 2012                     10       61.3
## # i 14 more rows
```

Points: 2 for getting the year and month, one each for the group-by and the summarize.

Extra: `https://quickbooks.intuit.com/ca/resources/accounting/what-is-accounts-receivable/` is a very readable introduction to the world of accounts receivable and why this is something that anyone running a business needs to know about.

5. Twenty-six psychiatric inpatients, admitted to the University of Iowa hospitals during the years 1935-1948, were observed over a period of several years.

   Data for each patient consists of:

   - `age` at first admission to the hospital
   - `sex` (1 = male, 2 = female)
   - `time`: number of years of follow-up (years from admission to death or censoring)
   - `death`: patient status at the follow-up time (1 = dead, 0 = alive when last observed).

   Our aim is to understand how age and sex influence survival time among psychiatric patients. The dataset is shown in Figure 13.

   (a) [3] Some code is shown in Figure 14. What is `Surv` doing, what are its inputs, and why are some of the values in the last column displayed with a plus sign? Explain briefly.

   My answer:

   This is creating a response variable for a survival model (Cox proportional-hazards model), using the information in the survival `time`s and whether or not the patient was observed to die (in `death`).

   The inputs to `Surv` are the survival times, and something that will be `TRUE` if the event (death) happens; 1 means "dead" for these data.

   The values with plus signs in the new column are patients that were never observed to die (for whom the event "death" never happened; these are censored observations). Feel free to use the word "censored", but make sure your answer makes it clear that you know what censoring means for these data.

   There were some really clear answers, and a lot of people showed that they understood what was going on. The answers I thought were really clear were tagged "nice", which I hope you get to see if that is you.

   Extra: you might remember, at the beginning of the Feb 15 class, I mentioned that sometimes things are labelled as 1 and 2 because on the old punched cards, 0 was indistinguishable from "missing", and there was a kind of historical inertia that meant that things are *still* done that way today. I was amused to see that this seems to have happened here as well, with `sex`. I guess `death` being zero *is* itself a sort of missing, because the event of death was indeed missing for those patients.

   (b) [2] A Cox proportional-hazards model and its output is shown in Figure 15. Describe the effect of `age` as far you can deduce it from this Figure.

   My answer:

   The coefficient of `age` is positive. This means that, all else equal, an older patient has a greater hazard of death: that is to say, they are more likely to die sooner than a younger patient. (If you use the expression "hazard of death", you need to show me that you understand what it means.)

"All else equal" here means comparing patients of the same `sex`, although in actual fact `sex` is not significant, so that for these patients (or the population of patients of which these are a sample), there is no difference in survival times between males and females. The best answer says "all else equal" and then qualifies it by saying that there is no significant "else" to worry about.

The "sooner" in your answer is important. Everyone dies eventually, but the interest in a survival analysis is *how long* a person is likely to live (or, in general, how long they are likely to survive before the event of interest happens to them). So you need to say that an older person is more likely to die sooner, or is less likely to survive for a long time. The time aspect has to be in there somewhere.

(c) [3] A dataframe of values to predict for is set up in Figure 16, and Figure 17 shows estimated survival curves for these values. Describe how this plot supports your conclusions about the effects of `age` in the previous part (or does not support them, if that's the case).

My answer:

Compare the survival curves for the three ages for one of the sexes. For example, for females, the appropriate survival curves to compare are strata 2, 4, and 6 (yellow, light blue, and pink on the graph). As age increases, the survival curves go down the page, indicating that older patients are more likely to die sooner, as we concluded in the previous part. (Or, the younger patients are further up and to the right, so the younger patients are more likely to survive for longer.)

A complete answer includes an explanation of which strata you are comparing, their numbers or colours on the graph, and what their relative positions mean.

As before, "more likely to survive" is not a complete description of what is happening here, because everybody dies eventually; the interest in a survival model lies in *how long* a patient might survive, given their age and sex (and in other cases, treatment, or other variables).

(d) [2] Figure 15 shows that there is no significant effect of `sex`. How does this show up on the graph in Figure 17? Explain briefly.

My answer:

Pick an age, say 28. The two strata for age 28 are 1 and 2. These are red and yellow on the graph. These two survival curves are close together, indicating that any sex effect is small.

Or, do this and say that the sex effect is small compared to the (significant) age effect that you assessed in the previous part.

Be clear about what you are comparing. An answer of "the survival curves are close together" is only one point unless you also say which survival curves you are comparing and why.

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

Figures

