University of Toronto Scarborough
Department of Computer and Mathematical Sciences
STAC32 (K. Butler), Final Exam
December 7, 2017 9:00-12:00

IT IS ASSUMED THAT YOU HAVE READ THE BOX BELOW.

Aids allowed:
- My lecture slides
- Any notes that you have taken in this course
- Your assignments and feedback on them
- My assignment solutions
- The course R text
- The course SAS text
- Non-programmable, non-communicating calculator

Past exams are *not* allowed.

Before you begin, complete the signature sheet, but sign it only when the invigilator collects it. The signature sheet shows that you were present at the exam.

This exam has 72 numbered pages of questions. Please 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). If you need more space, use the backs of the pages, but be sure to draw the marker's attention to where the rest of the answer may be found.

The maximum marks available for each part of each question are shown next to the question part. In addition, the total marks available for each *page* are shown at the bottom of the page, and also in the table on the next page.

When giving SAS code, you can provide code that runs either on the online version of SAS Studio, or on the version that runs on a virtual machine. Either version is acceptable.

Code for R graphs should be in `ggplot` style, as in lecture. There may be partial credit for "base" graphs.

For any questions below involving R code, you may assume that this code has already been run:

```
library(tidyverse)
```

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.

Last name: _____

First name: _____

Student number: _____

For marker's use only:

Page	Points	Score
1	7	
3	3	
4	3	
5	4	
8	3	
10	2	
11	4	
15	2	
20	3	
21	3	
24	4	
25	6	
27	6	
28	2	
29	2	
31	2	
34	2	
36	2	
39	5	
43	3	
47	2	

1. Four weight loss programs are being compared. These are: (i) a low calorie diet, (ii) a low fat diet, (iii) a low carbohydrate diet, (iv) a placebo diet (the participants in this group are told that they are "participating in a study of healthy behaviour"). Twenty participants are randomly allocated to one of these four programs. Each participant's weight is measured at the start, and again after 8 weeks. The response variable is the weight loss, measured in pounds (the weight at the start minus the weight at the end). A larger weight loss indicates that the diet is more helpful. A negative weight loss indicates that the diet actually resulted in a *gain* of weight.

   The data are shown in Figure 1. We will be analyzing the data in R. The diet names are written as single words.

   (a) (2 marks) Explain briefly what feature the data has that will allow it to be read in with `read_delim`.

Solution: The data values are separated by *exactly one* space. Two marks for this, one mark for "separated by spaces", since if the values were separated by more than one space, `read_delim` would not work.

For the second mark, you have to say or imply that there is only one space between each data value. "Separated by a space" is two marks, "separated by space" is one (since the data values could then be separated by any number of spaces). It is important that your use of English is good enough to make that distinction. Likewise, "space-delimited" is only one mark, because this doesn't rule out having a variable number of spaces between the data values (which would prevent `read_delim` from working).

   (b) (2 marks) Why would analysis of variance be a plausible technique to use to analyze these data? Explain briefly.

Solution: We are comparing four different diets (to see whether the mean weight loss differs).

A minimal acceptable answer is "we are comparing more than two groups of independent observations". I saw that wording a lot, which suggests that it came from one of your other courses. I would like some additional words that say what the groups are (the diets) and why the observations are independent (each person only does one diet and thus produces only one weight loss), but, this being a final exam, I would accept the minimal acceptable answer. (On a midterm, I would ask for more precision.) I changed my mind on this. I originally put in my solutions:

> One point for something less specific, like "we are comparing four different groups" without naming what the groups are.

But I decided that if you could get the wording correct, I was happy. If you messed up the wording, you probably got only one.

The point here is that there are more than two diets (otherwise we could have used something like a two-sample t-test), and that each person only does one of the diets (otherwise we'd be using something like repeated measures). I was willing to cut you more slack if you talked about mean weight loss of diets, since at least then you showed that you knew what we were trying to do.

The word "plausible" is the right one here. It means something like "possibly acceptable" without saying that ANOVA is "definitely acceptable". Here, the experimental design is right for ANOVA, but the actual data might not be (for example, approximate normality might be no good, in which case we'd be doing something like Mood's median test later). That means that talking about normality, equal variances etc. here is not relevant: it is the experimental design that makes ANOVA possibly the right thing to do, so it is the experimental design that you need to talk about.

   (c) (3 marks) Explain briefly how the data in Figure 1 are not currently in the correct format for an analysis of variance. In answering this question, you can describe what the correct format would

be, and how the format shown in the Figure differs from that.

Solution: The data values in the four different columns are all weight losses. We need to have *one* column of weight loss values, with a second column containing the diet that the person in question was on.

The key thing is that the weight losses are currently in different columns, but they should all be in one column. The last point is for saying that there should be a column of diets.

It was pretty easy to get three marks here. Perhaps the easiest way was to draw something like my data frame `diets2` below; this is an easy way to show that you know what's going on.

The issue here is of untidy vs. tidy data, which you can say if you like, *but* copying verbatim from your notes the definition of tidy data is worth *nothing* unless you can translate that into something that shows an understanding of what needs to be done *for these data*. This course is only partly about knowledge; it's mainly about *thinking*, of how to apply your knowledge in different situations. What I want to see is how tidy data looks *for this data set*.

(d) (3 marks) The data in Figure 1 have been read into a data frame called `diets`. Using this data frame, give R code that will create a new data frame called `diets2` that contains the data in the appropriate format for the analysis of variance.

Solution: I didn't ask you earlier for code to read in the data, since the question about `read_delim` pretty much gave it away, so I'd better do that first. No points if you supply this:

```
diets=read_delim("diet.txt"," ")
```

```
## Parsed with column specification:
## cols(
##  LowCalorie = col_integer(),
##  LowFat = col_integer(),
##  LowCarbo = col_integer(),
##  Control = col_integer()
## )
```

```
diets
```

```
## # A tibble: 5 x 4
##   LowCalorie LowFat LowCarbo Control
##        <int>  <int>    <int>   <int>
## 1          8      2        3       2
## 2          9      4        5       2
## 3          6      3        4      -1
## 4          7      5        2       0
## 5          3      1        3       3
```

Now, your bit. This is a standard application of `gather`:

```
diets2 = diets %>% gather(diet,weightloss,LowCalorie:Control)
diets2
```

```
## # A tibble: 20 x 2
##          diet weightloss
##         <chr>      <int>
##  1 LowCalorie          8
##  2 LowCalorie          9
##  3 LowCalorie          6
##  4 LowCalorie          7
##  5 LowCalorie          3
##  6     LowFat          2
##  7     LowFat          4
##  8     LowFat          3
##  9     LowFat          5
## 10     LowFat          1
## 11   LowCarbo          3
## 12   LowCarbo          5
## 13   LowCarbo          4
## 14   LowCarbo          2
## 15   LowCarbo          3
## 16    Control          2
## 17    Control          2
## 18    Control         -1
## 19    Control          0
## 20    Control          3
```

What makes the columns different (different diets), what makes them the same (they're all weight losses), and the columns to gather (all of them). Call the new columns what you like. As long as they look like a "diet" or a "program" followed by a "weight loss", I'm good with them. (If you call them something like `x` and `y`, you'd better explain somewhere what these represent.)

