University of Toronto Scarborough
Department of Computer and Mathematical Sciences
STAD29 / STA 1007 (K. Butler), Final Exam
April 10, 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 55 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 (9 marks)

An experiment is run to test what effect the dose of a drug (measured in mg) has on how lethargic a rat is. This is measured by the amount of time a rat spends sleeping or resting in a four-hour period. It is also suspected that the age of a rat (measured in months) will have some impact on how much time the rat spends sleeping or resting. The data are shown in Figure 2. The column `dose` is a number, but is treated here as a factor.

(a) (2 marks) What feature or features of this data set mean that analysis of covariance is a suitable method to analyze it? Explain briefly.

My answer: There is a continuous response variable, but most importantly, a categorical explanatory variable (dose) and a quantitative one (age). One point for each of those.

I'm looking for something that says what *types* of explanatory variable we have.

(b) (3 marks) A scatterplot is shown in Figure 23. What does this plot suggest about effects of age, dose and interaction between them? Explain briefly. (Note: the graphs in colour are at the end of the Booklet of Code and Output. I apologize in advance for any flipping back and forth you need to do.)

My answer: A point for each of these:

- There is likely an effect of age because the lines (mostly) go uphill (increased age goes with increased rest time at all the doses except 0).
- There is likely an effect of treatment because the group of coloured points for each dose looks clearly above the group of points for the next dose.
- There is likely an interaction because the lines don't look parallel. I would say the lines get steeper as you go up the page, and the points are close enough to the lines that these differences in slope appear real (rather than just chance).

  You can disagree with any of this if you want, and if you are persuasive enough I'm ok with it. For example, you might reckon that the lines are "approximately parallel", in which case you would say that there is no interaction. You're about to get a surprise in that regard, but it's a reasonable (or perhaps I should say "not unreasonable") inference from the graph.

Try to make a comment about each of dose effect, age effect and interaction, and how you know in each case. (The reasons as you see are different in each case.)

(c) (2 marks) An analysis of covariance for these data is shown in Figure 3. What do you conclude from it, in the context of the data?

My answer: The interaction between dose and age is strongly significant. This means that the effect of dose on rest time is different for each age. Or, flip it around and say that the effect of age on rest time is different for each dose.

That is to say, the interaction effect that you (probably) expected to see is indeed there.

This is, as you recall, where you stop. I was going to deduct a point if you kept on going, until I realized that I misled you by asking for all three things on the graph. It would have been better to ask you only about the interaction on the graph, and then I would have been justified in doing that, but I couldn't really penalize you for drawing a conclusion that was entirely consistent with what you said about the graph (if that's what you said).

Having said that, for two points here I think you need to say something about what a significant interaction *means* (even if it's as simple as "rest time depends on the combination of age and dose", or something more precise like "the effect of age is different for each dose", or something inferred from the graph like "age has a larger influence on rest time at higher doses"), so if you ended up with one, I didn't feel you did that. I didn't penalize you for talking about the main effects, if you did, for reasons I explained above.

(d) (2 marks) In class, we didn't talk about simple effects in this kind of model, but describe briefly how you might use a simple-effects idea to understand your conclusion for this data set.

My answer: The idea of simple effects is that you understand an interaction by holding one effect constant and investigating the effect of the other one.

I think the easiest way to tackle this one is to hold dose constant and look at the effect of age for each dose. The significant interaction means that we would expect the *slopes* for each dose to be different. So, (i) condition on dose and (ii) work out the age slope for each dose. (Getting some reasonable way towards this is two points.)

That's as far as you need to go with this idea, since you don't have the actual data to work with. I, however, do:

```
rats %>% nest(-dose) %>%
    mutate(line=map(data, ~lm(resttime~age, data=.))) %>%
    mutate(slope=map_dbl(line, ~pluck(., "coefficients", 2)))
```

```
## # A tibble: 4 x 4
##   dose  data              line      slope
##   <fct> <list>            <list>    <dbl>
## 1 0     <tibble [15 x 2]> <S3: lm>  0.325
## 2 10    <tibble [15 x 2]> <S3: lm>  5.61
## 3 20    <tibble [15 x 2]> <S3: lm>  8.43
## 4 30    <tibble [15 x 2]> <S3: lm> 12.5
```

The slopes with age do indeed get bigger as dose gets bigger.

The strategy is: first make mini-data-frames `data` containing everything but `dose` for each dose (that is, they contain `resttime` and `age`). For each of those doses, we run a regression of rest time on age, saving everything for the moment (so that `line` is a column of fitted regression objects). From each of *those*, we pull out the slope. This is done by remembering (or looking up) that the intercept and slope(s) are in a thing called `coefficients`, with the intercept first, so the one slope we want is the second thing in there.

The other way around is to condition on age and look for a dose effect. This is trickier because age is quantitative and there are a whole lot of different values. One way to make this fly is to make age into a categorical variable first. This loses some information about age, but it's a reasonable way to think about simple effects in this context. The idea here is that the nature and/or size of the dose effect changes depending on which age group the rat is in.

That's the kind of thing you need to say if you're going this way: (i) the effect of dose conditioning on age, and (ii) some way of making age categorical.

I continue, extra to the question as asked:

The standard R way of chopping quantitative variables into categories is called `cut`. You pass `cut` a quantitative variable and a thing `breaks` that says where the category boundaries are. (This is like making classes for a histogram.) So what boundaries are we going to use? How about quartiles:

```
quantile(rats$age)
##    0%   25%   50%   75%  100%
##  5.00  7.00 11.00 13.25 16.00
```

and then

```
rats %>% mutate(age_f=cut(age, breaks=c(4, 7, 11, 13.25, 18))) -> rats2
rats2
## # A tibble: 60 x 4
##    dose    age resttime age_f
##    <fct> <dbl>    <dbl> <fct>
##  1 0        12       55 (11,13.2]
##  2 0         9       59 (7,11]
##  3 0        15       52 (13.2,18]
##  4 0         5       39 (4,7]
##  5 0         9       57 (7,11]
##  6 0        11       62 (7,11]
##  7 0         6       53 (4,7]
##  8 0         7       59 (4,7]
##  9 0        12       47 (11,13.2]
## 10 0         8       59 (7,11]
## # ... with 50 more rows
```

My habit with `cut` is to make the endmost `breaks` something that is definitely smaller than the smallest value, and definitely bigger than the biggest one. That way, I don't get messed around with values equal to the smallest and largest (which category do they go in?).

R makes category names that are "half-open intervals": the square bracket means that the value named is *in* the category, and the round bracket means that the value is *not in* the category. For example, age 7 months is in the 4–7 category rather than the 7–11 one.

You could now do a one-way ANOVA of rest time on dose for each age category. From what I saw on the graph, though, I think the dose differences are always going to be significant, and so I think the main interest is how big they are. Hence, rest time means for each dose within each age group:

```
rats2 %>% group_by(age_f, dose) %>%
    summarize(m=mean(resttime))
## # A tibble: 16 x 3
## # Groups:   age_f [4]
##    age_f     dose      m
##    <fct>     <fct> <dbl>
##  1 (4,7]     0      54
##  2 (4,7]     10     80.4
##  3 (4,7]     20    116.
##  4 (4,7]     30    133
##  5 (7,11]    0      59.2
##  6 (7,11]    10    101.
##  7 (7,11]    20    141.
##  8 (7,11]    30    171.
##  9 (11,13.2] 0      51.7
## 10 (11,13.2] 10    126.
## 11 (11,13.2] 20    162.
## 12 (11,13.2] 30    216
## 13 (13.2,18] 0      57.5
## 14 (13.2,18] 10    126.
## 15 (13.2,18] 20    193
## 16 (13.2,18] 30    252.
```

The rest time means all get substantially bigger as dose gets bigger, but I think you can see that they get bigger *faster* as age (group) increases.

The place where I got this data from called this a "synergistic" relationship, in that the effect of an increased dose *and* age is bigger than you would expect from considering the increases in dose and age separately. (A fixed increase in dose has a *bigger* impact on rest time in older rats.) This comes from the graph. The idea is that, having found a significant interaction, we try to explain why it came out significant.

The point of this question is to suggest a way in which simple effects might work *for these data*. There are different ways to say it, but you need to get at the idea of holding one explanatory variable fixed ("focus on each value of dose", say) and assessing the effect of the other one on rest time.

Question 2 (19 marks)

An experiment is conducted, with the results shown in Figure 4. The columns are: treatment (labelled A, B, or C), time point (labelled T1, T2, and T3), subject ID, and a response variable value, called `y`. There are 27 observations in total.

(a) (1 mark) How many different treatments does each subject undergo?

My answer: Just one. (This means, looking ahead, that treatment will be a "between-subjects factor", because each treatment is done by different subjects.)

There are certainly three treatments *altogether*, but no subject does more than one (they do the same one at all three time points). Check the data. If a subject did more than one, we would have something a lot more complicated to analyze, with two "within-subjects factors", that would be time *and* treatment. (I should look into whether a mixed-models approach handles these as smoothly as I think it does.)

(b) (2 marks) What feature of the data indicates that a repeated-measures analysis of variance will be necessary? Explain briefly.

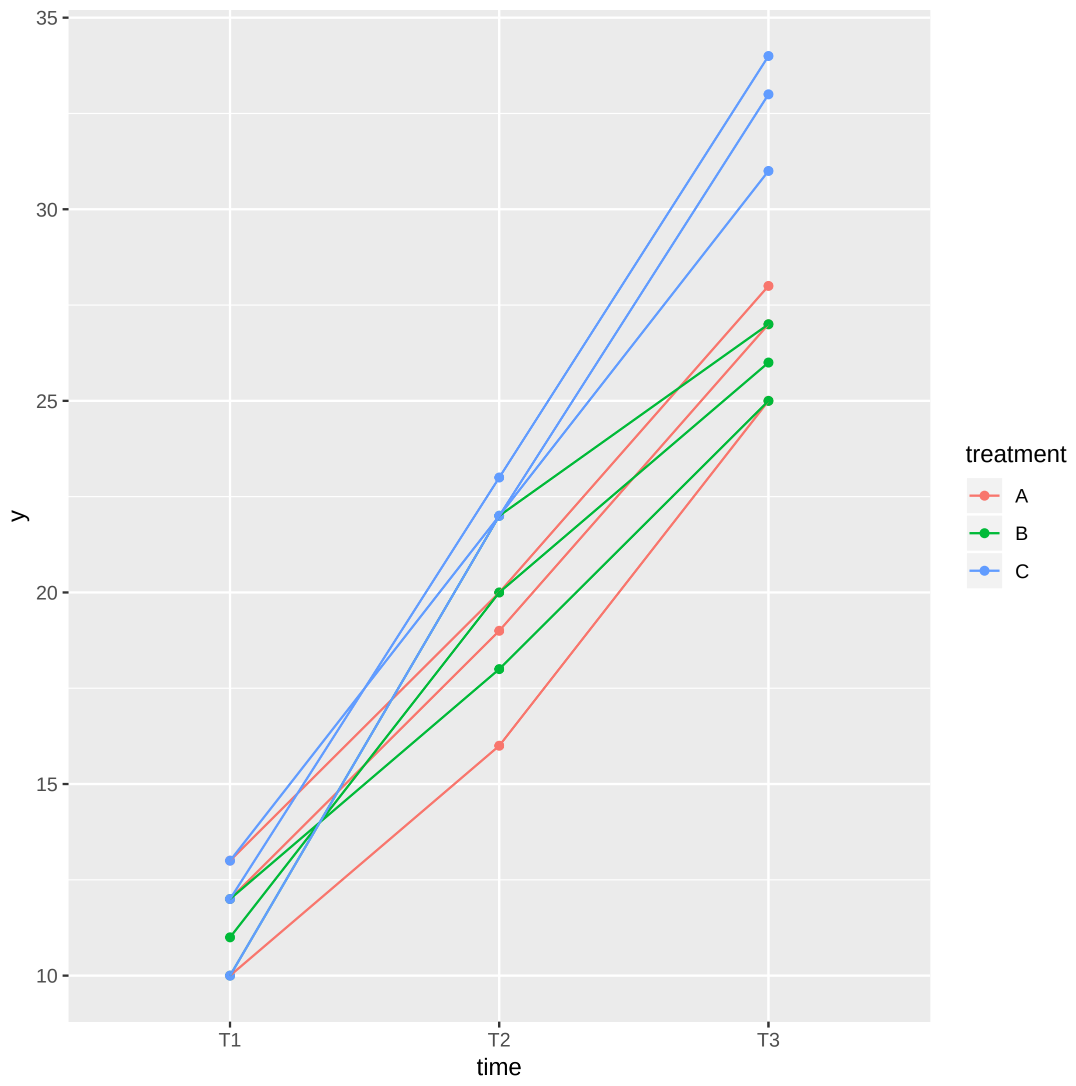
My answer: Each subject is measured several times (at the three different time points). (The three measurements on the same subject are likely to be correlated, so we won't have independent observations.)

I tried to be sympathetic here if you thought that each subject did more than one treatment (in (a)).

(c) (3 marks) Give code to create a spaghetti plot for these data (that is, a plot showing how the response variable changes over time for each treatment, with the values for each subject joined by lines). Assume that the data frame shown in Figure 4 is called `rm`.

My answer:

```
ggplot(rm, aes(x=time, y=y, colour=treatment, group=subject)) +
  geom_point()+ geom_line()
```



Note that the `colour` and `group` are *different* for these. (This is the code that was used in Figure 24.)

I'll take omitting the `geom_point` since this still shows how things change over time.

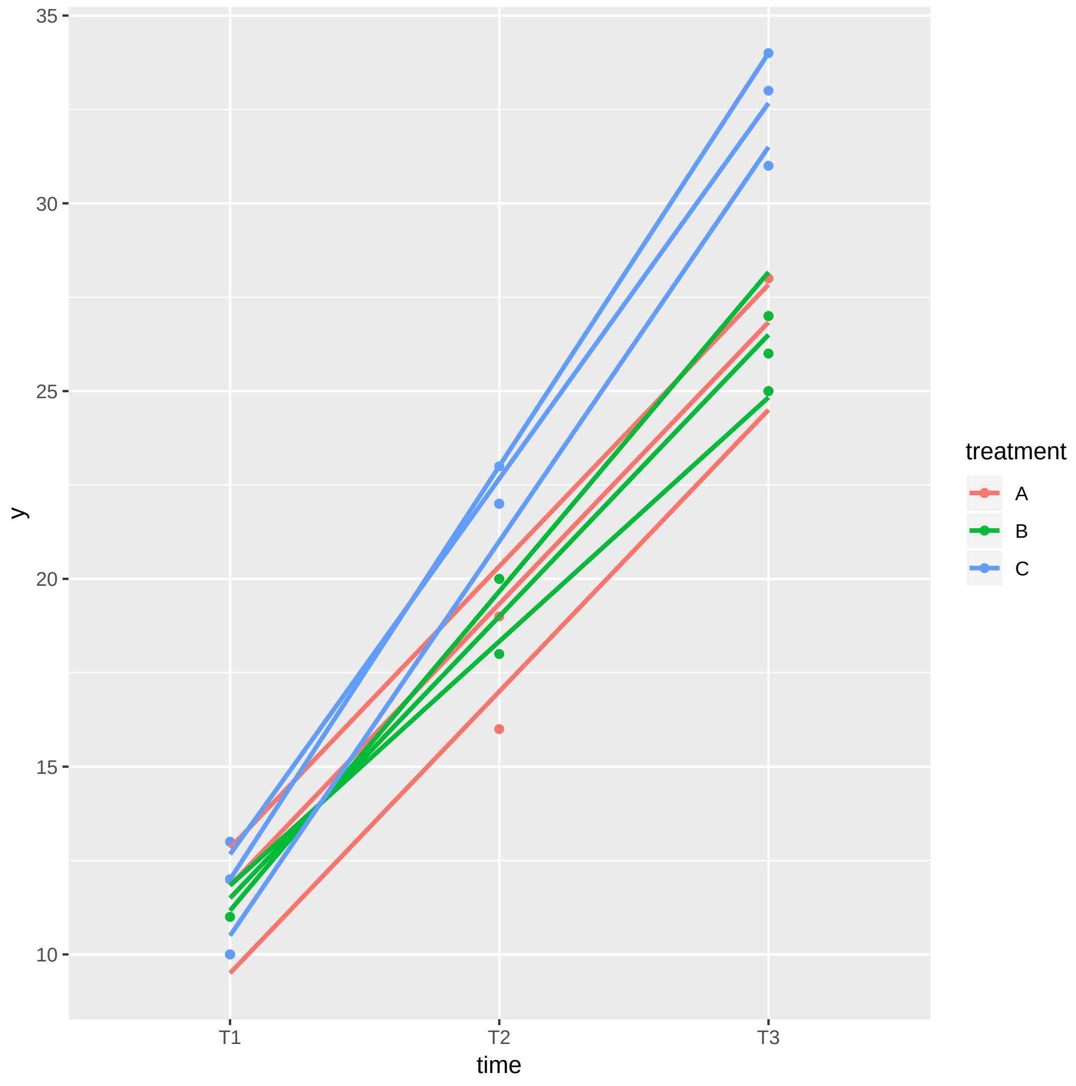
Getting `group` and `colour` the wrong way around counts for me as one error, so two marks.