```
library(tidyverse)
library(marginaleffects)
library(lubridate)
library(MASS)
library(survival)
library(survminer)
library(conflicted)
conflict_prefer("select", "dplyr")
```

```
## [conflicted] Removing existing preference.
## [conflicted] Will prefer dplyr::select over any other package.
```

Figure 1: Packages

```
my_url <- "http://ritsokiguess.site/datafiles/spector.csv"
spector <- read_csv(my_url)
```

```
## Rows: 32 Columns: 4
## -- Column specification -----------------------------------------------------------------------------
## Delimiter: ","
## chr (2): grade_improved, psi
## dbl (2): tuce, gpa
##
## 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.
```

```
spector
```

```
## # A tibble: 32 x 4
##    grade_improved psi    tuce   gpa
##    <chr>          <chr> <dbl> <dbl>
##  1 no             no       20  2.66
##  2 no             no       22  2.89
##  3 no             no       24  3.28
##  4 no             no       12  2.92
##  5 yes            no       21  4
##  6 no             no       17  2.86
##  7 no             no       17  2.76
##  8 no             no       21  2.87
##  9 no             no       25  3.03
## 10 yes            no       29  3.92
## # i 22 more rows
```

Figure 2: Teaching method study data (some)

```
spector.1 <- glm(factor(grade_improved) ~ psi + tuce + gpa, data = spector, family = "binomial")
summary(spector.1)
```

```
##
## Call:
## glm(formula = factor(grade_improved) ~ psi + tuce + gpa, family = "binomial",
##     data = spector)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -13.02135    4.93127  -2.641  0.00828 **
## psiyes        2.37869    1.06456   2.234  0.02545 *
## tuce          0.09516    0.14155   0.672  0.50143
## gpa           2.82611    1.26293   2.238  0.02524 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 41.183  on 31  degrees of freedom
## Residual deviance: 25.779  on 28  degrees of freedom
## AIC: 33.779
##
## Number of Fisher Scoring iterations: 5
```

Figure 3: Teaching method study analysis part 1

```
spector.2 <- update(spector.1, .~. - tuce)
summary(spector.2)
```

```
##
## Call:
## glm(formula = factor(grade_improved) ~ psi + gpa, family = "binomial",
##     data = spector)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -11.602      4.213  -2.754  0.00589 **
## psiyes         2.338      1.041   2.246  0.02470 *
## gpa            3.063      1.223   2.505  0.01224 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 41.183  on 31  degrees of freedom
## Residual deviance: 26.253  on 29  degrees of freedom
## AIC: 32.253
##
## Number of Fisher Scoring iterations: 5
```

Figure 4: Teaching method study analysis part 2

```
psis <- c("no", "yes")
gpas <- c(2.8, 3.1, 3.5)
new <- datagrid(psi = psis, gpa = gpas, model = spector.2)
new %>% select(psi, gpa)
```

```
##   psi gpa
## 1  no 2.8
## 2  no 3.1
## 3  no 3.5
## 4 yes 2.8
## 5 yes 3.1
## 6 yes 3.5
```

```
cbind(predictions(spector.2, newdata = new)) %>%
  select(estimate, conf.low, conf.high, psi, gpa)
```