Grading: 3 points for the above, or for doing it without a pipe but with `diets` as the first thing inside the `gather`, or even for something like this:

```
diets %>% gather(diet,weightloss,LowCalorie:Control) -> diets2
```

This does the assignment ("saving") at the end, so whatever comes out of the pipe goes into `diets2`.

2 points for getting it basically right, but messing something up, like getting the diet and weightloss variables the wrong way around. I tried to give 2 for any almost-correct `gather`, though having `gather` wasn't quite enough to guarantee you 2 (see below).

1 point covered a largish range, from something like a `gather` with the `diets` columns listed in it but nothing else down to pretty much anything with a pipe symbol in it that looked at all plausible.

No points for a `read_delim` (I had previously given that away) or an `aov` (next part, though if you get it correct here that's OK for the next part).

This was pretty quick marking, since most people got it basically right or basically wrong, and I didn't have many judgement calls to make (these are the things that take the time when marking).

(e) (3 marks) Using your new data frame `diets2`, give R code to run an analysis of variance comparing the diets, and to display the F-test from the analysis of variance.

Solution: With the column names I chose (use yours), mine goes like this:

```
weightloss.1=aov(weightloss~diet,data=diets2)
summary(weightloss.1)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## diet         3  75.75   25.25   8.559 0.00128 **
## Residuals   16  47.20    2.95
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This is the code that produced Figure 2. Use your own column names, and give the output any name you like. I'm using my preferred response variable plus number, but I don't insist on that.

If you couldn't do the previous part, *explain what you needed to get from it*, and then give the code to produce the ANOVA. You can get full marks for this part that way.

As we saw in the lecture on regression with categorical variables, this also works (and is thus worth full marks):

```
weightloss.1a=lm(weightloss~diet,data=diets2)
anova(weightloss.1a)
```

```
## Analysis of Variance Table
##
## Response: weightloss
##           Df Sum Sq Mean Sq F value   Pr(>F)
## diet       3  75.75   25.25  8.5593 0.001278 **
## Residuals 16  47.20    2.95
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results are identical apart from rounding.

The basic grading was 2 points for getting the `aov` correct, and one for the `summary`. You don't, indeed shouldn't, do anything fancy to get the F-test, since that's exactly what `summary` does. `oneway.test` does a Welch test, *which you should justify doing*, since that's not what I asked for (though there is some credit for this), and `var.test` tests a sample variance against a null value, which is just plain wrong. Note that the order of variables in `aov` is the *opposite* way around from the `gather`, not surprising since they are doing different things. I didn't ask for a Tukey, so it is strictly speaking *wrong* if you put one in, though this time I didn't penalize you for doing so (I am somewhat more relaxed about marking finals than midterms: on a midterm I might have deducted one).

Last, `with` is equally good:

```
weightloss.1c=with(diets2,aov(weightloss~diet))
summary(weightloss.1c)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## diet         3  75.75   25.25   8.559 0.00128 **
## Residuals   16  47.20    2.95
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

My preference is to go with `data=` if it works, and `with` if `data=` doesn't, but that's your call.

(f) (4 marks) The ANOVA and Tukey analyses for these data are shown in Figures 2 and 3. Give a

complete conclusion based on these two Figures, in the context of the data. In your conclusion, explain briefly whether or not you need to refer to Figure 3, and if you do, why.

Solution:

I'm leaving this one relatively unstructured, to see how you can handle it. What you need to do is refer to Figure 2, say what its significance means, then say that because there are significant differences to find, we need to look at Figure 3, and then interpret that. Thus:

- The P-value of the F-test in Figure 2 is 0.00128, which is less than 0.05. We reject the null hypothesis (that all the diets have the same mean weight loss) and conclude that the mean weight losses are not all the same (or that at least one of them is different). Note the use of "and conclude that" to make it clear that the end of my sentence is a conclusion, not a hypothesis.
- We have determined that some diets differ (in terms of mean weight loss). In order to figure out which ones, we need to look at the Tukey analysis in Figure 3.
- Three of the differences, all involving the Low Calorie diet, are significant, but the other three are not. By looking at the table of means at the bottom of Figure 3, this is because:
  - the Low Calorie diet has a significantly *higher* mean weight loss than all the other diets (at $\alpha = 0.05$).
  - The other diets do not differ significantly from each other.

  (I am looking for both of those last two points. If you used $\alpha = 0.01$ your conclusions will be different.)

Thus, the four marks are the first two major bullet points and the last two subpoints.

I put the table of means below the Tukey output to make it easier for you to figure out what was going on. You can figure out that the mean weight loss for the Low Calorie diet is higher than for all the other diets, just using the Tukey output, but it's easier to see if you have the means as well. (This last is done via `group_by` and `summarize`, as you might have guessed.) Low Calorie's mean is at least 3 higher than any of the others, which is large enough to be significant; even though Control seems clearly the lowest, it doesn't differ from Low Carbo or Low Fat by enough to be significant (it only differs from them by about 2).

You did notice that the captions are *below* the Figures they refer to, didn't you? I said that on the front page of the Code and Output, since this page was potentially confusing.

As for marking: I am hoping to see things in the order I listed them above, but if you have drawn a valid conclusion from Figure 2 somewhere, I'll live with that. (A minimal answer for the first part would be something like "we need to look at Figure 3 because the test in Figure 2 is significant, indicating that there are some differences between diets to find". That catches both the first two points.)

As I write this, I have been through the pile looking for answers to my first bullet point, and haven't allotted any marks yet. But my aim is to reserve 4 points out of 4 for those answers that show proper step-by-step reasoning, and give 3 for answers that get to the right place without appropriate justification. I'll see how that looks when I go through the pile again, though.

Coming back to this from the viewpoint of seeing everyone's answers: the major thing in the Tukey output was to make sure that you used those P-values. It's one thing to say that Low Calorie is best and Control is worst by looking at their means, but in fact Control is *not* significantly worse than Low Carbo or Low Fat (look at the P-values), so it is only apparently worse than them by chance. Said differently, if you were to repeat this study, Low Calorie would most likely come out best again, but the other three could come out in any order.

I was wishing that I had shown you a rather old-fashioned diagram that us grizzled veterans used to draw (by hand) to summarize the ANOVA (sums of squares calculated by hand) and the Tukey (using a table called the Studentized Range). What you do is to list the means in order across the page:

```
Control  LowFat  LowCarbo  LowCalorie
 1.2       3.0      3.4        6.6
```

and then all the ones that are *not* significantly different are joined by a line. To figure that, you start from Control, and, using the Tukey output, see whether it's different from Low Fat or Low Carbo (not, in either case) or from Low Calorie (it is different). Then you join the non-significantly-different ones by a line:

```
Control  LowFat  LowCarbo  LowCalorie
 1.2       3.0      3.4        6.6
 ----------------------
```

Then you start from Low Fat and repeat, adding lines as necessary (no extra lines here), and then from Low Carbo (no lines).

This makes it clear that Low Calorie is better than *all* of the others, and also that there are no other significant differences (the last two points in the question). I find it easier to see this on the diagram than by trying to synthesize all six of those Tukey comparisons.

The old-timey calculation with the Studentized Range table gave you a single number, that I think we called W, and any differences in means bigger than W were significant at your α. Here, with $\alpha = 0.05$, W is about 3. Then it was easy enough to eyeball the means and see which ones you had to draw lines underneath.

This works if the groups are all the same size, as they are here (5 observations in each). If they are different sizes, you use something called Tukey-Kramer, which is easy enough in software (done automatically by `TukeyHSD`), but a real pain in the neck when done by hand.

The other thing that should probably also be said is that I asked you to compare weight loss on *all* the pairs of diets. This is perhaps not so realistic, but it corresponds with what we have been doing (and I didn't want to throw a surprise at you on the final exam). The realistic way to tackle this is perhaps to compare all the other diets *with the control* rather than with each other, to answer a question like "are any of these diets better than nothing?". This calls for **Dunnett's method** (named for a Canadian statistician who worked at McMaster). Dunnett developed a Tukey-like method for comparing all the real treatments with a designated control, adjusting the type I error probability appropriately. I was sure I had a writeup on this, but I can't seem to find it, so you are spared reading about that now. (It was in the 2016 final exam if you want to look.)

2. In a packing plant, a machine packs cartons with jars. A new packing machine is being tested. Will it pack faster on average than the machine currently used? To test that hypothesis, the times it takes each machine to pack ten cartons are recorded. The data are shown in Figure 4, and have been uploaded as a file `packing.txt` to my account `ken` on SAS Studio.

   (a) (3 marks) Give code to read the data into a SAS data set called `packing`. Look carefully at the format of the data before you write any code.

Solution: First, note that the data values are separated by commas (that was my hint), so this is actually a CSV file. The code should therefore reflect that:

```
proc import
  datafile='/home/ken/packing.txt'
  out=packing
  dbms=csv
  replace;
  getnames=yes;
```

I didn't ask you to list the data, but I want to check that this is correct:

```
proc print;
```

Obs	New	Old
1	42.1	42.7
2	41	43.6
3	41.3	43.8
4	41.8	43.3
5	42.4	42.5
6	42.8	43.5
7	43.2	43.1
8	42.3	41.7
9	41.8	44
10	42.7	44.1

If you didn't realize that this was an example of a CSV file, you could treat it as a delimited text file, as long as you remembered to say that the delimiter was a comma, thus:

```
proc import
  datafile='/home/ken/packing.txt'
  out=packing
  dbms=dlm
  replace;
  getnames=yes;
  delimiter=",";


proc print;
```

Obs	New	Old
1	42.1	42.7
2	41	43.6
3	41.3	43.8
4	41.8	43.3
5	42.4	42.5
6	42.8	43.5
7	43.2	43.1
8	42.3	41.7
9	41.8	44
10	42.7	44.1

Minus one point per error, with the proviso that if you got *something* right, you would get a point. I wasn't fussy about semicolons (SAS will generally figure out what you meant in a `proc import` anyway), and I didn't care what you called the dataset (in `out=`). Some people treated the file as if it had been uploaded to a web server and downloaded it from there, which I went with if it looked plausible.

I didn't do reading in data with a `data` step in this course, but if you could do it and get it right, I was good with that. (Most people who tried this got something wrong.) Here's how it would go:

```
data packing;
  infile '/home/ken/packing.txt' dlm="," firstobs=2;
  input new old;


proc print;
```

Obs	new	old
1	42.1	42.7
2	41.0	43.6
3	41.3	43.8
4	41.8	43.3
5	42.4	42.5
6	42.8	43.5
7	43.2	43.1
8	42.3	41.7
9	41.8	44.0
10	42.7	44.1

(b) (2 marks) Give SAS code to find the mean packing times classified by whether the machine is old or new. (Output that contains the mean and other things is fine.)

Solution: This sort of thing usually has a `class` in it, which I was trying to make you think here, but there is a column of packing times for the old machine and a column of packing times for the new machine, so just feed them into `var`:

```
proc means;
  var old new;
```

```
                          The MEANS Procedure

Variable     N            Mean         Std Dev         Minimum         Maximum
------------------------------------------------------------------------------
old         10      43.2300000       0.7498889      41.7000000      44.1000000
new         10      42.1400000       0.6834553      41.0000000      43.2000000
------------------------------------------------------------------------------
```

I realized afterwards that `old` and `new` are the only variables in the dataset, so all you actually need to do is this:

```
proc means;
```

```
                          The MEANS Procedure

Variable     N            Mean         Std Dev         Minimum         Maximum
------------------------------------------------------------------------------
new         10      42.1400000       0.6834553      41.0000000      43.2000000
old         10      43.2300000       0.7498889      41.7000000      44.1000000
------------------------------------------------------------------------------
```

and that would net you two points, by happy accident.

Does this work?