The data, you will note, is already in long format, so this part is straightforward enough. (The fun and games happens later.) So having a `gather` here is an error. The clue is that the data frame already has all the things you appear to need in the `aes`; if it were wide, you'd have a column for each time point which you would have to gather together into a column called something like `time`, but we already have that.

The "fun and games" later is that we'll have to take this long data frame and make it wide for the repeated measures analysis. This may be backwards from what you've seen before, but a key part of being a data analyst is flexibility of thinking.

I also saw something like this:

```
ggplot(rm, aes(x=time, y=y, colour=treatment, group=subject)) +
  geom_point()+ geom_smooth(method="lm", se=F)
```



This puts a regression line through the points for each subject. It sort of gets at the idea (and you certainly do see that the blue points are going up faster than the red and green ones), but I think it's nicer to join up the actual observations for each subject, which is what `geom_line` does (so that you see *whether* the time trends are linear instead of *assuming* that they are). Minus one, therefore. (I figured I should try it and see how it looked rather than automatically deducting one.)

(d) (2 marks) What advantage does a spaghetti plot have over an interaction plot for repeated-measures data? Explain briefly.

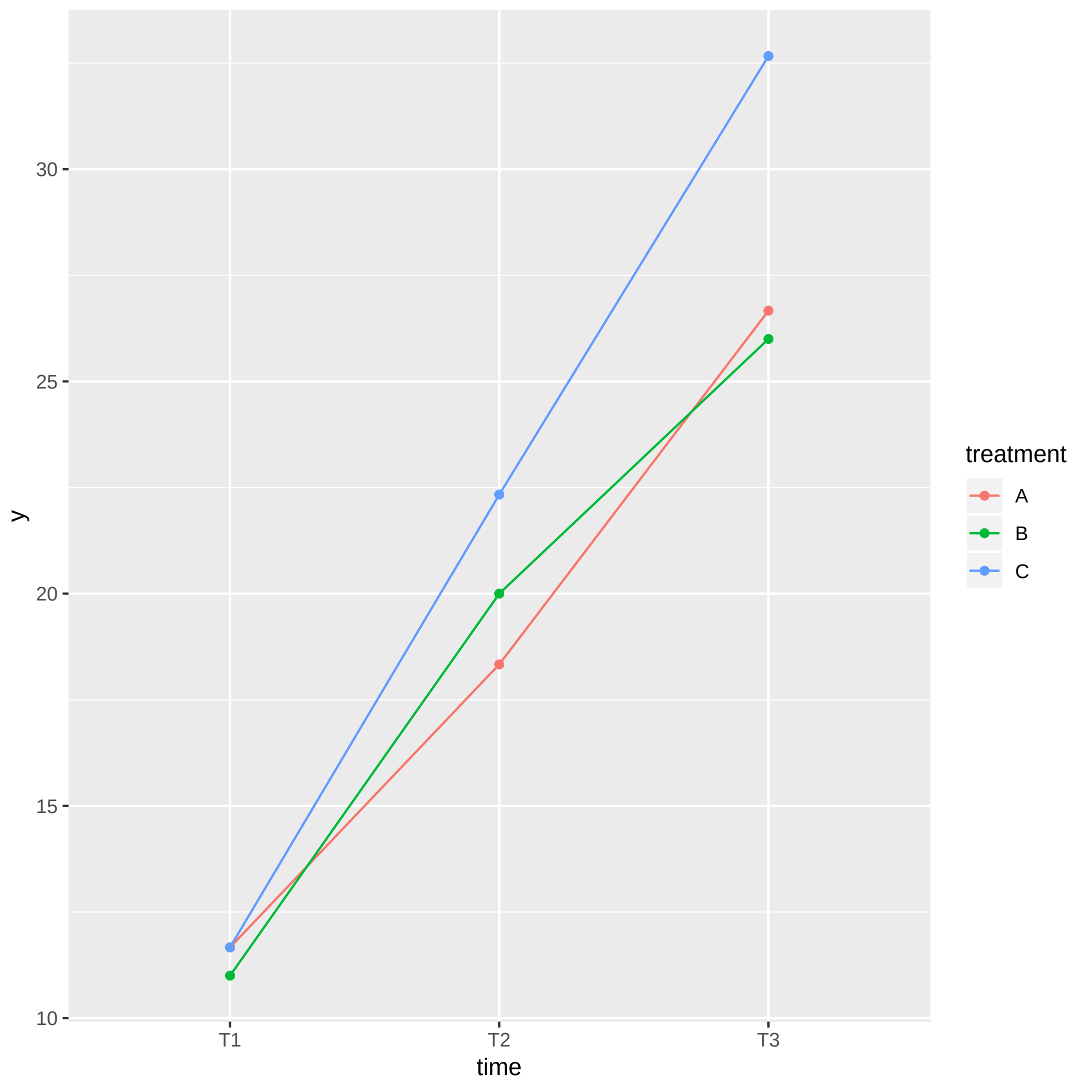
My answer: I think the key thing is that the spaghetti plot shows *variability* as well, so that you can judge whether the treatments are "really" different, given how much variability there is. (Or, likewise, whether there is "really" a time effect.)

Making the point that the spaghetti plot shows "all the data" is part of an answer, but not really a complete one. For both points, you need to get at either variability or assessing the significance of effects.

You can spell `colour` the British way or the American way ("color"); they both work.

Extra: the interaction plot looks like this:

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



I had to do it this way, because I didn't calculate any group means before.

This shows the increasing difference between C and the other treatments over time more clearly, but doesn't give a sense of whether that's a "real" difference or whether the lines are "approximately parallel".

(e) (3 marks) The spaghetti plot produced by your code is shown in Figure 24. What does this plot suggest about likely (i) treatment effect, (ii) time effect, (iii) treatment-time interaction? Explain briefly.

My answer: Treatment C (in blue) appears overall better than the other treatments; there doesn't seem to be much difference between treatments A and B. Thus, overall, we would expect to see a treatment effect. (Point out some treatment difference.)

All of the `y` values (regardless of treatment) are going up over time, so we would expect to see a time effect.

I think the blue spaghetti strands are getting further above the red and green ones: that is, the difference between treatments is getting bigger over time. (You're welcome to disagree on this one; for example you can say that the rates of increase over time for the different treatments are "approximately the same", and therefore that you don't expect to see an interaction. Consistency of logic is what I care about.)

This is the same kind of thing as the ANCOVA question, and I see I painted myself into the same corner when it comes to the interpretation, which I guess I will have to grade the same way.

Convince me of how you know there is a time effect, treatment effect and interaction effect (or no interaction effect).

(f) (4 marks) What code would run a suitable repeated-measures ANOVA, using `Manova`? Pay close attention to the layout of the data in Figure 4, which is the data frame `rm`.

My answer: The data in Figure 4 is in "long format" with each observation of `y` in one row (and thus multiple rows per subject). This means that you have several things to do:

- put the data in wide format
- create a response variable
- run `lm`
- create the within-subject structure
- pass that into `Manova`.

Thus, something like this:

```
rm %>% spread(time, y) -> rm_wide
response=with(rm_wide, cbind(T1, T2, T3))
rm.1=lm(response~treatment, data=rm_wide)
times=colnames(response)
times.df=data.frame(times)
Manova(rm.1, idata=times.df, idesign=~times)
##
## Type II Repeated Measures MANOVA Tests: Pillai test statistic
##                 Df test stat approx F num Df den Df    Pr(>F)
## (Intercept)      1   0.99751  2399.02      1      6 4.857e-09 ***
## treatment        2   0.70412     7.14      2      6  0.025902 *
## times            1   0.99876  2010.30      2      5 5.437e-08 ***
## treatment:times  2   1.34513     6.16      4     12  0.006206 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

My marking plan is: one mark for making it wide, one for making the response, and two for the rest of it.

There is some subtlety here. When you create the wide data frame, the column you're spreading, `time`, will give its values to the *names* of the new columns, and so the new columns will be called `T1`, `T2`, and `T3`. I want to see how well you understand this.

If you don't create a wide data frame, you are likely to get into further trouble later (thus you are likely to end up with 2 rather than 3). This is kind of a clue to you: if you are thinking "what am I going to make my response from?", that's a hint to think about what kind of stuff you need to make the response out of (columns called T1, T2 and T3, maybe) and how you might get those.

From here on, it's exactly the standard thing as I did it in class.

I asked for the `Manova` way of doing it, so full credit is for the above. But there is definitely something (probably two) for the mixed-models way, if you come up with that instead:

```
rm.2=lmer(y~treatment*time+(1|subject), data=rm)
```

This uses the original long-format data set directly, so there is no need for `spread`.

Extra: if you have the development version of `tidyr` (which you download from Github), instead of doing `spread` you can also do this:

```
rm %>% pivot_wide(-time, names_from=time, values_from=y)
## # A tibble: 9 x 5
##   treatment subject    T1    T2    T3
##   <chr>     <chr>   <dbl> <dbl> <dbl>
## 1 A         S1         10    16    25
## 2 A         S2         12    19    27
## 3 A         S3         13    20    28
## 4 B         S4         12    18    25
## 5 B         S5         11    20    26
## 6 B         S6         10    22    27
## 7 C         S7         10    22    31
## 8 C         S8         12    23    34
## 9 C         S9         13    22    33
```

This is exactly the same as `spread`, except that you might like the syntax better: you tell it where to get the names for the new columns in the wide data frame (`time`) and where to get the values from (`y`). It also has a lot more features to handle more complicated cases of making data wide. (The new counterpart of `gather` is called `pivot_long`.) `gather` and `spread` are not going anywhere; the new functions are just alternatives to them that might be easier to use.

(g) (2 marks) The analysis for which you gave code in the previous part is shown in Figure 5. What do you conclude from this, in the context of the data?

My answer: Look first at the interaction. This is significant (P-value 0.006), and shows that the effect of treatment depends on the time point you are looking at (or, that the effect of time is different for each treatment).

Further than that we do not go: the significant interaction *is* the finding. (If the interaction *had not* been significant, then, for this kind of analysis only, we could have gone on to interpret the main effects without refitting.) In the light of what I said above, I'm not going to penalize you if you *do* go on. I do, however, want to see something about effects of treatment over time, which is what "in the context of the data" is supposed to be directing you towards.

Extra: the mixed-models analysis comes to the same conclusion, but with a smaller P-value:

```
drop1(rm.2, test="Chisq")
```

```
## Single term deletions
##
## Model:
## y ~ treatment * time + (1 | subject)
##                Df    AIC   LRT   Pr(Chi)
## <none>            102.53
## treatment:time  4 120.44 25.91 3.299e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There is no test for `subject`; we are *assuming* that the subjects will differ from another, and that each one contributes a random effect to `y` (some subjects have high scores on `y` and some have low, regardless of time or treatment).

Extra extra: one of the assumptions hiding behind repeated-measures ANOVA done with MANOVA is "sphericity", which is that the variance of each pair of differences between time-period means is constant. This plays the role of equal group spreads that you'll be familiar with from standard ANOVA. To be complete, what we should do here is this:

```
rm.3=Anova(rm.1, idata=times.df, idesign=~times, type=3)
summary(rm.3, multivariate=F)
```

```
##
## Univariate Type III Repeated-Measures ANOVA Assuming Sphericity
##
##                 Sum Sq num Df Error SS den Df  F value    Pr(>F)
## (Intercept)     3211.1      1   27.111      6 710.6557 1.840e-07 ***
## treatment         64.5      2   27.111      6   7.1393 0.0259021 *
## times            338.9      2   12.889     12 157.7586 2.419e-09 ***
## treatment:times   41.5      4   12.889     12   9.6552 0.0009899 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Mauchly Tests for Sphericity
##
##                 Test statistic  p-value
## times                  0.29964 0.049149
## treatment:times       0.29964 0.049149
##
##
## Greenhouse-Geisser and Huynh-Feldt Corrections
##  for Departure from Sphericity
##
##                  GG eps Pr(>F[GG])
## times           0.58811  3.285e-06 ***
## treatment:times 0.58811   0.008332 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                    HF eps   Pr(>F[HF])
## times           0.6461293 1.182316e-06
## treatment:times 0.6461293 6.137921e-03
```

What you do with this is to start with the Mauchly test for sphericity for the thing you're testing (the interaction). Here, the P-value is 0.049, which is *just* significant. This means that sphericity is rejected (or at least questionable), so we should be careful. That means that the P-value we got before is not completely to be trusted. There are two corrections to the P-value to allow for non-sphericity, due to Greenhouse and Geisser (GG) and Huynh and Feldt (HF), which are at the bottom. GG is “safe”, but can produce a P-value that is bigger than it needs to be (0.008 here); HF gives a slightly more significant 0.006. These are both very close to the 0.006 that we had before, so the impact of rejecting sphericity is small, and the conclusions we had before about the interaction are still sound.

Extra-cubed: you might think about doing something like simple effects to understand the interaction. In a repeated measures, the easiest way to do that is to condition on *time*: that is, to look at the treatment effects at each time. The reason for this is that you then have *one* observation per individual, and you can do a standard one-way ANOVA. If you instead condition on *treatment*, you still have repeated measures, because each person on a particular treatment was measured at all three time points: you still have the within-subjects factor of time, but you have lost the between-subjects factor of treatment.