```
##    estimate    conf.low conf.high psi gpa
## 1 0.0463473 0.006413699 0.2678833  no 2.8
## 2 0.1085996 0.024086475 0.3755384  no 3.1
## 3 0.2932235 0.088128350 0.6404104  no 3.5
## 4 0.3348431 0.100939593 0.6929838 yes 2.8
## 5 0.5579014 0.257191196 0.8214082 yes 3.1
## 6 0.8112260 0.468529084 0.9544377 yes 3.5
```

Figure 5: Teaching method study predictions

```
troutegg
```

```
## # A tibble: 20 x 4
##    survive total location period
##      <dbl> <dbl> <chr>     <dbl>
##  1      89    94 A             4
##  2     106   108 B             4
##  3     119   123 C             4
##  4     104   104 D             4
##  5      49    93 E             4
##  6      94    98 A             7
##  7      91   106 B             7
##  8     100   130 C             7
##  9      80    97 D             7
## 10      11   113 E             7
## 11      77    86 A             8
## 12      87    96 B             8
## 13      88   119 C             8
## 14      67    99 D             8
## 15      18    88 E             8
## 16     141   155 A            11
## 17     104   122 B            11
## 18      91   125 C            11
## 19     111   132 D            11
## 20       0   138 E            11
```

Figure 6: Trout egg data

```
troutegg %>%
  mutate(dead = total - survive) %>%
  select(survive, dead) %>%
  as.matrix() -> response
response
```

```
##       survive dead
##  [1,]      89    5
##  [2,]     106    2
##  [3,]     119    4
##  [4,]     104    0
##  [5,]      49   44
##  [6,]      94    4
##  [7,]      91   15
##  [8,]     100   30
##  [9,]      80   17
## [10,]      11  102
## [11,]      77    9
## [12,]      87    9
## [13,]      88   31
## [14,]      67   32
## [15,]      18   70
## [16,]     141   14
## [17,]     104   18
## [18,]      91   34
## [19,]     111   21
## [20,]       0  138
```

Figure 7: Trout egg code

```
troutegg.1 <- glm(response ~ period + location + I(period^2),
                  family = "binomial", data = troutegg)
summary(troutegg.1)
```

```
##
## Call:
## glm(formula = response ~ period + location + I(period^2), family = "binomial",
##     data = troutegg)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  9.69222    0.79999  12.115  < 2e-16 ***
## period      -1.60962    0.19677  -8.180 2.83e-16 ***
## locationB   -0.41505    0.24618  -1.686   0.0918 .
## locationC   -1.24068    0.21952  -5.652 1.59e-08 ***
## locationD   -0.94708    0.22882  -4.139 3.49e-05 ***
## locationE   -4.60742    0.24955 -18.463  < 2e-16 ***
## I(period^2)  0.08436    0.01200   7.031 2.05e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1021.469  on 19  degrees of freedom
## Residual deviance:   65.687  on 13  degrees of freedom
## AIC: 156.22
##
## Number of Fisher Scoring iterations: 5
```

Figure 8: Trout egg logistic regression

```
wg93
```

```
## # A tibble: 871 x 3
##         D sex      edu
##     <dbl> <chr>  <dbl>
##  1      3 male       3
##  2      3 female     4
##  3      4 male       2
##  4      2 female     3
##  5      3 female     2
##  6      5 female     2
##  7      4 male       2
##  8      2 female     3
##  9      1 female     2
## 10      2 female     2
## # i 861 more rows
```

Figure 9: German science survey data

```
D.1 <- polr(factor(D) ~ sex + edu, data = wg93)
drop1(D.1, test = "Chisq")
```