Because each simple-effects analysis has only one time point, you can actually use either the long data or the wide data. It looks different, but the result is the same. Here's time point T1 from the long data:

```
rm %>% filter(time=="T1") %>%
    aov(y~treatment, data=.) -> s1
summary(s1)
##             Df Sum Sq Mean Sq F value Pr(>F)
## treatment    2  0.889  0.4444   0.235  0.797
## Residuals    6 11.333  1.8889
```

and from the wide data, which I called `rm_wide`:

```
aov(T1~treatment, data=rm_wide) -> s1a
summary(s1a)
##             Df Sum Sq Mean Sq F value Pr(>F)
## treatment    2  0.889  0.4444   0.235  0.797
## Residuals    6 11.333  1.8889
```

No differences in y between the treatments at time point T1. The second time point was where things started to show up:

```
rm %>% filter(time=="T2") %>%
    aov(y~treatment, data=.) -> s2
summary(s2)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## treatment    2  24.22  12.111   4.192 0.0726 .
## Residuals    6  17.33   2.889
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Or, actually, not quite, but at least nearer.

At the third time point, the gap between treatment C and the others appears to be bigger enough to be significant (and presumably it is, because *something* ought to be making that interaction significant):

```
rm %>% filter(time=="T3") %>%
    aov(y~treatment, data=.) -> s3
summary(s3)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## treatment    2  80.89   40.44   21.41 0.00186 **
## Residuals    6  11.33    1.89
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
TukeyHSD(s3)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = y ~ treatment, data = .)
##
## $treatment
##           diff       lwr       upr     p adj
## B-A -0.6666667 -4.109784  2.776450 0.8283109
## C-A  6.0000000  2.556883  9.443117 0.0042124
## C-B  6.6666667  3.223550 10.109784 0.0024602
```

Now treatment C does indeed have a significantly bigger y value than the others.

(h) (2 marks) Why do you think that taking out the first time point *would not* change the significance of the interaction as you found it in the previous part, in contrast to the dogs example in class? (Taking out the first time point may change the P-value, but would not change whether or not the interaction is significant at, say, $\alpha = 0.05$.)

My answer: For me, the interaction was significant, so I need to justify why the interaction would *still* be significant if I took out time point T1.

I would justify that by going back to the spaghetti plot and saying that the gap between treatment C and the others is still getting bigger between time points T2 and T3. It doesn't seem to be the case that everything was the same at the first time point and everything differed by a constant amount after that, which was how the class dogs example played out.

If you somehow thought there was no significant interaction, you would need to make the argument that there would continue to be no interaction if you take out time point T1. You could do this by saying something like "all the treatments are mixed up at all three time points", implying that taking one of the time points out won't change anything that way. I don't really agree with this, but it's a reasonable line of argument to make this point, especially if it's what you said about the interaction from the spaghetti plot (I will check back to see).

Since we are talking about interactions here, your answer will need to talk about treatment effects over time (otherwise you are talking only about the time effect).

You could also make the argument that the P-value is *very* small here, and taking out one time point would have to do a lot of "damage" to the P-value to make it non-significant. This is, to my mind, a less insightful argument, but it answers the question, so it is good.

Question 3 (23 marks)

Crude oil samples were taken from sandstone of different types, known as "zones". The three zones are Wilhelm, Sub-Mulinia, and Upper (Mulinia). The zone names are abbreviated in the data set. The aim is to see whether the following measurements are associated with the zone of sandstone from which the oil sample was taken:

- vanadium
- iron
- beryllium
- saturated hydrocarbons
- aromatic hydrocarbons.

A random sample of the data set is shown in Figure 6.

(a) (2 marks) Why might MANOVA be a sensible method of analysis for these data? Explain briefly.

> **My answer:** We want to see how zone impacts any or all of the five response variables listed above. (Or, there are five response variables, more than one, and we want to know if and how they depend on zone.)
>
> I inexplicably omitted `vanadium` from my analysis, so replace "five" with "four" and you're still good.
>
> I want to see a statement or implication that the five (four) response variables may *depend on zone*, in the same way that the single response variable in ANOVA may depend on some categorical explanatory variable(s).

(b) (2 marks) A MANOVA analysis is shown in Figure 7. What do you conclude from it, in the context of the data?

> **My answer:** The P-value is (very) small, so the zone of sandstone from which the oil sample was taken affects the mean of one or more of the response variables.
>
> Further than that, we can not say (for example, *which* of the response variables, or which zones are higher or lower on that response). All that we know so far is that there is a difference *somewhere*.

(c) (2 marks) Given the results of the MANOVA, why might we want to do a discriminant analysis? Explain briefly.

> **My answer:** The MANOVA does not tell us anything about which zones are different on which variables (mention, ideally, both of those things), but a discriminant analysis will give us some insight.
>
> I was scanning for a word like "how" that gets at these ideas.

(d) (2 marks) Based on Figure 8, how many linear discriminants should we use? Explain briefly.

My answer: Look at the Proportion of Trace, and make a decision about whether LD2's value is small. So one of these two:

- 0.1754 is reasonably large, so we should use both LD1 and LD2.
- 0.1754 is small compared to 0.8246, so we should use only LD1.

I actually think the first one is better, but I would accept either.
The other question I sometimes ask, which is not this one, is "how many linear discriminants are there?", to which the answer is $min(4, 3-1) = 2$. But that is not an answer to *this* question.

(e) (3 marks) Which of the original variables contribute to LD1, and how do they do so? (For example, what kind of values would make LD1 large?)

My answer: I would say that `beryllium` has a negative coefficient, `saturated` has a positive one, and the other two are close to zero. Thus LD1 would be large if `saturated` is large and/or `beryllium` is *small*.

It's really "either this, or that, or both", so you could justify either "and" or "or", and I accept either.

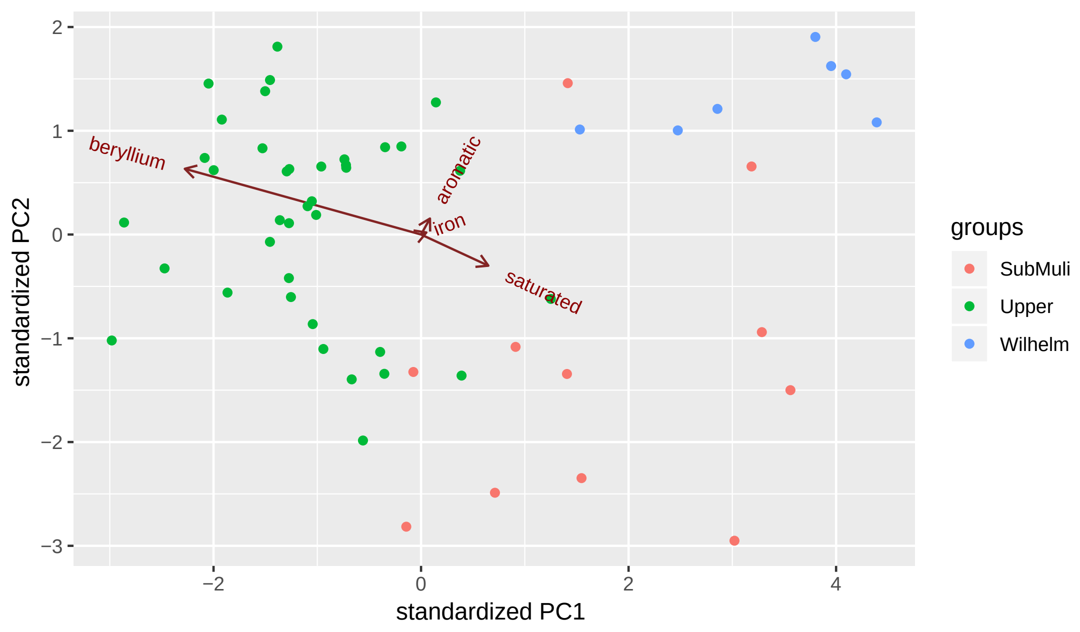
You might take the point of view that only `beryllium` matters, and therefore small `beryllium` goes with large LD1. That's fine too.

(I realize that you can also look at the group means on Figure 8, and come to more or less the same conclusion (with the addition of low iron). That wasn't what I intended, but it's fine if you did it.)

Extra: the same two variables contribute to LD2 as well. You can make a call about whether `aromatic` contributes (positively) to LD2 as well, but it seems clear to me that `iron` has nothing to say about separating the zones at all.

To pursue this "extra", I did a biplot (which I didn't put on the original exam) which clarifies some of these issues:

```
ggbiplot(crude.2, groups=crude$zone)
```



beryllium and **saturated** point very much left and right (respectively), so that LD1 will be high if **saturated** is high and/or **beryllium** is low. (The biplot makes this a lot easier to see.) Nothing points very much up or down (which is an argument for not considering LD2, but see later when we talk about whether and how the zones are separated). **beryllium** points slightly up, **saturated** slightly down, and **aromatic** points a tiny bit up. This supports the coefficients on LD2 that I discussed above.

(f) (2 marks) I created a plot of the LD scores, shown in Figure 25. I did this by making predictions, then creating a data frame `d` containing both the original data and the predictions. The code I used to make the plot is as shown. (The numbers beside the points are the numbers of the oil samples in the data set.)

Would you say that the zones are relatively distinct, or not? Explain briefly.

My answer: I would say that they are, with Upper mainly on the left, Wilhelm top right, and SubMuli bottom right, for the most part. There is a small amount of mixing of the points, but not much.

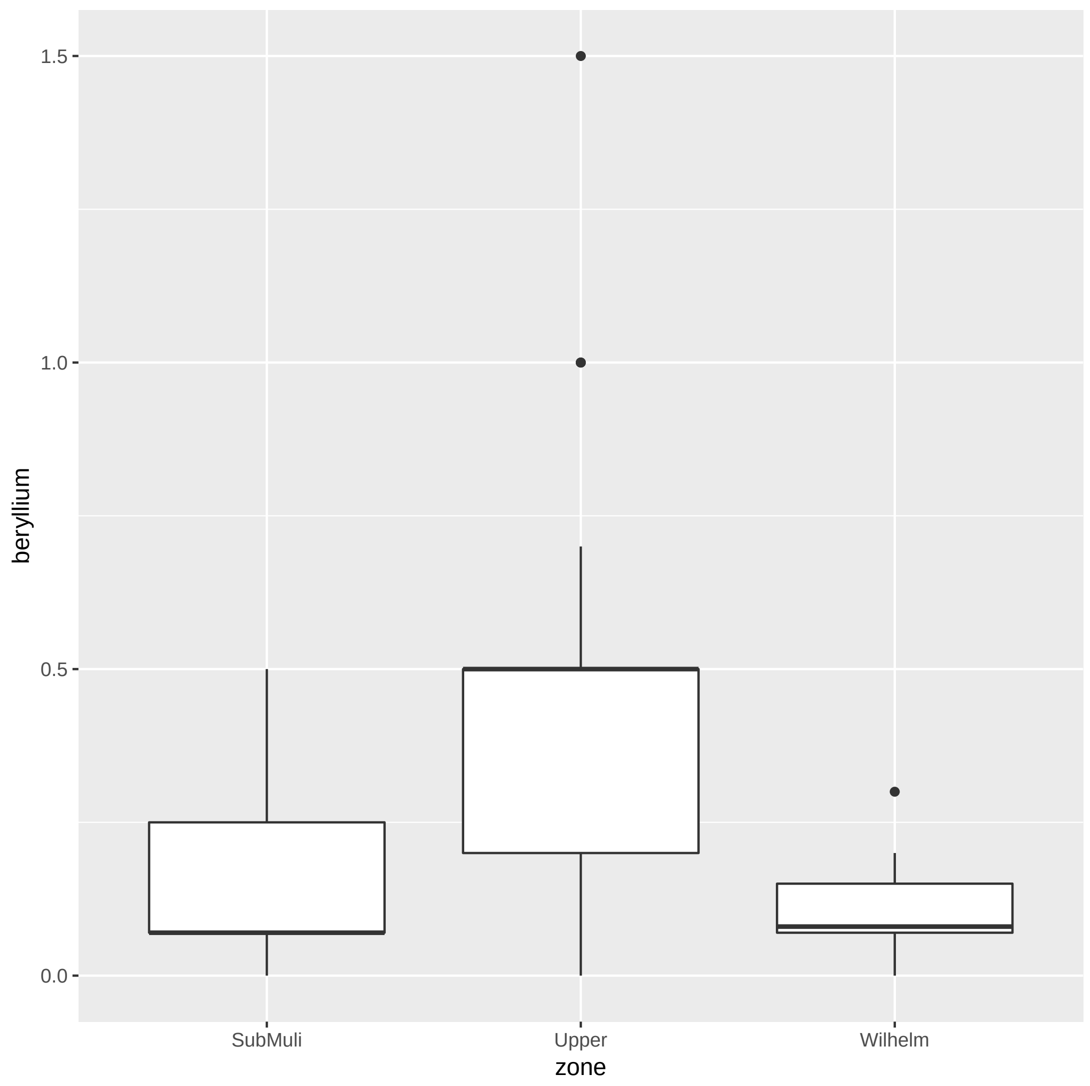
You need to make a call about the distinctness, and then say something about how you know (such as where on the plot the zones are typically found).

(g) (2 marks) For oil samples from the Upper zone, what in terms of the original variables distinguishes them from the other zones? Explain briefly.

My answer: These have mostly a *low* score on LD1. Look back at your answer to (e) to see that they must have a *high* value of `beryllium` and a *low* value of `saturated`.

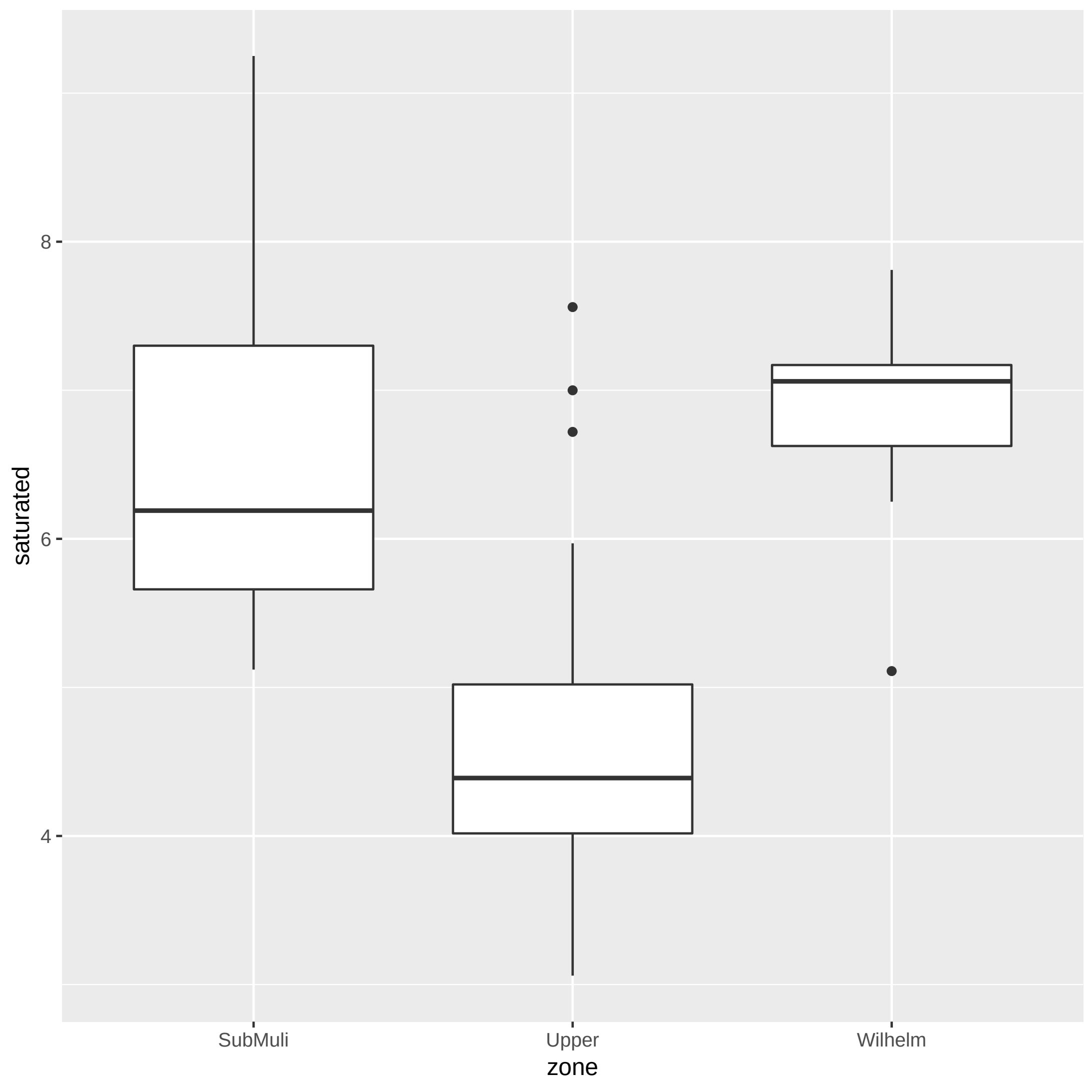
Was I right about that? Boxplots will tell us:

```
ggplot(crude, aes(x=zone, y=beryllium))+geom_boxplot()
```



and

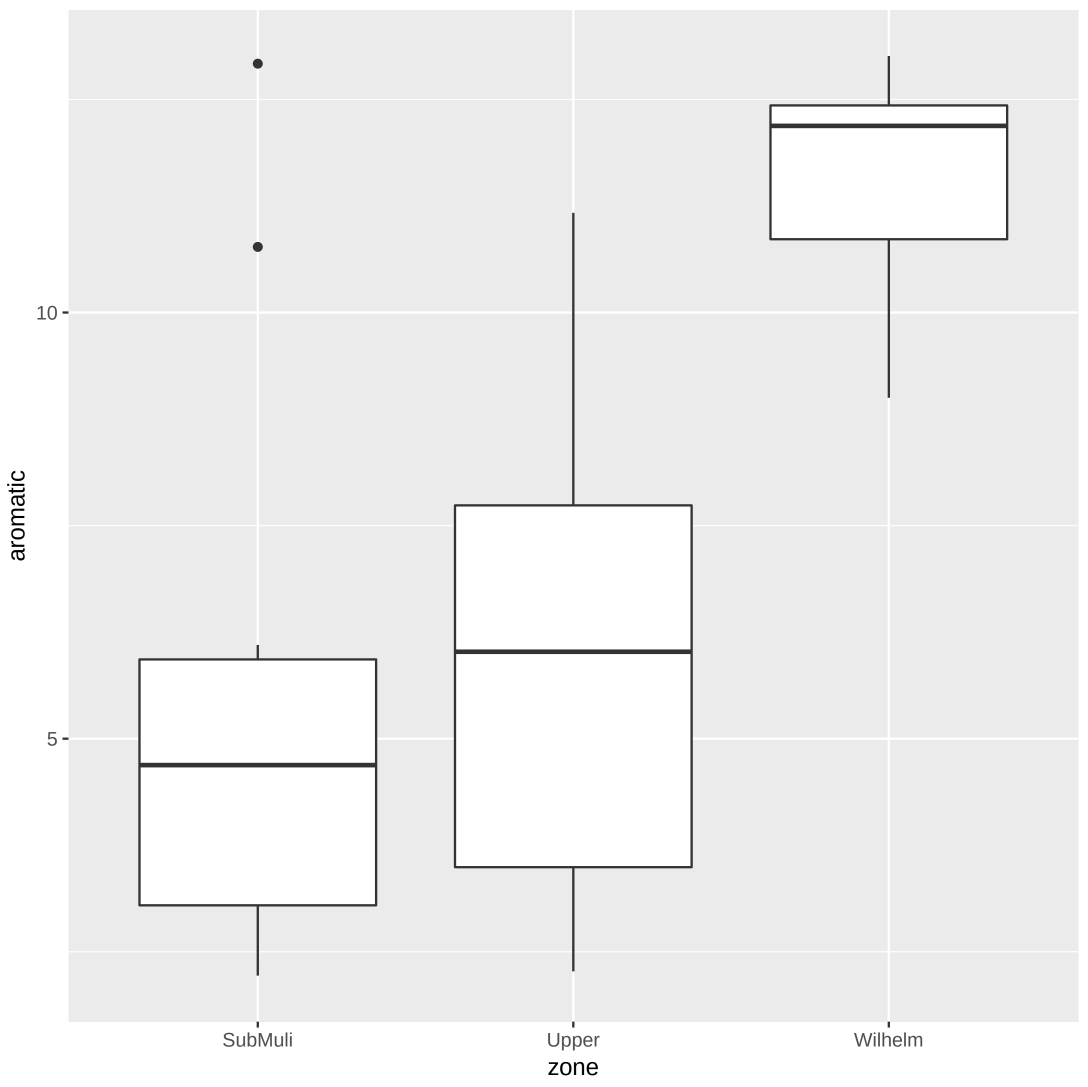
```
ggplot(crude, aes(x=zone, y=saturated))+geom_boxplot()
```



`Beryllium` is generally higher and `saturated` generally lower for the Upper zone. This is how the Upper zone is different from the others.

Extra: what distinguishes `SubMuli` from `Wilhelm`? This is up and down on the plot, so it has to do with LD2. If you look back at my biplot, the distinction between red and blue seems to be in the direction of `aromatic`, so this might do something to separate them:

```
ggplot(crude, aes(x=zone, y=aromatic))+geom_boxplot()
```



The Wilhelm samples are consistently high on `aromatic`. The `SubMuli` ones are usually lower, but with a couple of high outliers (the red dots in with the blue ones on the plot). (The `Upper` samples are all over the place on `aromatic`, rather as we expect.)

(h) (3 marks) My data frame `d` contains all the original data plus a predicted zone for each observation. What R code would use `d` to calculate the proportion of all the observations that were misclassified, that is, for which the predicted zone is different from the actual zone? (If you also wish to calculate the proportion that were *correctly* classified, that is fine too.)

My answer: The true zones are in the column `zone` (as shown in Figure 6), and the predicted zones are in a column of the predictions that is always called `class`. (You need to remember this.) To work out the misclassification rate, first create a column that indicates whether that oil sample was correctly classified, then count up how many observations that were correctly and incorrectly classified, then work out the proportion. This is how I would do it:

```
d %>% mutate(is_correct=ifelse(zone==class, "correct", "wrong")) %>%
    count(is_correct) %>%
    mutate(proportion=n/sum(n))
```

```
## # A tibble: 2 x 3
##   is_correct     n proportion
##   <chr>      <int>      <dbl>
## 1 correct       51     0.911
## 2 wrong          5     0.0893
```

The `n` is the column of frequencies that comes out of the `count`. It is also correct to do something like this that explicitly names the counts:

```
d %>% mutate(is_correct=ifelse(zone==class, "correct", "wrong")) %>%
    group_by(is_correct) %>%
    summarize(how_many=n()) %>%
    mutate(proportion=how_many/sum(how_many))
```

```
## # A tibble: 2 x 3
##   is_correct how_many proportion
##   <chr>         <int>      <dbl>
## 1 correct          51     0.911
## 2 wrong             5     0.0893
```

Or you can short-cut things by using a TRUE and a FALSE:

```
d %>% mutate(is_correct=(zone==class)) %>%
    count(is_correct) %>%
    mutate(proportion=n/sum(n))
```

```
## # A tibble: 2 x 3
##   is_correct     n proportion
##   <lgl>      <int>      <dbl>
## 1 FALSE          5     0.0893
## 2 TRUE          51     0.911
```

This actually also works, and saves you a step:

```
d %>% count(zone==class) %>%
    mutate(proportion=n/sum(n))
```

```
## # A tibble: 2 x 3
##   `zone == class`     n proportion
##   <lgl>           <int>      <dbl>
## 1 FALSE               5     0.0893
## 2 TRUE               51     0.911
```

You might also be able to persuade base R `table` to help you. If you tabulate the observed and predicted zones you get part of the way:

```
tab=with(d, table(obs=zone, pred=class))
```

```
tab
##          pred
## obs        SubMuli Upper Wilhelm
##   SubMuli        8     1       2
##   Upper          2    36       0
##   Wilhelm        0     0       7
```

but then you have to figure out a way to count up the off-diagonal entries and work out what they are as a proportion, for example:

```
total=sum(tab)
wrong=sum(tab[row(tab)!=col(tab)])
wrong/total
## [1] 0.08928571
```

Base R is a lot less friendly than the Tidyverse. *I* know that the line defining `wrong` means "sum up the entries of `tab` for which the row number and column number are different", but if you do, or that this is what you need, I'm impressed.

Whichever way you do it, about 9% of the oil samples were misclassified, which seems about right given how much overlap there was on the plot (not much).

Marks: one point for creating a column of correct-wrong values (either explicitly or implicitly like the last way), one for counting up how many corrects and wrongs you had, and one for calculating the proportions. Something less direct is also good if it will work.

If you used `wt=n` in your solution, it probably won't work. The place where you need it is if you're summing up some frequencies you *already have*, rather than counting the number of rows. Here there is one correct reason to use it, which is about like this:

```
d %>% count(zone, class) %>%
    mutate(is_correct=(zone==class)) %>%
    count(is_correct, wt=n) %>%
    mutate(proportion=n/sum(n))
## # A tibble: 2 x 3
##   is_correct     n proportion
##   <lgl>      <int>      <dbl>
## 1 FALSE          5     0.0893
## 2 TRUE          51     0.911
```

Here, I counted up *all* the combinations of zone and class *first*, *then* created `is_correct`. I already had a data frame with an `n` in it (from the first `count`), and I wanted to get the total number of rights and wrongs, not the number of rows that were right or wrong. The `n` in the last line, confusingly, is from the *second* `count`. One person did it this way and got it right; a funky but correct way to three points. (I checked again to make sure that other people who had `wt=n` put it in a place that would not work.)

(i) (2 marks) What *change* to your code of the previous part would obtain the misclassification proportions for *each* zone?

> **My answer:** A `group_by(zone)` before you count up. Thus, for example,
>
> ```
> d %>% mutate(is_correct=ifelse(zone==class, "correct", "wrong")) %>%
>     group_by(zone) %>%
>     count(is_correct) %>%
>     mutate(proportion=n/sum(n))
> ```
>
> ```
> ## # A tibble: 5 x 4
> ## # Groups:   zone [3]
> ##   zone    is_correct     n proportion
> ##   <chr>   <chr>      <int>      <dbl>
> ## 1 SubMuli correct        8     0.727
> ## 2 SubMuli wrong          3     0.273
> ## 3 Upper   correct       36     0.947
> ## 4 Upper   wrong          2     0.0526
> ## 5 Wilhelm correct        7     1
> ```
>
> The `group_by` can be before or after the `mutate` but definitely needs to be before the `count` or `summarize`.
>
> As it turns out, all the `Wilhelm` were gotten right, but over a quarter of the `SubMuli` were gotten wrong. This makes sense from the LD plot: the `Wilhelm` samples are a very compact group, but the `SubMuli` samples are all over the place.
>
> I asked for the *change* in code, so I don't need to see all your previous code again (and thus you are wasting your time by writing it out again). Tell me *what* you're adding and *where* you're adding it, for the two points. If it looks like a sensible adjustment to whatever you did in (h), I'm good with it. For example, if you went the `table` way, you could do this, borrowing an idea from the final lecture:
>
> ```
> tab %>% prop.table(margin=1)
> ```
>
> ```
> ##          pred
> ## obs          SubMuli      Upper    Wilhelm
> ##   SubMuli 0.72727273 0.09090909 0.18181818
> ##   Upper   0.05263158 0.94736842 0.00000000
> ##   Wilhelm 0.00000000 0.00000000 1.00000000
> ```
>
> and then add up the off-diagonal proportions (or take one minus the diagonal ones).

(j) (3 marks) Find an observation in Figure 25 that could be misclassified, given where it is on the plot, and justify your choice briefly. The observations are numbered according to which oil sample they are. By assessing the appropriate row of Figure 9, describe briefly whether there is doubt about the zone of the oil sample you chose, given its values on the quantitative variables. (The column `r` contains the numbers of the oil samples.)

> **My answer:** I don't mind which observation you pick, as long as there appears to be some doubt about which zone it belongs to. On Figure 25, I think observation 7 is an interesting one, because it looks as if it could be from any zone.
>
> Moving to Figure 9, the posterior probabilities of `SubMuli`, `Upper` and `Wilhelm` are 0.239, 0.281,

and 0.480 respectively. These are all reasonably high, and so this sample could reasonably be from any zone. It turns out that this one was correctly classified (the highest posterior probability actually *was* for the `Wilhelm` zone).

I'd like you to find an oil sample for which at least two of the posterior probabilities are reasonably high, or one that was actually misclassified (for which `zone` and `class` are different). Some possibilities are 7, 9, 10, 13, 18, 42, 50, 51 that have a reasonable chance of being at least two different zones. The ones that actually were misclassified were 13, 16, 18, 42, and 51. All but #16 are in the first list as well.

Of course, I can't stop you looking at this Figure first and then reverse-engineering to find an observation that stands out on Figure 25. In fact, I have no problem if you do it this way around. The point of this part is to understand the relationship between posterior probabilities and (likely) misclassification. Some kind of discussion of why you picked the oil sample you did, and what its posterior probabilities are, is what I'm after.

Question 4 (14 marks)

For people who enjoy listening to music, the loudspeaker is an important part of the listening experience, because the quality of the loudspeaker has a big impact on the quality of the sound that is produced. People who like listening to high-quality sound are called "audiophiles". A magazine for audiophiles tested 19 brands of mid-sized loudspeakers for several characteristics:

- Price: manufacturer's suggested list price, in dollars
- Accuracy: how accurately the loudspeaker can reproduce every frequency in the musical spectrum (scale of 0 to 100, higher better).
- Bass: how well the loudspeaker handles very loud bass notes (scale of 1 to 5, higher better).
- Power: the number of watts per channel needed to reproduce moderately loud music.

Our aim is to group the loudspeakers, labelled A through S, into clusters of similar ones. The data are shown in Figure 10.

(a) (3 marks) The numerical values in Figure 10 are on very different scales. Give code to create a data frame called `speakers.s` that contains only the four numerical columns, and replaces the values shown in Figure 10 with their standardized values. For full credit, do this *without naming any of the numerical columns.*

My answer:

We need to (i) get rid of the column `id` of speaker labels (not numeric) and (ii) use one of the `mutate_` extensions to standardize all the numeric columns at once. There are several ways to do this, any of which are good.

This is my favourite:

```
speakers %>% select(-id) %>% mutate_all(~scale(.)) -> speakers.s
```

I like this one best because `mutate_all` is simpler than `mutate_if` (below), so my first thought was "make all the columns numeric *first*".

Either of these is also full marks:

```
speakers %>% select(-id) %>% mutate_if(is.numeric, ~scale(.)) -> speakers.s
```

(this one is kind of dopey because the columns you have left after getting rid of `id` are all numeric anyway)

```
speakers %>% mutate_if(is.numeric, ~scale(.)) %>% select(-id) -> speakers.s
```

All of these produce:

```
speakers.s
## # A tibble: 19 x 4
##    price[,1] accuracy[,1] bass[,1] power[,1]
##        <dbl>        <dbl>    <dbl>     <dbl>
##  1   0.401          1.22    1.47       1.05
##  2   0.362          1.49    0.696     -0.700
##  3  -0.578          0.946   0.696      0.876
##  4  -1.56           0.946   0.696      0.263
##  5   0.988          0.946   0.696     -0.963
##  6   0.00927        0.129   1.47      -1.84
##  7  -2.34           0.129   1.47      -0.963
##  8   0.401          0.401   0.696      0.263
##  9   0.205          0.401  -0.0818    -0.963
## 10   0.381          0.674  -0.0818    -0.263
## 11   0.362         -0.416  -0.859     -0.263
## 12   0.753         -0.688  -0.859     -1.23
## 13   0.401          0.401  -0.0818     1.75
## 14   0.401         -1.23   -0.0818     0.263
## 15   0.401         -0.416  -0.859      0.876
## 16  -1.56          -0.960  -0.859      1.66
## 17  -0.794         -1.78   -1.64      -0.263
## 18  -0.206         -0.143  -1.64      -0.438
## 19   1.97          -2.05   -0.859      0.876
```

as required.

Does this work?

```
speakers %>% select(-id) %>% scale()
##              price   accuracy       bass      power
##  [1,]  0.400876914  1.2182424  1.4728338  1.0506751
##  [2,]  0.361716701  1.4905555  0.6955048 -0.7004501
##  [3,] -0.578128403  0.9459294  0.6955048  0.8755626
##  [4,] -1.557133721  0.9459294  0.6955048  0.2626688
##  [5,]  0.988280105  0.9459294  0.6955048 -0.9631189
##  [6,]  0.009274787  0.1289904  1.4728338 -1.8386815
##  [7,] -2.340337975  0.1289904  1.4728338 -0.9631189
##  [8,]  0.400876914  0.4013034  0.6955048  0.2626688
##  [9,]  0.205075851  0.4013034 -0.0818241 -0.9631189
## [10,]  0.381296808  0.6736164 -0.0818241 -0.2626688
## [11,]  0.361716701 -0.4156357 -0.8591530 -0.2626688
## [12,]  0.753318828 -0.6879487 -0.8591530 -1.2257877
## [13,]  0.400876914  0.4013034 -0.0818241  1.7511252
## [14,]  0.400876914 -1.2325747 -0.0818241  0.2626688
## [15,]  0.400876914 -0.4156357 -0.8591530  0.8755626
## [16,] -1.557133721 -0.9602617 -0.8591530  1.6635690
## [17,] -0.793509573 -1.7772008 -1.6364820 -0.2626688
## [18,] -0.206106383 -0.1433226 -1.6364820 -0.4377813
## [19,]  1.967285422 -2.0495138 -0.8591530  0.8755626
## attr(,"scaled:center")
##      price   accuracy       bass      power
## 579.526316  86.526316   3.105263  26.000000
## attr(,"scaled:scale")
##     price  accuracy      bass     power
## 51.072246  3.672245  1.286457 11.421228
```

Yes, this is also good, and is, you might say, easier. (It is actually a matrix rather than a data frame, but for our purposes either is good.)

It occurs to me that you can also do it this way, explicitly standardizing each numeric thing, and then save the result:

```
speakers %>% select(-id) %>%
    mutate_all(~(.-mean(.))/sd(.))
```

```
## # A tibble: 19 x 4
##       price accuracy    bass  power
##       <dbl>    <dbl>   <dbl>  <dbl>
##  1  0.401      1.22   1.47    1.05
##  2  0.362      1.49   0.696  -0.700
##  3 -0.578      0.946  0.696   0.876
##  4 -1.56       0.946  0.696   0.263
##  5  0.988      0.946  0.696  -0.963
##  6  0.00927    0.129  1.47   -1.84
##  7 -2.34       0.129  1.47   -0.963
##  8  0.401      0.401  0.696   0.263
##  9  0.205      0.401 -0.0818 -0.963
## 10  0.381      0.674 -0.0818 -0.263
## 11  0.362     -0.416 -0.859  -0.263
## 12  0.753     -0.688 -0.859  -1.23
## 13  0.401      0.401 -0.0818  1.75
## 14  0.401     -1.23  -0.0818  0.263
## 15  0.401     -0.416 -0.859   0.876
## 16 -1.56      -0.960 -0.859   1.66
## 17 -0.794     -1.78  -1.64   -0.263
## 18 -0.206     -0.143 -1.64   -0.438
## 19  1.97      -2.05  -0.859   0.876
```

("for each column, calculate it minus its mean divided by its SD". There needs to be some care with brackets on this one. This way has the advantage of not giving the columns odd names.)

Trying to standardize a non-numeric column is an error:

```
speakers %>% mutate_all(~scale(.))
```

```
## Error in colMeans(x, na.rm = TRUE): 'x' must be numeric
```

Marks: one point for getting rid of `id`, two points for scaling all the numeric columns. (Either way around is good, as long as you do them both and do them properly.)

If you don't think of any of those, then do four `mutate`s, one after each other, one for each column. It'll take you some time to write out, but it'll get the job done (which, if you are in the workplace, is what you need most), so you'll get most of the points for it.

Extra: this one does not use `mutate_at`, because the columns you want to standardize don't have *names* that have something in common. What they have in common is a *property* (being numeric), so `mutate_if`. You *could* use `mutate_at` like this, but don't forget to remove the `id` column after; the below says "standardize all the columns that are not called `id`", but the implication is that any other columns are left untouched, and if you don't want them you have to explicitly remove them:

```
speakers %>%
    mutate_at(vars(-id), ~scale(.)) %>%
    select(-id)
```

```
## # A tibble: 19 x 4
##    price[,1] accuracy[,1] bass[,1] power[,1]
##        <dbl>        <dbl>    <dbl>     <dbl>
##  1   0.401          1.22     1.47       1.05
##  2   0.362          1.49     0.696     -0.700
##  3  -0.578          0.946    0.696      0.876
##  4  -1.56           0.946    0.696      0.263
##  5   0.988          0.946    0.696     -0.963
##  6   0.00927        0.129    1.47      -1.84
##  7  -2.34           0.129    1.47      -0.963
##  8   0.401          0.401    0.696      0.263
##  9   0.205          0.401   -0.0818    -0.963
## 10   0.381          0.674   -0.0818    -0.263
## 11   0.362         -0.416   -0.859     -0.263
## 12   0.753         -0.688   -0.859     -1.23
## 13   0.401          0.401   -0.0818     1.75
## 14   0.401         -1.23    -0.0818     0.263
## 15   0.401         -0.416   -0.859      0.876
## 16  -1.56          -0.960   -0.859      1.66
## 17  -0.794         -1.78    -1.64      -0.263
## 18  -0.206         -0.143   -1.64      -0.438
## 19   1.97          -2.05    -0.859      0.876
```

(b) (2 marks) What code would produce a K-means clustering of the standardized data, obtaining four clusters, and allowing suitably for the fact that K-means is a randomized algorithm so may not produce the same result every time?

My answer: The last part was a hint about `nstart`, so one of these:

```
kmeans(speakers.s, 4, nstart=20)
## K-means clustering with 4 clusters of sizes 4, 8, 5, 2
##
## Cluster means:
##        price   accuracy       bass      power
## 1  0.1561256  0.7416947  0.6955048  0.9850079
## 2  0.1659156 -0.9602617 -0.9563192  0.1860571
## 3  0.3891289  0.7280790  0.5400391 -0.9456076
## 4 -1.9487358  0.5374599  1.0841693 -0.3502250
##
## Clustering vector:
##  [1] 1 3 1 4 3 3 4 1 3 3 2 2 1 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1]  3.552861 18.691443  4.652019  1.693797
##  (between_SS / total_SS =  60.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

or

```
speakers.s %>% kmeans(4, nstart=20)
## K-means clustering with 4 clusters of sizes 5, 8, 2, 4
##
## Cluster means:
##        price   accuracy       bass      power
## 1  0.3891289  0.7280790  0.5400391 -0.9456076
## 2  0.1659156 -0.9602617 -0.9563192  0.1860571
## 3 -1.9487358  0.5374599  1.0841693 -0.3502250
## 4  0.1561256  0.7416947  0.6955048  0.9850079
##
## Clustering vector:
##  [1] 4 1 4 3 1 1 3 4 1 1 2 2 4 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1]  4.652019 18.691443  1.693797  3.552861
##  (between_SS / total_SS =  60.3 %)
##
## Available components:
```

```
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

The first way translates more easily to the calculations for the scree plot (coming up), but either is good here. Only two marks because this is not really very difficult.

If you forgot to get rid of `id` in (a) but do so here, I have tried to go back and give you the credit in (a). As long as it's done somewhere (and you don't try to standardize `id` in (a)), it'll work.

Extra: because of the randomization, these two bits of code produce clusters numbered differently (which doesn't matter), but they do produce clusters of the same sizes, which suggests that they have the same loudspeakers in them (that is to say, any pair of loudspeakers that is in the same cluster in one will be in the same cluster in the other).

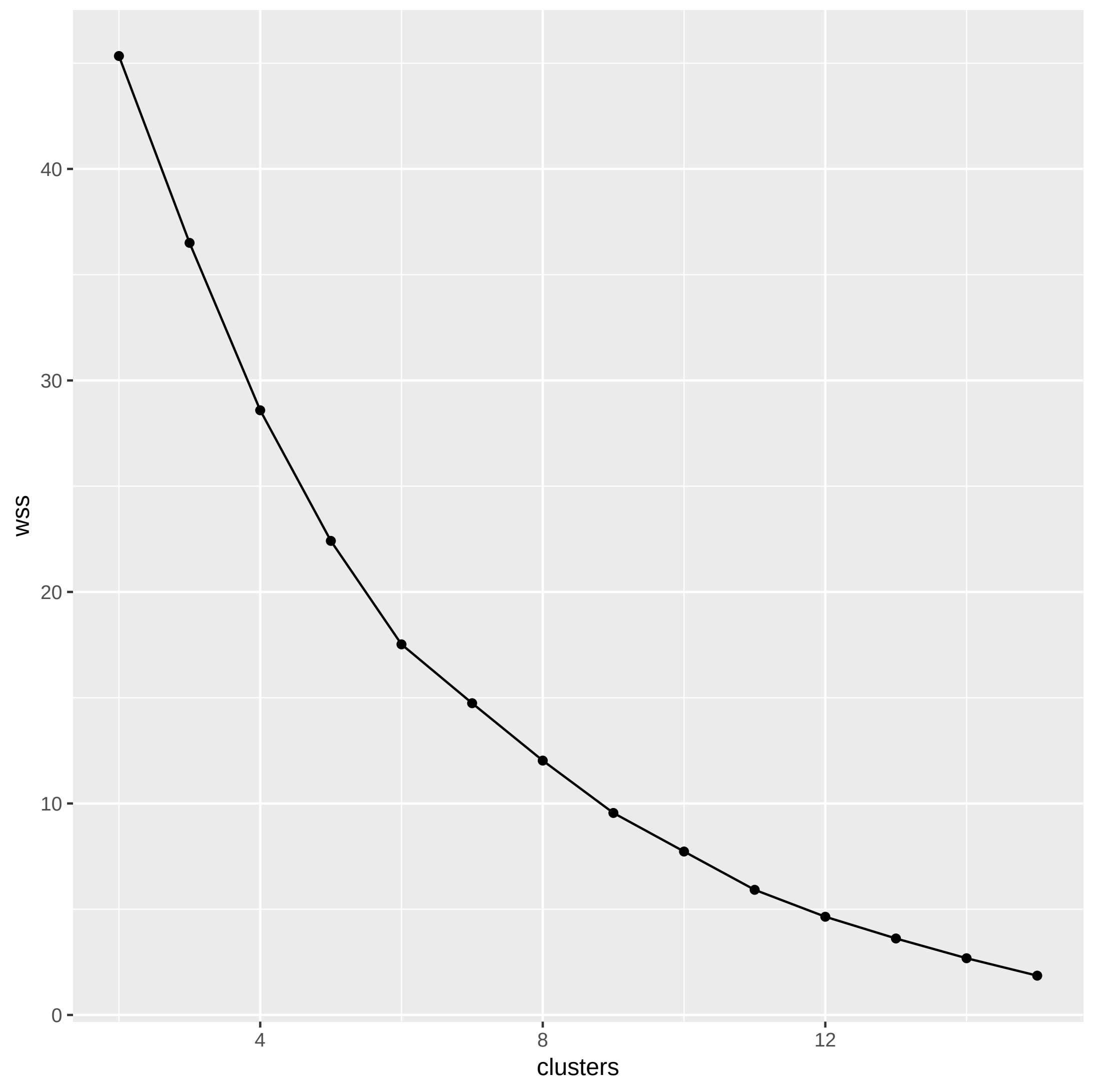
(c) (4 marks) Give code to create a scree plot, which will enable us to choose a sensible number of clusters to divide the loudspeakers into. Go up to 15 clusters.

My answer: There are several steps:

- create a data frame containing candidate numbers of clusters
- obtain the K-means clustering for each number of clusters
- obtain the total within-group sum of squares from each clustering
- save the results into a data frame
- plot the data frame, making a scatter plot with the points joined by lines.

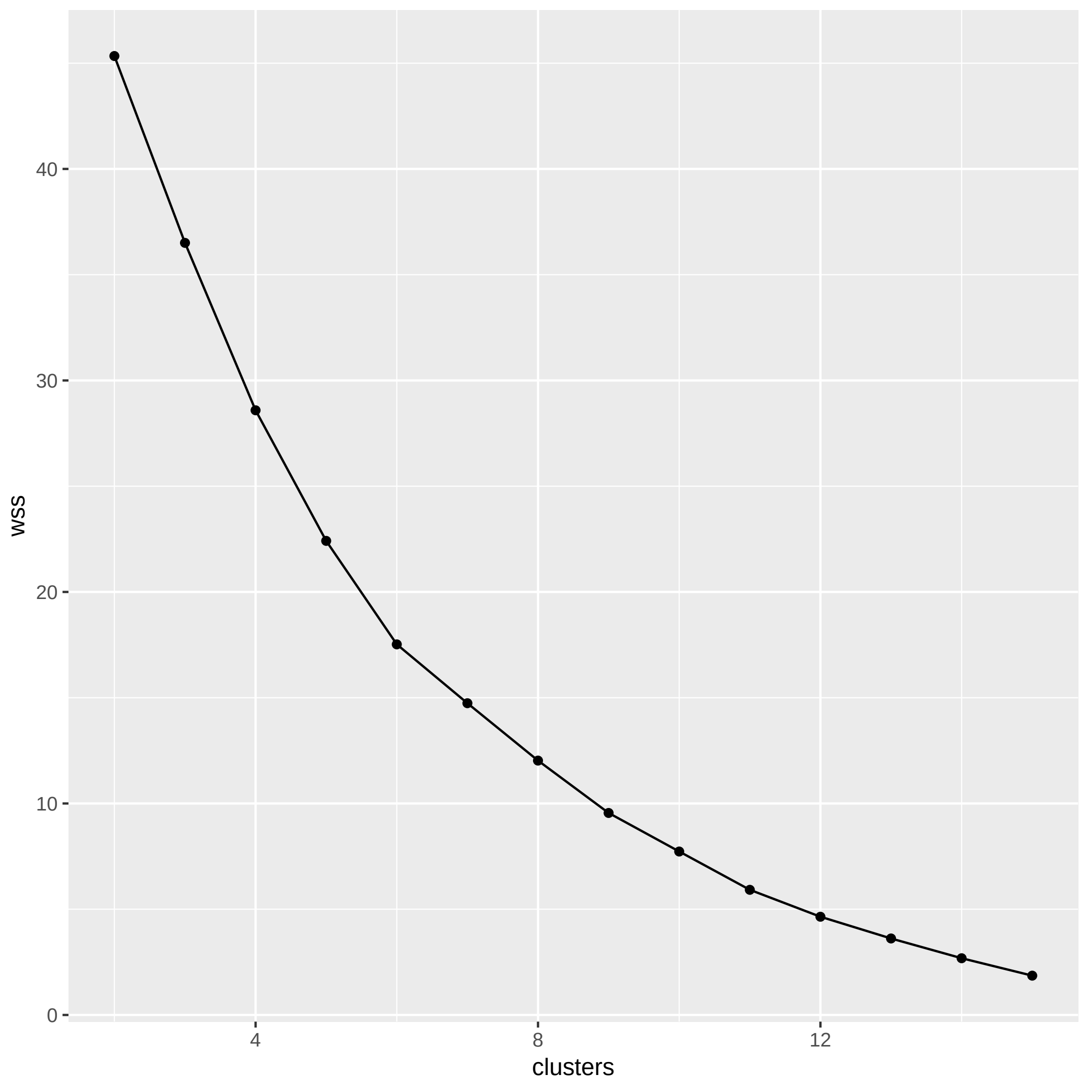
My preferred procedure is this:

```
tibble(clusters=2:15) %>%
  mutate(km=map(clusters, ~kmeans(speakers.s, ., nstart=20))) %>%
  mutate(wss=map_dbl(km, "tot.withinss")) -> scree
ggplot(scree, aes(x=clusters, y=wss)) + geom_point() + geom_line()
```



You can shorten some of the steps. For example, you can pull out *just* the total within-group sums of squares from the `kmeans`, and instead of saving the data frame, you can pipe it directly into the `ggplot`. So the shortest approach (that I can see) would look like this:

```
tibble(clusters=2:15) %>%
  mutate(wss=map_dbl(clusters,
                     ~kmeans(speakers.s, ., nstart=20)$tot.withinss)) %>%
  ggplot(aes(x=clusters, y=wss)) + geom_point() + geom_line()
```



This is the code I used to make Figure 11.

So I am looking for: (i) make the numbers of clusters (one point, rather easy); (ii) get hold of the total within-group SS for each number of clusters (two points); (iii) plot them (one point).

Doing (ii) by defining a function and then using it in the `map_dbl` is also fine (I had an example of that, probably in PASIAS). My personal preference nowadays is the "squiggle" notation within the `map`, but defining a function and passing its name in is just fine too.

I'm also willing to take a for-loop to get the total within-cluster SS; full marks for that, if you do it right (but there are more ways to go wrong).

(d) (2 marks) My scree plot is shown in Figure 11. What do you think is a sensible number of clusters?

Explain briefly.

My answer: I see an elbow at 6 clusters, so that's how many clusters I would take. There is also a little elbow at 11 clusters, but bear in mind that we only have 19 speakers altogether, and so dividing them into 11 clusters doesn't offer much insight. Even dividing them into 6 clusters may not offer much, but that's what the scree plot says.

Convince me that you have found an elbow, and propose the number of clusters you found an elbow at. One point for each of those. I can be convinced that there is an elbow at 5, maybe 4, and also at 9. Suggest something. If you suggest there should be more clusters than that, I think that's too many for 19 observations; we are trying to divide them into a "smallish" number of groups. (I could be persuaded by 11, but you'll have to be persuasive.)

(e) (3 marks) Describe a procedure by which you could obtain a graph of the results with your chosen number of clusters. I am looking for a description in words.

My answer:

- Do the K-means analysis with six (or your preferred number) of clusters;
- use those clusters as groups for a discriminant analysis;
- plot LD1 and LD2 (perhaps having first checked that using two LDs is sensible). Or make a biplot.

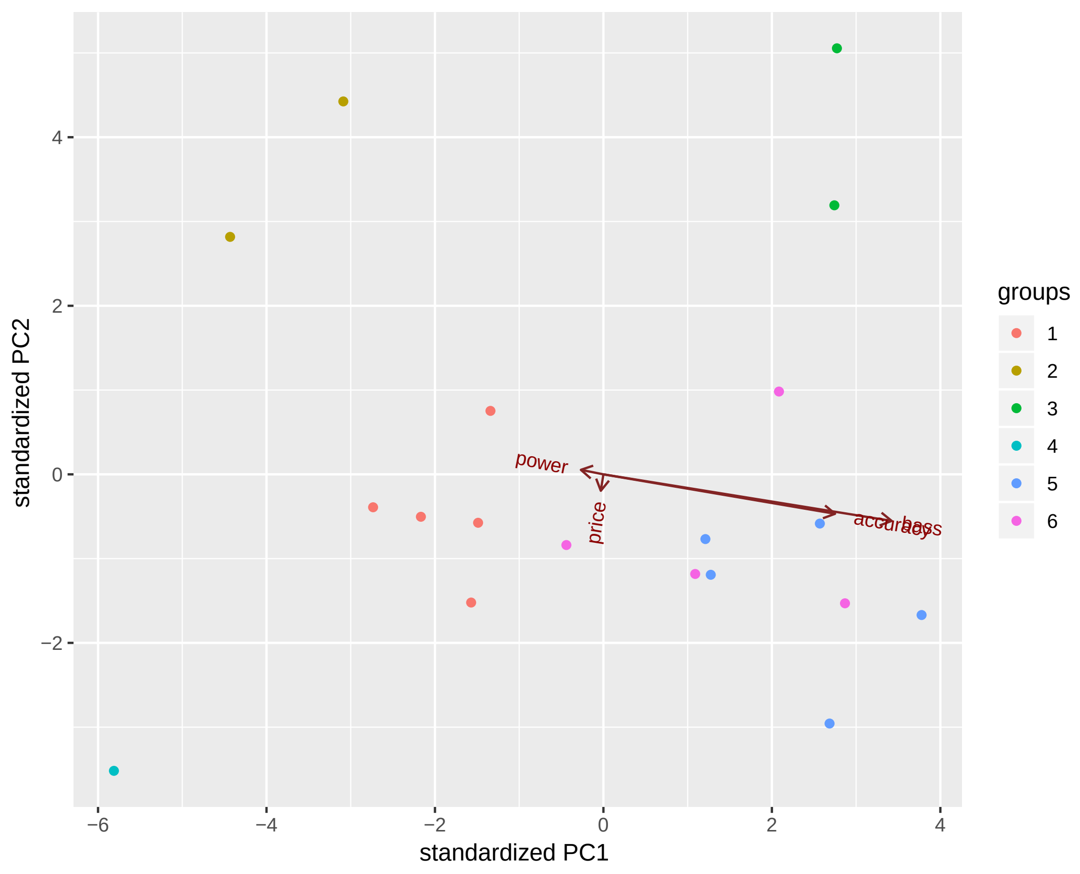
A point for each of those ideas, or something equivalent to them. Or something along the lines of the hierarchical analysis below. (I am persuadable.)

This is what I did:

```
speakers.1=kmeans(speakers.s, 6, nstart=20)
d=cbind(speakers, cluster=speakers.1$cluster)
d.1=lda(cluster~price+accuracy+bass+power, data=d)
d.1
## Call:
## lda(cluster ~ price + accuracy + bass + power, data = d)
##
## Prior probabilities of groups:
##          1          2          3          4          5          6
## 0.26315789 0.10526316 0.10526316 0.05263158 0.26315789 0.21052632
##
## Group means:
##   price accuracy bass power
## 1 597.0    84.40  2.0 24.20
## 2 519.5    81.50  1.5 34.00
## 3 480.0    88.50  4.5 22.00
## 4 680.0    79.00  2.0 36.00
## 5 599.4    89.20  3.8 15.20
## 6 587.5    89.25  4.0 37.25
```

```
##
## Coefficients of linear discriminants:
##                   LD1         LD2          LD3          LD4
## price    -0.005966786 -0.04594936  0.002872381 -0.001390057
## accuracy  0.527864393 -0.11023049 -0.081138532  0.300002517
## bass      0.656798365 -0.13018730  0.709359592 -0.879879879
## power    -0.051322757  0.01253970  0.111490993  0.025583820
##
## Proportion of trace:
##    LD1    LD2    LD3    LD4
## 0.5438 0.3608 0.0830 0.0124
```

```
ggbiplot(d.1, groups=factor(d$cluster))
```



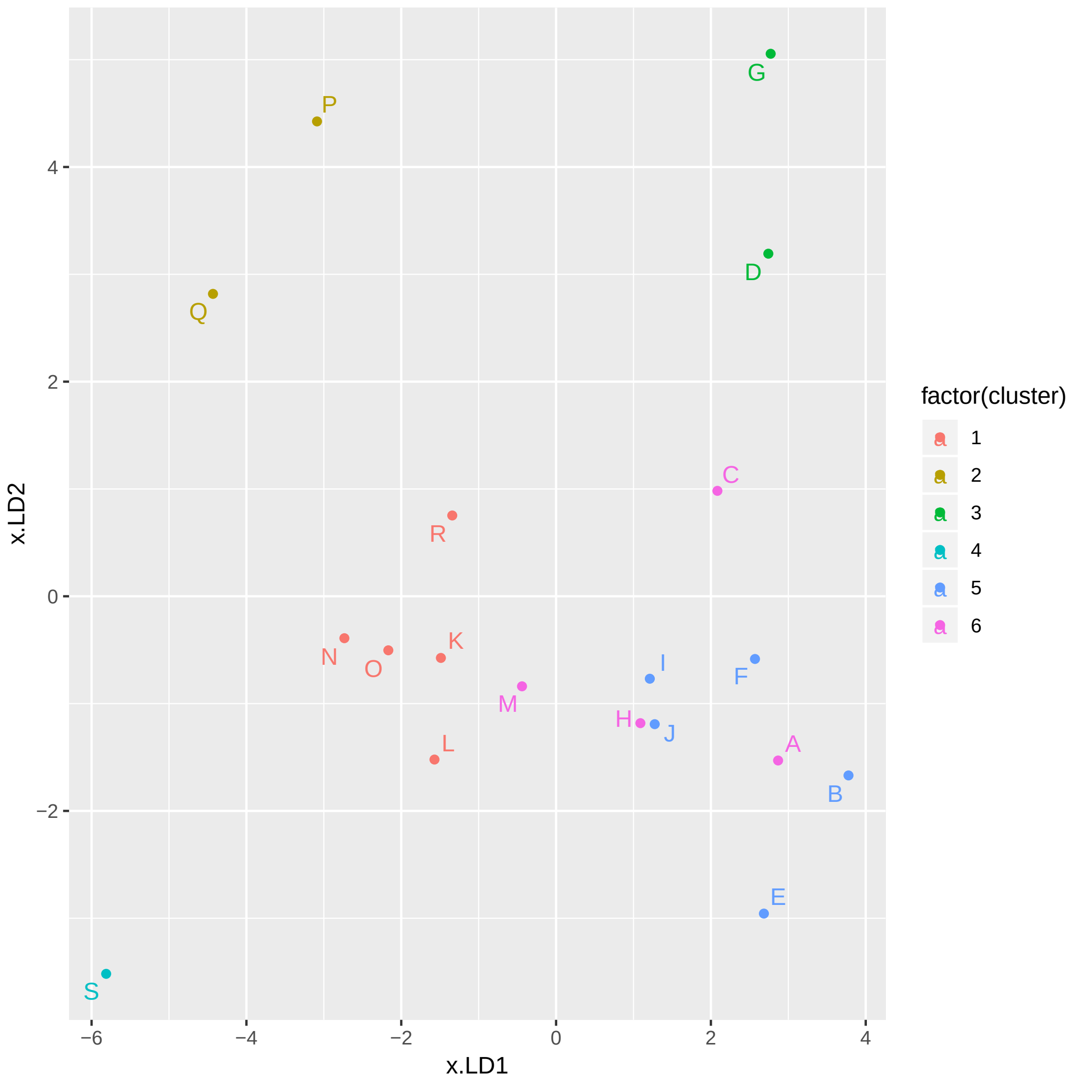
I like this visual because it shows the original variables (the ratings in each criterion for each speaker). Here, it's mainly accuracy and bass that distinguish the clusters (not power or price); these point definitely right and a little down (which is where the importance of LD2 was coming from).

Some people had the smart idea of running a MANOVA first to check that the groups really are distinct. This is smart because K-means will find as many clusters as you ask for, regardless of whether they are "really" there. However, looking at a scree plot first gives us some kind

The way you probably think of has a couple more steps (which is why I was not asking for code):

```
p=predict(d.1)
```

```
dd=cbind(d, p)
ggplot(dd, aes(x=x.LD1, y=x.LD2,
               colour=factor(cluster), label=id)) +
    geom_point()+geom_text_repel()
```

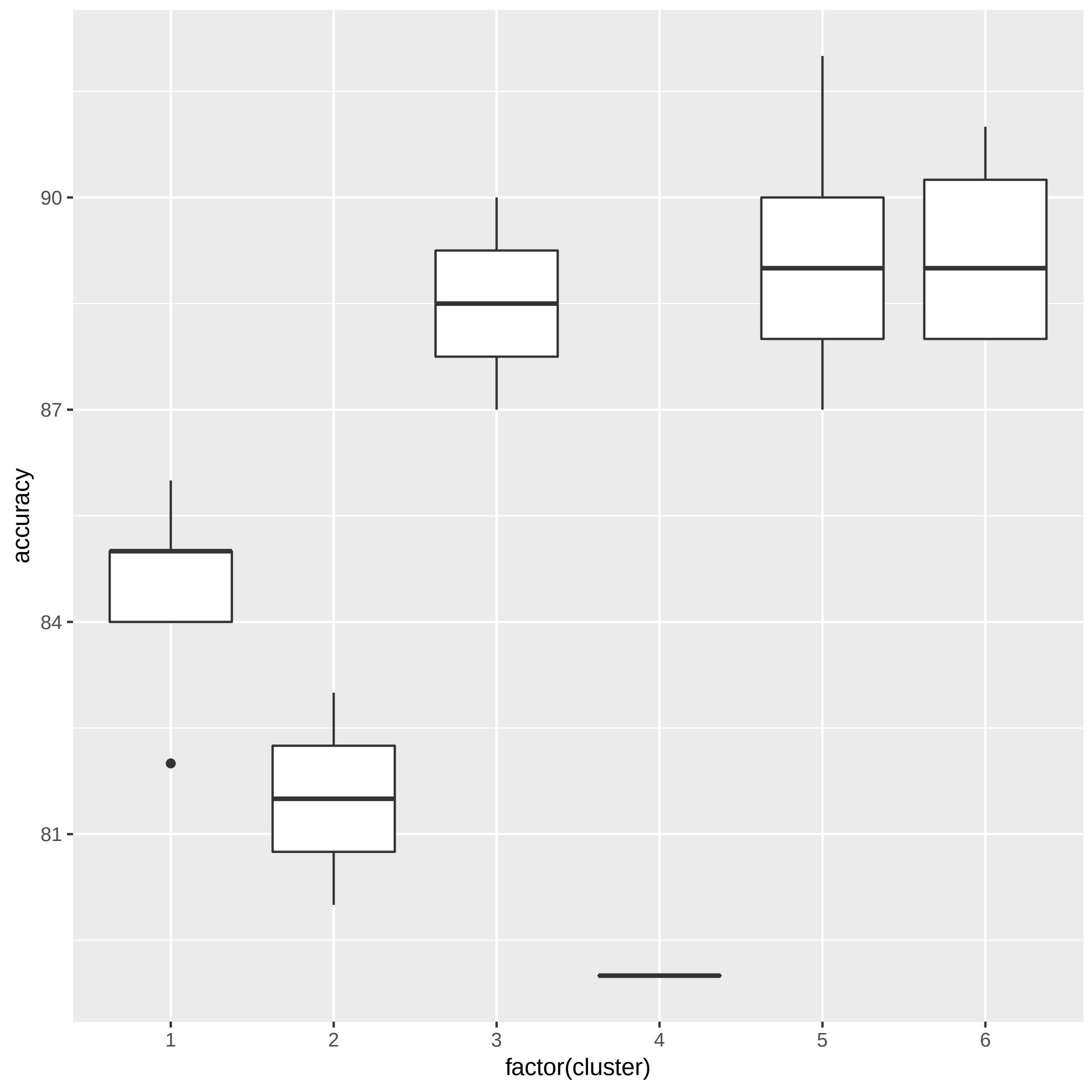


I ran out of temporary data frame names, so this one got called `dd`.

Some of the clusters are really small: cluster 4 (bottom left) has only one loudspeaker in it, and cluster 2 (top left) has only two.

Extra: The biplot said that `accuracy` and `bass` were what distinguished things left and right, eg.:

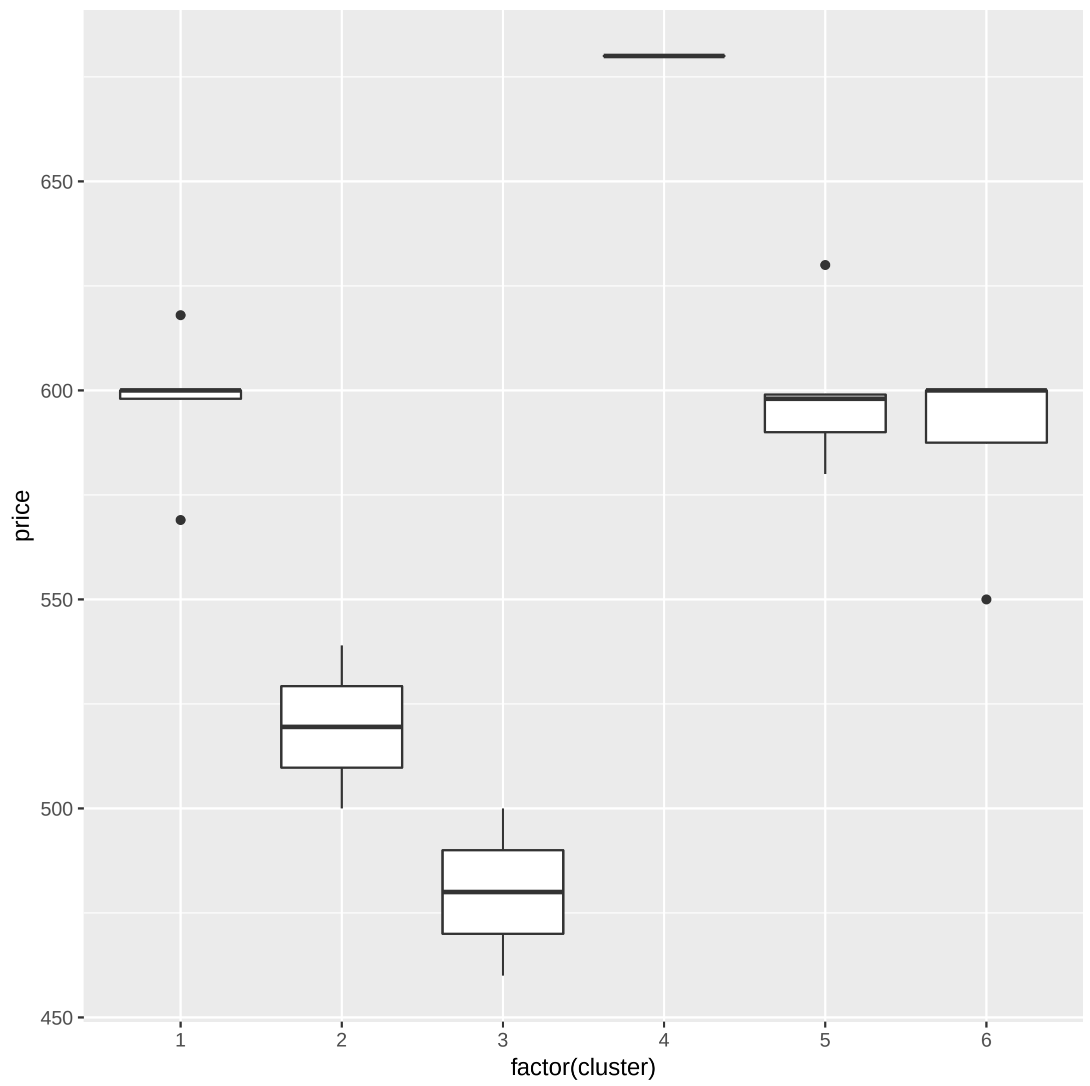
```
ggplot(dd, aes(x=factor(cluster), y=accuracy)) + geom_boxplot()
```



Clusters 3, 5, and 6 are high and the rest are low. Check.

There is definitely a distinction up and down (eg. the small clusters 2 and 4). Is that being driven by price?