```
## Single term deletions
##
## Model:
## factor(D) ~ sex + edu
##         Df    AIC    LRT Pr(>Chi)
## <none>    2664.9
## sex     1 2666.4 3.4622 0.062785 .
## edu     1 2671.7 8.8181 0.002983 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 10: German science survey model

```
new <- datagrid(sex = c("female", "male"), edu = c(1, 6), model = D.1)
cbind(predictions(D.1, newdata = new)) %>%
  select(group, estimate, sex, edu) %>%
  pivot_wider(names_from = "group", values_from = "estimate")
```

```
##
## Re-fitting to get Hessian
## # A tibble: 4 x 7
##   sex      edu    `1`   `2`   `3`   `4`   `5`
##   <chr>  <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 female     1 0.0969 0.328 0.235 0.218 0.123
## 2 female     6 0.0499 0.215 0.221 0.291 0.222
## 3 male       1 0.0788 0.292 0.237 0.244 0.149
## 4 male       6 0.0402 0.183 0.207 0.305 0.264
```

Figure 11: German science survey predictions

```
receivables
```

```
## # A tibble: 2,466 x 4
##    invoice_date due_date   invoice_amount settled_date
##    <chr>        <chr>               <dbl> <chr>
##  1 1/2/2013     2/1/2013             55.9 1/15/2013
##  2 1/26/2013    2/25/2013            61.7 3/3/2013
##  3 7/3/2013     8/2/2013             65.9 7/8/2013
##  4 2/10/2013    3/12/2013           106.  3/17/2013
##  5 10/25/2012   11/24/2012           72.3 11/28/2012
##  6 1/27/2012    2/26/2012            94   2/22/2012
##  7 8/13/2013    9/12/2013            74.7 9/9/2013
##  8 12/16/2012   1/15/2013            75.1 1/12/2013
##  9 5/14/2012    6/13/2012            80.1 7/1/2012
## 10 7/1/2013     7/31/2013            48.3 7/26/2013
## # i 2,456 more rows
```

Figure 12: Accounts receivable data

```
psych
```

```
##    sex age time death
## 1    2  51    1     1
## 2    2  58    1     1
## 3    2  55    2     1
## 4    2  28   22     1
## 5    1  21   30     0
## 6    1  19   28     1
## 7    2  25   32     1
## 8    2  48   11     1
## 9    2  47   14     1
## 10   2  25   36     0
## 11   2  31   31     0
## 12   1  24   33     0
## 13   1  25   33     0
## 14   2  30   37     0
## 15   2  33   35     0
## 16   1  36   25     1
## 17   1  30   31     0
## 18   1  41   22     1
## 19   2  43   26     1
## 20   2  45   24     1
## 21   2  35   35     0
## 22   1  29   34     0
## 23   1  35   30     0
## 24   1  32   35     1
## 25   2  36   40     1
## 26   1  32   39     0
```

Figure 13: Psychiatric patients data

```
psych %>%
  mutate(y = Surv(time, death == 1)) -> psych
head(psych)
```

```
##   sex age time death   y
## 1   2  51    1     1   1
## 2   2  58    1     1   1
## 3   2  55    2     1   2
## 4   2  28   22     1  22
## 5   1  21   30     0 30+
## 6   1  19   28     1  28
```

Figure 14: Psychiatric patients: some code and its output. Note that “head” displays the first six lines of its input.

```
psych.1 <- coxph(y ~ age + sex, data = psych)
summary(psych.1)
```

```
## Call:
## coxph(formula = y ~ age + sex, data = psych)
##
##   n= 26, number of events= 14
##
##         coef exp(coef) se(coef)      z Pr(>|z|)
## age  0.20753   1.23063  0.05828  3.561  0.00037 ***
## sex -0.52374   0.59230  0.73753 -0.710  0.47762
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##     exp(coef) exp(-coef) lower .95 upper .95
## age    1.2306     0.8126    1.0978     1.380
## sex    0.5923     1.6883    0.1396     2.514
##
## Concordance= 0.816  (se = 0.081 )
## Likelihood ratio test= 20.91  on 2 df,   p=3e-05
## Wald test            = 14.3  on 2 df,   p=8e-04
## Score (logrank) test = 21.27  on 2 df,   p=2e-05
```