```
ggplot(dd, aes(x=factor(cluster), y=price)) + geom_boxplot()
```



At least somewhat. Cluster 4 is high on price and cluster 2 is low. Cluster 3 is also low-price and also appears at the top of the LD plot. I went back to the data and discovered that what distinguishes clusters 2 and 3 is that the latter has good bass and accuracy despite the low price.

I didn't rule out hierarchical clustering as a way of getting a picture, so if you can make that happen, I'm good with that too. The picture is more obvious (a dendrogram), but getting from data to distances is less so. I think the best way is to take the standardized data and run it through `dist` (not `as.dist`, since we are now *making* distances):

```
speakers.s %>% dist() -> my_dist
attr(my_dist, "Labels")=speakers$id
```

```
my_dist
```

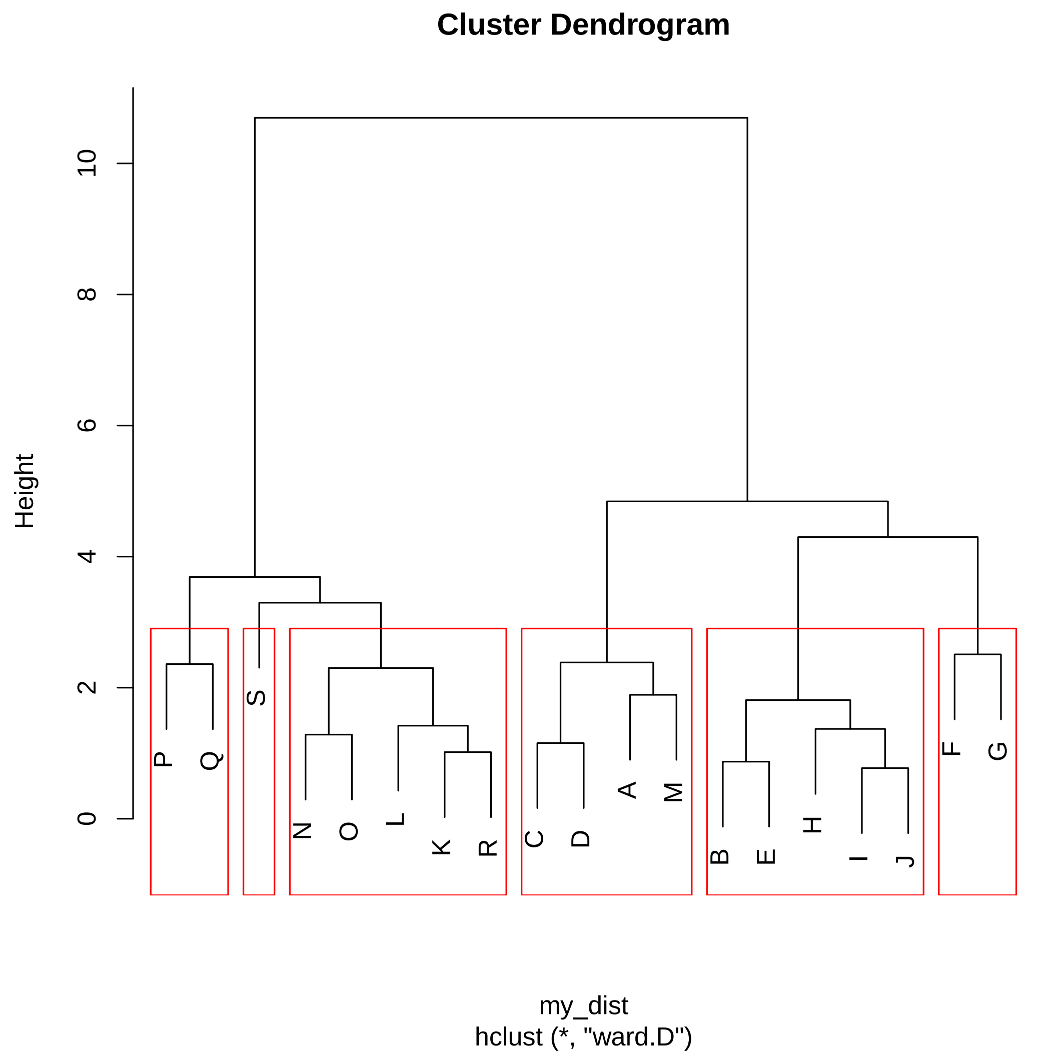
```
##           A         B         C         D         E         F         G
## B 1.9355536
## C 1.2913212 1.9140905
## D 2.2656466 2.2149949 1.1550282
## E 2.2536201 0.8707434 2.4154472 2.8251879
## F 3.1125880 1.9692348 2.9972976 2.8532329 1.7310953
## G 3.5715676 3.1349951 2.7852759 1.8405418 3.5144456 2.5074469
## H 1.3757121 1.4545108 1.2769917 2.0323443 1.4643140 2.2907247 3.1137131
## I 2.6791892 1.3726768 2.2124684 2.3470824 1.2305554 1.8155064 2.9950371
## J 2.1068541 1.2098205 1.7013119 2.1706882 1.2399410 2.3099340 3.2575305
## K 3.1359410 2.4984352 2.5396260 2.8685863 2.2702495 2.8883925 3.6778404
## L 3.7918630 2.7553899 3.3578183 3.5553076 2.2827003 2.6523183 3.9680296
## M 1.8907620 2.7932893 1.6204688 2.6363167 2.9347949 3.9409618 4.1680326
## N 3.0073942 2.9914561 2.5853841 3.0304993 2.6828564 2.9731878 3.6452209
## O 2.8527854 2.9216220 2.2867602 2.9120895 2.8278283 3.6407765 4.0779638
## P 3.7938750 4.2064283 2.7622330 2.8307326 4.4078505 4.6199596 3.7599908
## Q 4.6681855 4.2003087 3.7676996 3.7030666 4.0643579 4.0533551 4.0230234
## R 3.7557470 2.9153281 2.9134010 2.9900732 2.8856838 3.4279599 3.8174498
## S 4.3128521 4.4733341 4.2271471 4.9180088 3.9659604 4.6243885 5.6674873
##           H         I         J         K         L         M         N
## B
## C
## D
## E
## F
## G
## H
## I 1.4646275
## J 0.9771170 0.7719058
## K 1.8335386 1.3367110 1.3383175
## L 2.4378575 1.4697876 1.8772581 1.0747579
## M 1.6792090 2.7212973 2.0322165 2.3083608 3.2828244
## N 1.8093640 2.0519384 1.9773537 1.2446456 1.8001599 2.2102172
## O 1.8601047 2.1658064 1.7568906 1.1389048 2.1480323 1.4276693 1.2834596
## P 3.1728769 3.5302934 3.2774537 2.7729039 3.7095454 2.5098947 2.5445474
## Q 3.4476922 2.9411978 3.1310758 1.9474721 2.2607491 3.5559481 2.1014560
## R 2.5678472 1.7772532 1.8601230 1.0156191 1.5627887 2.8059435 2.1124607
## S 3.3545104 3.6189608 3.4395652 2.5579248 2.7826691 3.1354410 2.0250689
##           O         P         Q         R
## B
## C
## D
## E
## F
## G
## H
## I
## J
## K
## L
## M
## N
## O
## P 2.1797654
## Q 2.2760118 2.3590556
## R 1.6648410 2.7409081 1.7450686
## S 2.2634472 3.7721286 3.0977337 3.2689921
```

I said "in words", so I would like you to say something like "pass the standardized data into `dist`", or "run `dist` on the standardized data". This part is the key, so two marks for saying something about how this would be done, and one mark for saying to draw a dendrogram.

I'm never quite sure about whether `dist` works on rows or columns, so I like to check. This one came out right. The `attr` line labels the speakers by the IDs they had, rather than numbers, so that the IDs come out on the dendrogram as well.

So now:

```
my_dist.1=hclust(my_dist, method="ward.D")
plot(my_dist.1)
rect.hclust(my_dist.1, 6)
```



If you decide to go this way, the points are: compute distances between each pair of loudspeakers (using the values in `speakers.s`), then do some kind of hierarchical cluster analysis and make a dendrogram. I'm not picky about clustering method; I just like Ward.

This clustering is a bit different from K-means, so it doesn't quite answer the question in fact, but I have no problem with that (and you have no way to know whether it will or won't).

Extra: the randomness of `kmeans` kept giving me different results each time I ran this, so I *eventually* remembered to put in a `set.seed` so that things wouldn't keep changing.

Question 5 (17 marks)

How do crabs of the species *Leptograpsus variegatus* differ from one other? Body measurements were taken on 200 of these crabs, collected at Fremantle, Western Australia. For each crab the following was recorded:

- `sp`: colour, blue (B) or orange (O).
- `sex`: male (M) or female (F).
- `index`: ID number, 1 through 200.
- `FL`: frontal lobe size.
- `RW`: rear width.
- `CL`: carapace length.
- `CW`: carapace width.
- `BD`: body depth.

All the body measurements were in millimetres. A sample of the data is shown in Figure 12.

(a) (2 marks) A principal components analysis is shown in Figure 13. Why did I use `select_if(is.double)` in my code? Explain briefly.

My answer: In my principal components analysis, I wanted to include only the columns that were *decimal numbers*. The column `index` was integers, and `is.double` rather than the usual `is.numeric` would not choose `index` but would choose the other numeric columns.

One point for saying "to select the numeric columns", the second for making the point that it was the *decimal* numbers I wanted. Make sure that you clarify that you know what `double` means in this context; "to use only the doubles in the principal components" does not show that you know what "double" means.

Some people latched on to the idea that the variables we want for the principal components have two letters in their names. That's not what `is.double` selects, though. Remember that if you are selecting columns whose *names* have a certain property, it'll be a `select_at` rather than `select_if`. If I wanted to select columns with two letters in their names, I could do it like this:

```
crabs %>% select_at(vars(matches("^..$"))) %>% head()
##   sp   FL  RW   CL   CW  BD
## 1  B  8.1 6.7 16.1 19.0 7.0
## 2  B  8.8 7.7 18.1 20.8 7.4
## 3  B  9.2 7.8 19.0 22.4 7.7
## 4  B  9.6 7.9 20.1 23.1 8.2
## 5  B  9.8 8.0 20.3 23.0 8.2
## 6  B 10.8 9.0 23.0 26.5 9.8
```

The `head` on the end displays only the first few (6) rows, since this is a `data.frame` rather than a `tibble`. The thing inside the `matches` is a "regular expression"; this one says to start at the beginning of the column name, match any character, match another one, and then match the end of the name. That is, only if the column name has *exactly* two characters will it be selected.

The output displays the colour `sp` as well, which is not needed in the principal components analysis, so this cannot have been the explanation.

(b) (2 marks) A scree plot is shown in Figure 14. What do you conclude from this? Is your conclusion consistent with Figure 13? Explain briefly.

My answer: There is a big elbow at 2, so we should take one component. (One point for both bits.) This is consistent with Figure 13 because one component by itself explains nearly 96% of the variability, and the others explain almost none. (The other point.)

I would also happily take a call that 2 is still on the mountain, that there is also an elbow at 3, and therefore we should take 2 components. Referring back to Figure 13, two components explain 99% of the variability, which you can assert is "clearly better" than the 96% of one component.

If you want to take another angle: elbow at 3, therefore use 2 components, but in Figure 13 *one* component explains almost all the variability, therefore it is *inconsistent*. Also good. As usual, the quality of discussion is what counts.

Saying something like "elbow at 2" (correct) followed by "take 2 components" (wrong) and no discussion of Figure 13 is worth a rather generous one point. (One point covers a largish range, therefore.) I could have graded the question out of 3 (one for the elbow, one for going back 1, one for discussion of Fig 13), but it didn't really seem to be worth 3. As it is, I think I am being rather generous with the 2's for this question, as well as some of the 1's.

(c) (2 marks) The component loadings are shown in Figure 15. The first principal component is often a measure of "size". Do you think that has happened here? Explain briefly.

My answer: The loadings for the first component are (i) all positive and (ii) all about the same size. Thus a crab will have a large component 1 score if all its measurements are large, and a small one if all its measurements are small.

I don't think the "same size" thing is as important as "all positive". Say that, and say something about what this implies for scores on component 1.

(d) (2 marks) Which of the original variables is most associated with component 2? Explain briefly. (Note: your previous answers may have said not to look at component 2. If that's what they said, humour me and look at component 2 here anyway.)

My answer: The largest (in size) loading for component 2 is -0.898 for RW, or rear width. (You ought to give the full description "rear width" rather than, or as well as, the abbreviated variable name RW.)

This means, looking ahead, that a crab with a large score on component 2 has a *small* rear width. (You don't need to say this here, but if you at least *think* it now, you'll have an easier time later.)

I meant to ask "which *one*", but I didn't, so if you mention more than one variable (and the ones you pick are sensible), I'll go with that.

(e) (3 marks) A plot of the principal component scores on the first two components is shown in Figure 16. The crabs are labelled on the plot by the value of `index`. Find two crabs that are shown in Figure 12 and that differ substantially on component 1. By looking at Figure 12, explain how those crabs differ. (Note: `geom_text` plots text *at* the points, as opposed to `geom_text_repel`, which adds text *next to* the plotted points.)

My answer: I did this by looking for some crabs on the left of Figure 16, such as 51, 101, 1, 2, 3, 4, 52, 55, 103 and some on the right, 149, 150, 195, 200. Crab 101 (also 103 and 4 and 53) is in Figure 12, and so is 149 (and 150, 100 and 183). Two points altogether for finding a crab at each end that is on Figure 12 as well, by index, one for each end.

How do they compare? Well, you should find that the crab on the right is bigger on *everything* (that is, *all* the original variables) than the crab on the left. This is certainly true for crabs 101 and 149. One point for saying how they actually do compare. I didn't ask what you expected to see here (maybe I could have done), but if you compare to what you had in (c), you ought to get something that makes sense in the light of that.

Pick two crabs that are *different* on component 1, not similar. (If you do this, you'll probably find that they are similar on all the variables, which makes sense, but is not what I asked.)

I stated in the question that the crabs are labelled by the value of `index`, so you need to be looking in the `index` column in Figure 12. Crabs 1, 2 and 3 are not there, so you have to pick something like the crab with index 4.

I ended up going through this question three times: once to see if you had a low-end crab, once to see if you had a high-end crab, and once to see if you had a sensible comparison of the crabs you chose. This last needs to be in terms of the original variables, since we already know you have a low-scorer and a high-scorer on component 1.

(f) (3 marks) Find two crabs on Figure 16 that are also on Figure 12 and differ substantially on component *2*. By looking at Figure 12, explain how those crabs differ.

My answer: This is the same idea as the previous part, but now you are looking for crabs at the top and bottom of Figure 16. At the top I see 45, 48, 150, 141, 142, 146 and at the bottom I see 97, 98, 99, 183, 188, 200. 45 and 183 are on Figure 12. 45 is noticeably smaller on RW (and, if you like, noticeably bigger on CL and CW). Since RW has a negative loading on component 2 and the other two variables have a positive loading, we would except crab 45 to have a higher component 2 score than crab 183, and so it does.

Your answer here will depend on which two crabs you choose. What *should* happen is that RW should be noticeably smaller for the crab at the top of the graph, but other variables might differ as well, and you ought to discuss how those play into component 2 as well. I will go with any kind of sensible discussion of the two crabs you chose.

The points on the plot are kind of in a sideways-opening V-shape, so crabs that are extreme on component 2 will tend to be large on component 1 as well. There is no problem in using the same crab as one of your examples in both of the last two parts.

(g) (3 marks) Another plot of component scores is shown in Figure 26. What do you conclude from this plot, and how does that relate to the original variables that were measured? Explain briefly.

My answer: This is a plot of component scores with the sexes distinguished by colour. The males are mostly at the top of this plot, so they tend to have a higher score on component 2 than the females. One point.

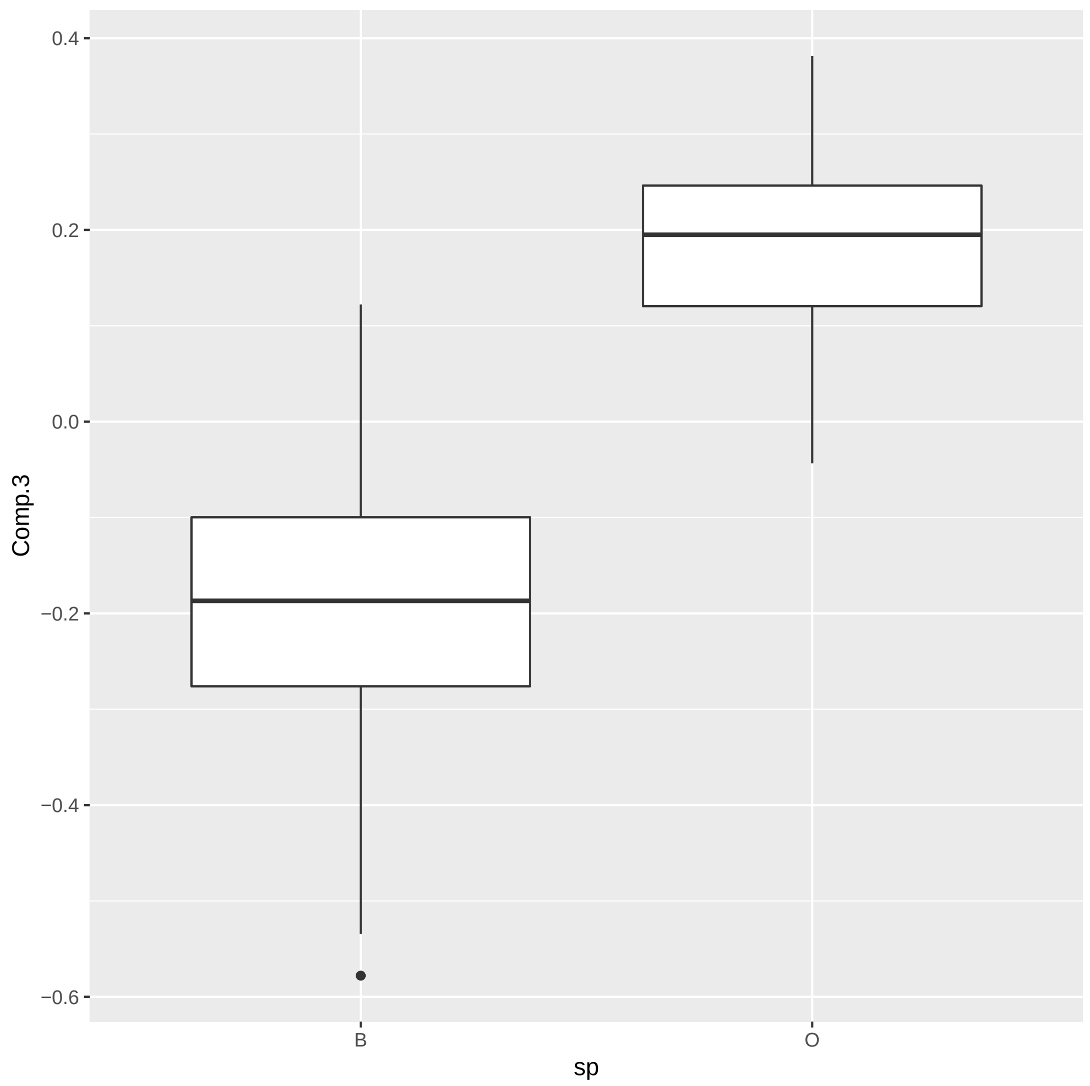
Then relate this back to the original variables. Component 2 depended mostly (and negatively) on rear width, so a crab that had a large score on component 2 had a *small* rear width. The second point. This means that the principal difference between males and females is that females have a *larger* rear width than males do. The third point.

If you like, you can say that the clearest distinction between males and females is for the large crabs, meaning that the small crabs don't differ in rear width quite so much (in an absolute sense; it may be that the *relative* difference, say a percent difference, is similar).

Component 1 doesn't distinguish the sexes, so there is no overall difference in size between males and females (which you might find surprising).

Extra 1: the third component, even though it has tiny importance by itself, turns out to distinguishes the colours:

```
ggplot(d, aes(x=sp, y=Comp.3)) + geom_boxplot()
```



The orange crabs had a higher score on component 3, which means (looking back at the loadings) that they are bigger on frontal lobe and body depth, and smaller on carapace width, than the blue crabs.

Extra 2: an unusual feature of this data set is that components 2 and 3, even though they explained little of the variability, had clear meanings in that they distinguished some other features of the crabs. I think this happened because the *dominant* thing about how the crabs differed was their size (component 1 was much more important than anything else), but after you get that out of the way, there is still something else to see. The data set is actually one of the ones in the MASS package; in the principal components example where I saw it used, the point was made that just because a component doesn't explain much of the variability doesn't mean it's not worth looking at. (Sorry about the double negative: components that explain little variability can still be interesting.)

Extra 3: we might have gone after the question "what distinguishes males and females" from the beginning (instead of seeing it as a by-product of the principal components analysis). This is "what distinguishes known groups", so it would be a discriminant analysis:

```
crabs.2=lda(factor(sex)~FL+RW+CL+CW+BD, data=crabs)
crabs.2
## Call:
## lda(factor(sex) ~ FL + RW + CL + CW + BD, data = crabs)
##
## Prior probabilities of groups:
##   F   M
## 0.5 0.5
##
## Group means:
##       FL     RW     CL     CW     BD
## F 15.432 13.487 31.360 35.830 13.724
## M 15.734 11.990 32.851 36.999 14.337
##
## Coefficients of linear discriminants:
##            LD1
## FL -0.17509926
## RW -1.61455655
## CL  0.90033985
## CW -0.27294518
## BD  0.08261892
```

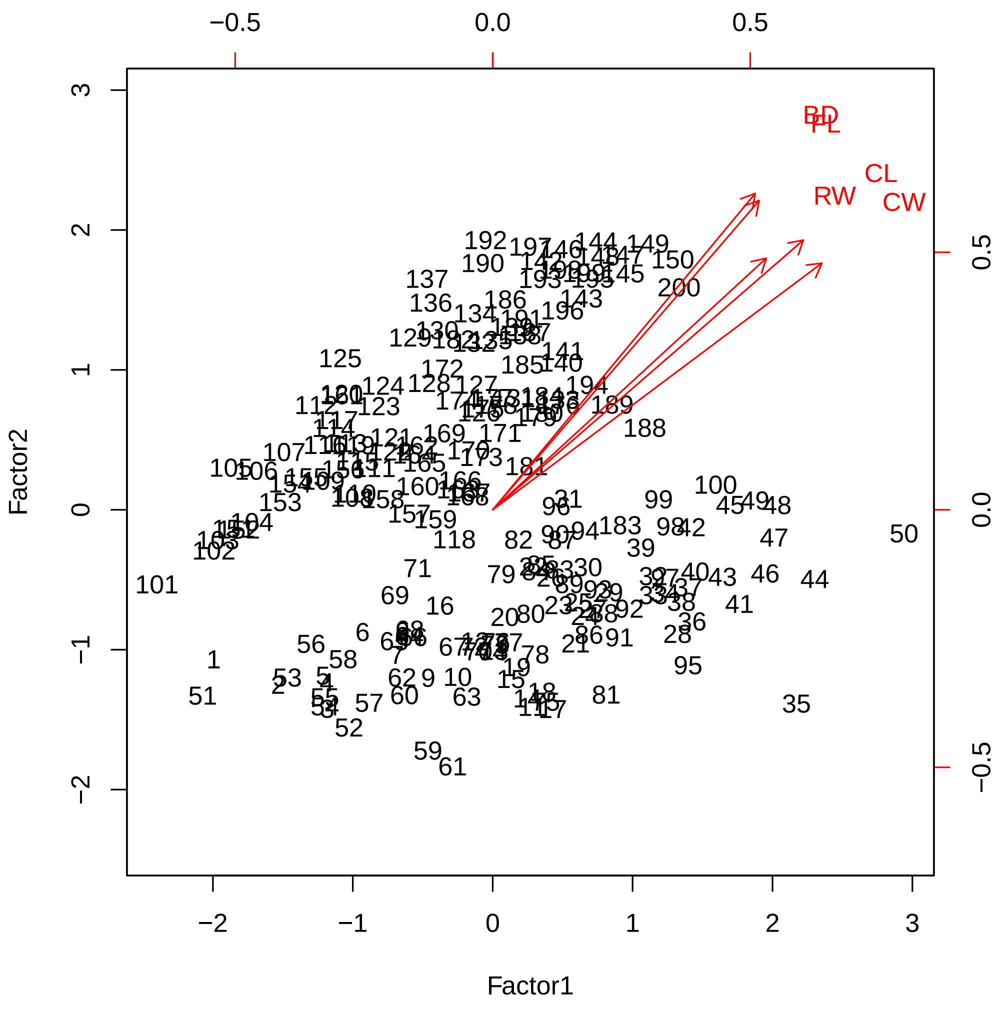
This says that females have a larger rear width (and to a lesser extent, smaller carapace length) than males do, so they will be smaller on LD1 than males are.

Extra 4 (yes, I know): I intended to make this a factor analysis question as well, but it turns out that the factor analysis doesn't clarify things all that much:

```
crabs %>% select_if(is.double) %>% factanal(2, scores="r") -> crabs.f
crabs.f$loadings
##
## Loadings:
##    Factor1 Factor2
## FL 0.647   0.750
## RW 0.664   0.611
## CL 0.754   0.655
## CW 0.799   0.598
## BD 0.637   0.768
##
##                Factor1 Factor2
## SS loadings      2.471   2.312
## Proportion Var   0.494   0.462
## Cumulative Var   0.494   0.956
```

In fact, to my mind it makes things *less* clear: pretty much everything is in both factors. The biplot is definitely not the usual thing from a factor analysis:

```
biplot(crabs.f$scores, crabs.f$loadings, xlabs=crabs$index)
```



You would have expected the arrows to point either up or across, but they don't (because everything is in both factors). I think the story is that, as far as the factor analysis is concerned, everything is "size".

I wondered whether three factors helps, but it turns out that you can't get three factors out of five variables:

```
crabs %>% select_if(is.double) %>% factanal(3, scores="r")
## Error in factanal(., 3, scores = "r"): 3 factors are too many for 5 variables
```

So I gave up on that idea.

Question 6 (9 marks)

Basketball fans often believe in the "hot hand": a player who has just successfully made a shot is more likely to make the next one as well. A study was made of this in the early 1980s. The study used free throws (also known as "foul shots"), because a free throw is always taken from the same place, and opposing players are not allowed to interfere with the shot.

A player that is fouled while shooting (and also at certain other times during the game) is awarded two free throws. Is a player who makes their first free throw, when they are awarded two, more likely to make the second one? The data we use comes from the Boston Celtics, 1980–1982; this is the data that was used for the original "hot hand" study, and is shown in Figure 17. The data is shown in "long" format, with one column of frequencies: the number of times the player shown "hit" (made) or missed the first free throw, and hit or missed the second one. For example, Larry Bird missed his first free throw and made the second one 48 times during the period the data were collected.

(Basketball fans among you will note that I have simplified things a little: a player who is fouled while shooting and *makes the shot anyway* only gets one free throw. Such single shots are not counted here.)

(a) (2 marks) The first analysis totalled up over all players. This is shown in Figure 18. The first part of the output shows that when the first shot was hit, the second shot was hit 79% of the time; when the first shot was missed, the second shot was hit 74% of the time. The bottom part of the output shows that this small difference is significant; there is a (small) association between hitting or missing the first shot and hitting or missing the second one. (The test is an ordinary chi-squared test for independence.)

Figure 19 shows the proportion of *second* shots hit by each player according to whether they hit or missed the first one. Using Figure 19, criticize the analysis in Figure 18.

My answer: The most obvious answer is that the players differ in ability to hit free shots, and so combining over all the different players is a mistake. For example, Larry Bird and Tiny Archibald hit over 80% of all their free shots, and Rick Robey hit only about 60%.

I don't think it's so insightful to describe the two columns in Fig 19 as "about the same", because it could be that the first column is consistently slightly higher than the second (which would lead to the same kind of result in Fig 18). What matters here is the *inconsistency*, an idea expanded on in the next paragraph.

Another way of saying it is that some of the players in Figure 19 have more success on the second shot if they hit the first one (Cedric Maxwell, Kevin McHale) and some of them are more successful on the second shot if they *miss* the first one (Chris Ford, ML Carr). If the "hot hand" is real, there should be a consistent direction: most of the players should have greater success on the second shot if they hit the first one.

Yet another way to say it is that what we have here is really repeated measures (multiple attempts for each player), and the analysis in Figure 18 ignores this. From this point of view, there are two sources of variability: from shot to shot, the same player will hit some and miss some, and also from one player to another (the players vary in their general ability to hit free shots).

Some people envisaged that there was some kind of Simpson's paradox happening here, which I talk about below.

(b) (2 marks) A log-linear analysis is shown in Figures 20 through 22. Why is Figure 22 a good place to stop? Explain briefly.

My answer: The usual thing: everything that could be taken out is significant, and so removing anything would be a mistake.

(c) (2 marks) What do the effects remaining in Figure 22 tell you, in the context of the data?

My answer: There are two remaining associations:

- between `Player` and `first_shot`
- between `Player` and `second_shot`

This says that whether or not the first shot is successful depends on the player taking it, and whether or not the second shot is successful depends on the player taking it. These two things are independent of each other. This is what was (strongly) suggested by Figure 19: the players differed quite a bit in terms of their ability to hit free throws, and the log-linear analysis is saying that this player difference is real, not just chance.

(d) (3 marks) What does the log-linear analysis in Figures 20 through 22 tell you about the evidence for a hot-hand effect? How is that consistent with Figure 18? Explain briefly.

My answer: The hot-hand effect would show up as an association (interaction) between `first_shot` and `second_shot`: knowing about the outcome of the first shot would tell you something about the second one. But this interaction term was removed in Figure 22 on account of not being significant in Figure 21. That is to say, there is *no* evidence of a hot-hand effect from the log-linear analysis.

This is apparently inconsistent with Figure 18, which I asked you to criticize earlier. The earlier analysis totalled up over all players, which we said was a mistake, something confirmed by the log-linear analysis, which said that you need to keep `Player` terms in the model (because the players differ in their ability to hit free throws). The bottom line is that there appears to be a hot hand effect when you total up over all players, but that *disappears* when you do the correct log-linear analysis that accounts for differences among players. It's kind of a Simpson's paradox, in that aggregating or not aggregating over players gives you a different conclusion, and then you have to ask yourself whether aggregating over the (very different) players was a smart thing to do. This is like the airlines example in class, where if you do it one airport at a time, you get a different answer from summing up over all the (very different) airports. From a model-building point of view (that is, as in C32), this is that business of keeping variables in the model that might make a difference, so that you can get better conclusions for the things you care about (the hot hand, in this case).

Another way of looking at it is this: if the first shot is missed, it is more likely to be a weak shooter who took it, and therefore it is more likely that the second shot will be missed *because the same shooter is taking that one as well.* This is how there may *appear* to be a hot-hand effect, but the analysis says that the only reason there may appear to be a connection between the results of the two shots is because of the identity of the player taking them.

I think a three-point answer recognizes the connection with the `first_shot:second_shot` interaction, notes its non-significance (or that it was removed during the modelling process), and expresses that the better analysis says that there is *no* hot hand. Less than three with things missing as I see it, according to how close I thought you got to a three-point answer. Not to be confused with a three-point basket.

I got the idea for this question from this paper: Vokey, J. R. (2003). Multiway frequency analysis for experimental psychologists. Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale, 57(3), 257. The author advocates doing log-linear analyses rather than chi-squared testing that adds up over the things you don't care about: that is, what we did here.

I had forgotten that this was rather a famous data set.