Figure 15: Psychiatric patients: model and output

```
new <- datagrid(age = c(28, 35, 42), sex = c(1, 2), model = psych.1)
new
```

```
##   age sex
## 1  28   1
## 2  28   2
## 3  35   1
## 4  35   2
## 5  42   1
## 6  42   2
```

Figure 16: Psychiatric patients: values to predict

```
s <- survfit(psych.1, newdata = new, data = psych)
ggsurvplot(s, conf.int = FALSE)
```

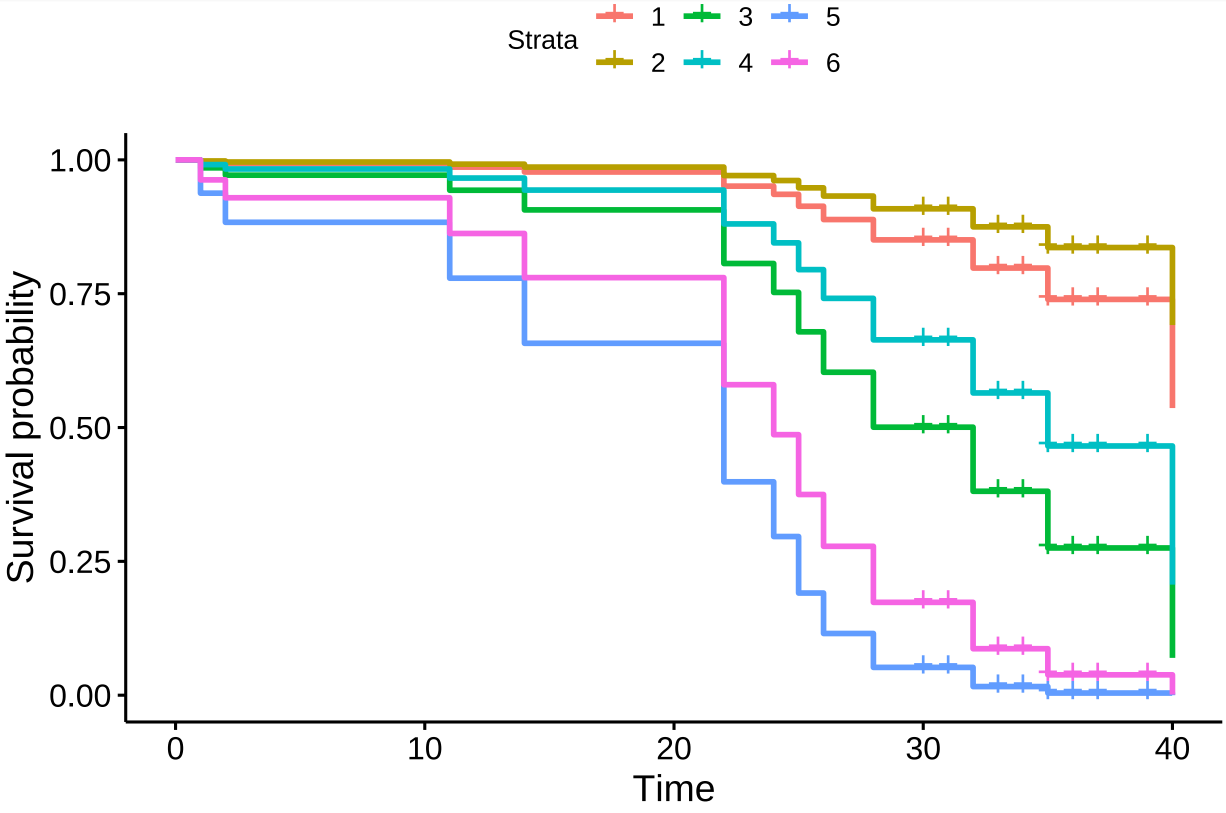


Figure 17: Psychiatric patients: predictions and plot of estimated survival curves