```
proc means;
  var old;
  var new;
```

```
                          The MEANS Procedure

Variable     N            Mean         Std Dev         Minimum         Maximum
------------------------------------------------------------------------------
old         10      43.2300000       0.7498889      41.7000000      44.1000000
new         10      42.1400000       0.6834553      41.0000000      43.2000000
------------------------------------------------------------------------------
```

It does. Two points. (I wasn't sure whether it would, so I ran it to see.) Also good is two separate `proc means`, one for each variable. (I didn't ask for "elegant", so if it works, it's good.)

Any of the above with `proc means mean` is also good; that'll get the means only, which is fine. I didn't insist on that, because I wanted to keep things simple (which is why I said "mean and other things" in the question), but it's no problem if you did it. If you want to name a statistic, it had better be the mean, though!

There were a few ways to get 1 point, basically one of the above, but with a small error, like saying `var=` or trying to add a `class` line with the `var` line (basically) correct, or calling it `proc mean` instead. I decided to forgive a comma between `old` and `new`, so that's two points.

Anything else is nothing. We haven't defined `packtime` and `machine` yet, so you can't use those. (That's (d).) I noticed that one person had answered (d) in (b) and left (d) blank. I'll try to find them some points.

(c) (4 marks) The analyst at the packing company wants all the packing times in one column of a SAS data set, with a second column saying which machine the packing time came from. The new columns should be called `packtime` and `machine` respectively. The columns from the original data set should be removed. Give SAS code to create a data set in this format.

Solution: This is the SAS version of `gather`. Find the example in your lecture notes and adapt for this situation. This is the array way (best, or at least scalable):

```
data packing2;
```

```
set packing;
array machine_array [2] new--old;
do i=1 to 2;
  packtime=machine_array[i];
  machine=vname(machine_array[i]);
  output;
end;
keep packtime machine;

proc print;
```

It has to be `new--old` since the times for the new machine are in the *first* column of the data set `packing`.

Obs	packtime	machine
1	42.1	new
2	42.7	old
3	41.0	new
4	43.6	old
5	41.3	new
6	43.8	old
7	41.8	new
8	43.3	old
9	42.4	new
10	42.5	old
11	42.8	new
12	43.5	old
13	43.2	new
14	43.1	old
15	42.3	new
16	41.7	old
17	41.8	new
18	44.0	old
19	42.7	new
20	44.1	old

Here, or below, you can `drop old` and `new` instead of `keep`ing what you want to keep. Equally good.

If you don't like this, the "tedious way" is also good, especially since there are only two columns here, so it is not actually that tedious:

```
data packing3;
  set packing;
  machine='old';
  packtime=old;
  output;
  machine='new';
  packtime=new;
  output;
  keep machine packtime;

proc print;
```

Or `new` and `old` in the other order, or with Capital Letters. This way is actually quite a lot more forgiving than the array one. (Strictly speaking, `new--old` needs two dashes rather than one because the variable names don't have numbers like the example in class. Thus `new--old` but `x1-x5`. But I am not going to penalize you for that.)

Obs	machine	packtime
1	old	42.7
2	new	42.1
3	old	43.6
4	new	41.0
5	old	43.8
6	new	41.3
7	old	43.3
8	new	41.8
9	old	42.5
10	new	42.4
11	old	43.5
12	new	42.8
13	old	43.1
14	new	43.2
15	old	41.7
16	new	42.3
17	old	44.0
18	new	41.8
19	old	44.1
20	new	42.7

I didn't ask for the `proc print` at the end of these, but I wanted to check for myself that it worked (optional for you).

I asked for variables with those names so that everyone would have the same code in the next part, which would be easier to mark consistently.

The array way and the tedious way of solving this came out on different pages of my solutions, so I actually went through the pile twice, once for the people who attempted the array way (or didn't attempt it at all) and once for the people who attempted the tedious way. I was impressed (a) with how many people attempted the array way and (b) how many of those got it right.

For those who attempted the array way and got it substantially right (that is, minus one or two small but consequential errors such as forgetting the `keep`, numbering the array or loop up to something other than 2, or forgetting the `output`), you probably got 3. If you got the structure right but were missing something substantial, you probably got 2.

For those who attempted the tedious way: something small like missing an `output` (there should be two) or forgetting the `keep` (or `drop`) got you 3 points. If you mixed up the quoting on `machine='old'` and `packtime=old`, I looked to see if you'd done it once (a slip, 3 points) or more than once (a lack of understanding of why the quotes were there or not, 2 points). The idea is that `machine` should contain the *name* of a machine, ie. text, with quotes, but `packtime` should be a *number*, without quotes (the *value* of `old` or `new`). This was perhaps a bit clearer the array way, where the i-th thing in the array was clearly a number (since `old` and `new` were both numbers) and to get its name you had to do the `vname` thing. That is to say, easier to think about in the "tedious" way, but also easier to get wrong.

I tried to give 1 point (on either approach) to anyone who had made an important step towards solving the problem but hadn't gone any further.

That might have been the hardest thing on the entire exam.

(d) (2 marks) Using the new data set created in the previous part, give code to find the mean packing time classified by whether the machine is old or new. (Output that contains the mean and other

things is fine.)

Solution: Having messed with your mind in part (b), I need you to see that this is now exactly the usual thing with a `var` and a `class`. Use your variable names from the new data set:

```
proc means;
  var packtime;
  class machine;
```

```
                          The MEANS Procedure

                      Analysis Variable : packtime

                N
machine      Obs    N           Mean        Std Dev        Minimum        Maximum
-------------------------------------------------------------------------------
new           10   10     42.1400000      0.6834553     41.0000000     43.2000000

old           10   10     43.2300000      0.7498889     41.7000000     44.1000000
-------------------------------------------------------------------------------
```

The means (and the other things) are exactly the same as before, as they should be.

In either (b) or (d), you can get just the mean with `proc means mean` if you wish to.

For the same reason that just `proc means` works in (b), you can leave off the `var` here too:

```
proc means;
  class machine;
```

```
                          The MEANS Procedure

                       Analysis Variable : packtime

                 N
machine    Obs    N          Mean        Std Dev        Minimum        Maximum
-------------------------------------------------------------------------------
new         10   10     42.1400000      0.6834553     41.0000000     43.2000000

old         10   10     43.2300000      0.7498889     41.7000000     44.1000000
-------------------------------------------------------------------------------
```

The logic is that the variables whose means are calculated are all the quantitative variables except the ones named in the `class`, which is only `packtime` here since we got rid of `old` and `new`. You can't leave the `class` off here, though, since then you'll get the mean packing time of both machines together. 1 point for doing that. (I didn't realize until partway through the pile that leaving off the `var` line would work, so I had to go back through and give an extra point to some people.) I thought a plain `proc means` with no `class` was not enough to get a point, so that's zero. This reserves 1 point for attempts that got close to a right answer.

You might not have been able to do (c), but *once you have the answer to (c)*, this part is not difficult. You're allowed to assume that you have the answer to (c) while doing (d), *even if you don't*. That's exam technique. It applies to any exam.

Some people had a very funky way of doing this part:

```
proc anova;
  class machine;
  model packtime=machine;
  means machine / tukey;
```

```
                          The ANOVA Procedure

                       Class Level Information

                    Class         Levels    Values

                    machine            2    new old
                  Number of Observations Read          20
                  Number of Observations Used          20
                            The ANOVA Procedure

                    Dependent Variable: packtime

                                     Sum of
Source                     DF       Squares    Mean Square   F Value   Pr > F

Model                       1     5.94050000     5.94050000     11.54   0.0032

Error                      18     9.26500000     0.51472222

Corrected Total            19    15.20550000
           R-Square     Coeff Var      Root MSE    packtime Mean

           0.390681      1.680781      0.717441         42.68500
Source                     DF       Anova SS    Mean Square   F Value   Pr > F

machine                     1     5.94050000     5.94050000     11.54   0.0032
                            The ANOVA Procedure

            Tukey's Studentized Range (HSD) Test for packtime

NOTE: This test controls the Type I experimentwise error rate, but it generally
                has a higher Type II error rate than REGWQ.


                Alpha                                      0.05
                Error Degrees of Freedom                     18
                Error Mean Square                     0.514722
                Critical Value of Studentized Range  2.97105
                Minimum Significant Difference        0.6741

        Means with the same letter are not significantly different.


        Tukey Grouping          Mean      N    machine

                    A       43.2300     10    old

                    B       42.1400     10    new
```

Right down at the bottom are the mean packing times for each machine. So this is good too, as is `proc ttest` since that shows the mean for each group as well.

This problem calls for a two-sample t-test, and so the data set in the layout in `packtime2` or `packtime3` is what is needed; the format of `packtime` is no good, because there needs to be a `class` and there isn't one. (This is the same thing that stopped the `proc means` of (d) working in (b).) The ANOVA above gives the exact same P-value as a two-sided two-sample t-test would, and indicates that there *is* a significant difference in mean packing time between the two machines. I don't know what units the times are measured in (seconds?); the new machine is not much quicker than the old machine, but it is consistently quicker enough for that difference to be significant.

3. A researcher is planning a study to look at a new treatment. The mean of the standard procedure (in suitable units) is 12, but the researcher hopes that the new treatment will increase the mean to 16. The standard deviation of the measurements is believed from previous studies to be about 8.5. The researcher will use a one-sample t-test. The researcher's primary question is: how large a sample size is needed to obtain probability at least 0.75 of correctly rejecting the null hypothesis that the mean is 12, in favour of an alternative that the mean is greater than 12?

   (a) (3 marks) Give SAS code to find the necessary sample size.

Solution: This is a power calculation (or, at least, a sample size for a given power). It's a one-sample t-test, thus:

```
proc power;
  onesamplemeans
  test=t
  mean=16
  nullmean=12
  sides=U
  stddev=8.5
  ntotal=.
  power=0.75;
```

```
                    The POWER Procedure
                 One-Sample t Test for Mean

                    Fixed Scenario Elements

Distribution                    Normal
Method                           Exact
Number of Sides                      U
Null Mean                           12
Mean                                16
Standard Deviation                 8.5
Nominal Power                     0.75
Alpha                             0.05
          Computed N Total

          Actual        N
           Power    Total

           0.755       26
```

This is really just the same as the one on the midterm (actually, the second part of the question on the midterm). The sample size required is 26. (This, as usual, gets you slightly more power than you wanted; the actual power is a 0.755, a bit greater than 0.75. A sample of size 25 would get you not quite enough power.)

As on the midterm, the way below also works. Instead of specifying the null mean and the true mean, you don't specify the null mean, and `mean` is the difference between the null and the truth:

```
proc power;
  onesamplemeans
  test=t
  mean=4
  sides=U
  stddev=8.5
  ntotal=.
  power=0.75;
```

The POWER Procedure
One-Sample t Test for Mean

Fixed Scenario Elements

Distribution	Normal
Method	Exact
Number of Sides	U
Mean	4
Standard Deviation	8.5
Nominal Power	0.75
Null Mean	0
Alpha	0.05

Computed N Total

Actual Power	N Total
0.755	26

Same answer, so it's equally good.

Marking: since there was a lot to get right, I decided to allot 2 marks to 1–2 mistakes, and 1 mark for 3 mistakes or more. Almost everybody found a way to get something wrong, the commonest being the `sides`: if you leave that out, you'll get the power for a two-sided test, which will be different, and not what you wanted. Another error was to say `test=diff_satt`, which only goes with a two-sample test (so would fail to run if you tried it in SAS). People managed to get the means mixed up, also; if you only give one, it has to be the *difference* between null and truth (ie. 4), while if you give the actual value for `mean`, you have to supply a value for `nullmean` also.

If you give both means, there are 8 things to say after the `proc power`, while if you give the difference in means, there are 7.

Any of SAS's variant names for things are good also, such as `side` instead of `sides`.

(b) (3 marks) The output of your code from part (a) is shown in Figure 5. Figure 6 shows the output from almost the same code, except that I added `alpha=0.01` to the code, to carry out the test at $\alpha = 0.01$. Compare the sample sizes from Figures 5 and 6. Are the results surprising, or not?

Explain (reasonably) briefly.

Solution: The input `alpha` is the α for the test. The default α is 0.05: we reject the null if the P-value is less than 0.05. This is what is used in the first analysis. In the second sample size analysis, we only reject if the P-value is less than 0.01. With a smaller α, it is harder to reject if the null hypothesis is true (the probability of a type I error is smaller), but it is also harder to reject if the null hypothesis is false, as it is in this situation. (Another way of seeing this is that the data have to be further away from the null hypothesis at $\alpha = 0.01$ than at $\alpha = 0.05$.) Thus, for a fixed sample size, the probability of a type II error is *larger* if α is smaller, and thus the power is *smaller*. Translating that into sample sizes, to make the power the same in the two cases, we have to use a *bigger* sample size when `alpha=0.01`. This is what is shown in the output (sample size 44 vs. 26), so the results shown in Figures 5 and 6 are not surprising at all.

That's the entire thought process. An acceptable but much briefer answer is something like this:

> The second output uses a smaller `alpha` than the first one (0.01 vs. 0.05). So, in the second case, we are less likely to reject the null whether it is true or whether it is false, and therefore it takes a larger sample size to get the same power. Hence, it is not at all surprising that the second output produced a larger sample size.

This hits the major points:

- The α values used, and what having a smaller α means in terms of rejecting the null (whether it is true or false)
- what that implies for power and hence sample size
- comparison with the output.

Expect to get one point for saying or implying each of these. If you say that the required sample size is bigger in Figure 6 than in Figure 5, and nothing else, you should end up with one point.

I didn't think that you could answer this by appealing to confidence intervals, but I changed my mind on that. $\alpha = 0.05$ corresponds to a 95% CI and $\alpha = 0.01$ to a 99% CI. The 99% CI will be longer if all else is equal (that is, if the sample sizes are the same) and thus is more likely to include 12 (the null mean) even if centred around the true mean of 16. To make it have the same chance of having 12 outside (and thus rejecting the null), we have to make the CI shorter, and the only freedom we have to do that is to make the sample size bigger.

I think that's a three-point answer, but I think you have to connect all those dots (talk at least about including/excluding the null mean). Something less detailed but based on the same ideas would be 2 points.

This isn't strictly correct because the test is one-sided and a confidence interval is two-sided, but it gets at the ideas sufficiently well for me.

Another line of argument you can follow is the "effect size" one: the power depends on the effect size (difference between null and true means), α and the sample size. The power and the effect size are the same, but α has changed, and so the sample size must change in response. That would be two points. The third point is for making the case that the sample size needs to be *bigger* rather than smaller; you can do that by saying that if we keep the sample size the same, the power will be *smaller*, since it will be harder to reject the null (whether correct or incorrect) at smaller α, so we had better make the sample size bigger. (There is a certain amount of argumentational looseness here in that it is not completely clear what we are allowed to assume and what has to be argued for.)

This wound up being more difficult to mark than I would have liked, and the marking scheme I mentioned above was a bit of a rough guide at times. My criterion turned out to be this: mentioning the results in Figs 5 and 6: 1 point. In addition, making an (incomplete) argument for the results as shown: 2

points. If I thought you made a complete argument, typically along the lines of my original answer, or the "confidence level" argument, or the "effect size" argument, then I gave you 3.

An argument is "a connected series of statements to establish a definite proposition". I got that from here: `https://www.youtube.com/watch?v=kQFKtI6gn9Y`. This may not be a reliable guide to anything else at all, but is a fine chance to watch John Cleese in action.

The required sample size is not bigger because the actual power is bigger. The only reason for "actual power" appearing there is because the sample size has to be a whole number. Doing the calculation to find a fractional sample size (to get the power to exactly 0.75) would come out like this:

```
power.t.test(delta=4,sd=8.5,sig.level=0.05,power=0.75,
             type="one.sample",alternative="one.sided")

##
##      One-sample t test power calculation
##
##               n = 25.69678
##           delta = 4
##              sd = 8.5
##       sig.level = 0.05
##           power = 0.75
##     alternative = one.sided
```

I didn't really need to input α (as `sig.level`), since it defaults to 0.05, but that makes the change for the second one clearer:

```
power.t.test(delta=4,sd=8.5,sig.level=0.01,power=0.75,
             type="one.sample",alternative="one.sided")

##
##      One-sample t test power calculation
##
##               n = 43.40601
##           delta = 4
##              sd = 8.5
##       sig.level = 0.01
##           power = 0.75
##     alternative = one.sided
```

The fractional sample size changes from 25.7 to 43.4. SAS rounded these up; since there was more rounding on the second one, the "actual power" was a little bigger.

This part is a situation you haven't seen before, so I am testing your ability to think about the issues.

4. A study is undertaken to explore how age is related to sense of smell. Altogether, 180 subjects, aged between 20 and 89, are exposed to 40 different odours: for each odour, subjects are asked to choose which one of four words best describes the odour. A "smell score" is calculated for each subject, based on the number of odours correctly identified (and therefore a higher smell score is better). A subject's age is recorded as an age group as follows: group 1: age ≤ 25; group 2: $25 <$ age ≤ 40; group 3: $40 <$ age ≤ 55; group 4: $55 <$ age ≤ 70; group 5: age > 70. A higher-numbered age group thus contains older people.

   It is believed that smell score will decrease with age on average, but that in older people, some will have an excellent sense of smell, but some will have a much worse sense of smell.

   Some of the data set is shown in Figure 7. (There are too many observations to show them all.)

   (a) (2 marks) Figure 8 shows side-by-side boxplots of smell score by age group. Because there is a lot of data, the statistician on the study is not concerned about either the skewness or the outliers. What other assumption is made in order to do analysis of variance, and how is that assumption not satisfied here? Explain briefly.

Solution: The other assumption is equal variance (spread) in all the groups. In this case, though, the age groups seem to have increasing spread (as measured by the IQR) as you go across the page, from younger to older.

I was relatively relaxed about the exact answers that I would accept, but I needed to see two things:

- a statement of the assumption that is needed (equal spreads within each group, or something equivalent)
- an assessment of whether that assumption was satisfied (I would accept an assertion of "the group spreads are not close to the same", but I would have preferred something more specific like "group 5 has a larger spread than the others", as evidenced by the height of its boxplot).

If you noted that the boxplots were of unequal heights, that was worth one point, but only one point unless coupled with a statement or strong implication that we need them to be similar.

   (b) (2 marks) Explain briefly why Welch's method for analysis of variance is preferable to the standard ANOVA here. (There are two points to make.)

Solution: Welch's method is used when (i) we are happy with the normality (as the statistician here apparently is) but (ii) we are not happy with equal variances (which we are not here), since it can account for these.

A complete answer mentions both those things: normality OK, but with unequal variances, and connects this piece of theory or knowledge with what is happening in this data set. If you copy something like "if normality OK but equal spreads not" from my notes and give that as your entire answer, expect to get only one point because you haven't connected it to this data set. If you use your own words to talk about the requirements of normality and equal spread, I was willing to cut you some slack. The word "here" in the question is a reminder that you are supposed to connect back to *this* data set rather than just giving me a piece of knowledge.

This is perhaps a giveaway to part (a), if you remember *why* the Welch ANOVA is usually done.

Remember that we still need at least some kind of approximate normality, or else we are off into the domain of Mood's median test. This is what we'd use if the normality definitely fails, even taking into account the sample sizes (regardless of whether the spreads are equal or not). That is to say, the first question you ask yourself is "are my data acceptably normal?" If the answer is no, you go straight to Mood's median test. If the answer is yes, you *then* think about whether you have equal enough spreads. If you do, regular ANOVA; if not, Welch ANOVA.

I should draw a flow chart. (This is possible in R, by the way.)

(c) (3 marks) Give SAS code to run the Welch ANOVA for these data.

Solution: This code was used to make Figure 9:

```
proc anova;
  class agegroup;
  model smell=agegroup;
  means agegroup / hovtest=levene welch;
```

The `hovtest=levene` is not needed, but was done in class, so I have no issues if it's there or not.

Minus one per error, as usual, and at the other end of the scale, `proc anova` (even with nothing else) guarantees you one. (Thus one point is a widish range.)

Levene's test is a test of equal variances (that uses the median, so is not affected by outliers). I discussed it in one of the assignment solutions (but not in class, so I won't ask you about it). Here, we resoundingly reject equal variances, which is further evidence for using the Welch ANOVA rather than the standard one.

(d) (3 marks) Give the F value and P-value for the Welch ANOVA, as shown in Figure 9. What do you conclude, in the context of the data?

Solution: We have to look at the table marked "Welch's ANOVA" and ignore the top table that we would normally use, so the F value is 13.72 and the P-value is less than 0.0001. Thus, allowing for the age groups having unequal spreads of smell scores, we can conclude that mean smell scores do differ for at least some of the age groups, or that the mean smell scores by age group are not all equal, or something equivalent to that.

You'll recall how the Satterthwaite t-test and the pooled t-test often give surprisingly similar results, even when the two groups have clearly unequal spread. Here, the comparison is with the regular ANOVA at the top of Figure 9 and the Welch ANOVA at the bottom. The P-values are both "very small", and the F-values, 13.72 and 16.65, are not that different, so the conclusions are, again, basically the same even though the age groups very clearly differ in spread.

Normally, we'd go on and do something like Tukey now, but Tukey only applies when we trust the equal-variances assumption. We ought to do something like Games-Howell here, but that takes us beyond the scope of what we've seen in class, so I stop here for this question.

Marking notes: I asked for the F-statistic value because there are three tests shown on the output, all with the same P-values, and I wanted to be sure you had the right one. One mark for the right F-statistic and P-value (for the latter, I would accept the implication from your next statement); one for making a decision about rejecting, and one for making a statement about mean smell scores depending on age. I was pretty relaxed about this last point, if you seemed to have said something sensible. I tried to find one point (altogether) if you'd said *something* correctly, and I tried to give you two if your answer was "mostly correct" in my opinion. (Mentioning Games-Howell or some other salient point might have tipped the balance on this.)

Note that the Welch ANOVA is like regular ANOVA in that it will find *any* differences in smell score between age groups. The research interest might have been in smell scores *decreasing* with age, but we were not actually testing that. There is a test, called the Terpstra-Jonckheere test, which tests that all the group *medians* are equal against an alternative that they are in a prespecified order. It's in the same

spirit as the Kendall (nonparametric) correlation. Wikipedia has it: look up "Jonckheere's trend test". It would work for these data.

You might also think of coding the age categories as numbers 1 through 5, and then doing a regression to predict smell score from age category. But this would suffer from the same problem as the ANOVA does: the scores in the last category are more spread out, so there will be fanning-out in the regression.

5. This question is about R Markdown.

   (a) (2 marks) Suppose you have just opened a new R Markdown document. What code would insert a new section called "Introduction"?

Solution: This, rather a gimme:

```
## Introduction
```

I'll accept more or fewer hash-signs (the correct word for these is "octothorpes"), or any of the other Markdown variant ways of getting section headings, such as "underlining" with equals signs. `*** Introduction` does *not* work. I tried it. Asterisks around text do *italics* (one) or **boldface** (two).

There were various things for which I gave one point, that I thought were reasonable misunderstandings of the question.

A distressing number of people left this whole question blank. *This was in your lecture notes,* which I hope you brought with you to the exam. The whole question was meant to be an easy one.

   (b) (2 marks) How would you add a code chunk to your R Markdown document? Give the code you would use, or describe the procedure you would follow.

Solution: The code way is this:

````
```{r}

```
````

and then the code goes in between the first line and the last one. Or you can get it by clicking Insert above the window with the R Markdown in it, and selecting R from the drop-down (to insert an R code chunk rather than one in another language). This inserts the above code into the R Markdown document at the cursor.

Another way is to use the Code menu (at the top of the screen) and from that select Insert Chunk.

If you're a keyboard fan like me, control-alt-I will do it too.

Describing the procedure clearly is enough (I didn't care if you then made any mistakes in what the code chunk would look like). But if you just showed what a code chunk would look like (in R Markdown code), you needed to get it right.

There was one mark available for "nearly right", whatever I thought that meant.

   (c) (2 marks) Describe briefly in your own words why it is a good idea to use an R Markdown document (at least if you are coding in R), compared to copying and pasting code and output into a Word document.

Solution: The value of an R Markdown document is that the output in it must have come from the code in it (because the code was actually run to produce the output). An R Markdown document is reproducible in the sense that if you give it and the data to someone else, they can re-run it and get exactly the same results as you, guaranteed. This is unlike copying and pasting to Word, where *you* rather than the computer have to make sure that everything is in sync, and the only check to make sure that happens is your carefulness.

Some of that. "Reproducible" with some discussion of what that means, or "guaranteed" with some discussion of what *that* means. "In your own words" means "don't expect to get any marks if you just

copy something I wrote". It is rare that copying what I wrote word-for-word will help you much; you'll normally have to change something.

I tried very hard to give you two points on this part. If you got less than that, it is because something important was, to me, clearly missing. There were a few people who lost marks by using some of my words instead of their own, and one whose answer was 100% my words (who got no marks). When your writing style is like mine, it is easy to recognize when your own sentences are being used!

(d) (2 marks) Your boss looks at the output from your R Markdown document and says "this would be better if you had a histogram *right there*" (and points at the spot). When you get back to your computer, what steps would you take to produce a new output for your boss with the histogram in the right place? Describe those steps. (There are two or three steps depending on how you count them.)

Solution: The steps are:

1. In the R Markdown document, create a new code chunk at the right spot.
2. Insert the code to make the histogram in it. (Or, less precisely, turning these two steps into one, "add the code to produce the histogram to the R Markdown document". This is precise enough.)
3. Knit the R Markdown again to produce the new output. (This must appear somewhere.)

Two marks if I saw something about making a histogram and something like "knit". I would also accept "run the code to make the histogram", since you could do that by clicking on the green arrow next to the code chunk. (However, if you were producing a new copy of the report, you would probably have to knit the whole document.) As I was going through, I was almost keying on the word "knit", but I *did* check that you were saying to produce a histogram in the right place as well.

One mark if you failed to mention, in some fashion, that you needed to re-knit the document (and probably a squiggle from me in the right margin).

6. In Chile in 1988, there was a "plebiscite" or special election, to determine whether army general Augustin Pinochet should continue as President for another 8 years. Before the plebiscite, a survey was taken of 2700 respondents. For each respondent, the following variables were recorded. They appear in the data frame in this order:

   - `region` of Chile where the respondent lives: SA, in Santiago; M, in the metropolitan Santiago area; N, in northern Chile; S, in southern Chile; C, in central Chile.
   - `population`: the population of the city where the respondent lives.
   - `sex`: M (Male) or F (Female).
   - `age` in years.
   - `education`: P: primary (elementary) only; S: secondary (high school): PS: post-secondary (university or college).
   - `income`: monthly income in pesos.
   - `statusquo`: result of a questionnaire about General Pinochet. A positive number means the respondent likes Gen. Pinochet, a negative number means they dislike him.
   - `vote`: how the respondent intends to vote: Y (yes, for Pinochet), N (no, against Pinochet), A (will abstain, that is, not vote at all), U (undecided).

   The data frame is called `chile`. Give R code, using ideas from the `tidyverse`, to accomplish the following tasks:

   (a) (2 marks) Display only the columns containing sex and education.

Solution: Let me grab the data frame first:

```
chile=read_csv("chile.csv")
```

```
## Warning:  Missing column names filled in:  'X1' [1]
```

```
## Parsed with column specification:
## cols(
##  X1 = col_integer(),
##  region = col_character(),
##  population = col_integer(),
##  sex = col_character(),
##  age = col_integer(),
##  education = col_character(),
##  income = col_integer(),
##  statusquo = col_double(),
##  vote = col_character()
## )
```

```
chile
```

```
## # A tibble: 2,700 x 9
##       X1 region population   sex   age education income statusquo  vote
##    <int>  <chr>      <int> <chr> <int>     <chr>  <int>     <dbl> <chr>
##  1     1      N     175000     M    65         P  35000   1.00820     Y
##  2     2      N     175000     M    29        PS   7500  -1.29617     N
##  3     3      N     175000     F    38         P  15000   1.23072     Y
##  4     4      N     175000     F    49         P  35000  -1.03163     N
##  5     5      N     175000     F    23         S  35000  -1.10496     N
##  6     6      N     175000     F    28         P   7500  -1.04685     N
##  7     7      N     175000     M    26        PS  35000  -0.78626     N
##  8     8      N     175000     F    24         S  15000  -1.11348     N
##  9     9      N     175000     F    41         P  15000  -1.01292     U
## 10    10      N     175000     M    41         P  15000  -1.29617     N
## # ... with 2,690 more rows
```

I seem to have gained an extra column `X1` (the respondent number) that I will ignore.

Then I use `select`:

```
chile %>% select(sex,education)
```

```
## # A tibble: 2,700 x 2
##      sex education
##    <chr>     <chr>
##  1     M         P
##  2     M        PS
##  3     F         P
##  4     F         P
##  5     F         S
##  6     F         P
##  7     M        PS
##  8     F         S
##  9     F         P
## 10     M         P
## # ... with 2,690 more rows
```

There is a trap here in that `sex` and `education` are not adjacent, so if you do this without thinking you'll get three columns instead of two:

```
chile %>% select(sex:education)
```

```
## # A tibble: 2,700 x 3
##      sex   age education
##    <chr> <int>     <chr>
##  1     M    65         P
##  2     M    29        PS
##  3     F    38         P
##  4     F    49         P
##  5     F    23         S
##  6     F    28         P
##  7     M    26        PS
##  8     F    24         S
##  9     F    41         P
## 10     M    41         P
## # ... with 2,690 more rows
```

As I write this, I was giving you one for

```
chile %>%
```

followed by pretty much anything, but I might be changing my mind on this.

(b) (2 marks) Display the columns `age`, `income` and `vote` *without naming* the columns.

Solution: Without naming them is a hint that you need a select-helper, which means finding what they all have in common: they all end with "e", and none of the other columns do, so:

```
chile %>% select(ends_with("e"))
```

```
## # A tibble: 2,700 x 3
##      age income  vote
##    <int>  <int> <chr>
##  1    65  35000     Y
##  2    29   7500     N
##  3    38  15000     Y
##  4    49  35000     N
##  5    23  35000     N
##  6    28   7500     N
##  7    26  35000     N
##  8    24  15000     N
##  9    41  15000     U
## 10    41  15000     N
## # ... with 2,690 more rows
```

A less elegant but still workable solution is to omit the ones you don't want:

```
chile %>% select(-region,-population,-sex,-education,-statusquo)
```

```
## # A tibble: 2,700 x 4
##       X1   age income  vote
##    <int> <int>  <int> <chr>
##  1     1    65  35000     Y
##  2     2    29   7500     N
##  3     3    38  15000     Y
##  4     4    49  35000     N
##  5     5    23  35000     N
##  6     6    28   7500     N
##  7     7    26  35000     N
##  8     8    24  15000     N
##  9     9    41  15000     U
## 10    10    41  15000     N
## # ... with 2,690 more rows
```

This way, it's very easy to go wrong, so it's worth taking a moment to find a better (less error-prone) way to do it. (This output contains my extra column `X1`, which you didn't know you had. I accept this code.)

This doesn't work:

```
chile %>% select(-(region,population,sex,education,statusquo))
```

```
## Error:  <text>:1:26:  unexpected ','
## 1:  chile %>% select(-(region,
##                             ^
```

To do it this way, you need this:

```
chile %>% select(-c(region,population,sex,education,statusquo))
```

```
## # A tibble: 2,700 x 4
##       X1   age income  vote
##    <int> <int>  <int> <chr>
##  1     1    65  35000     Y
##  2     2    29   7500     N
##  3     3    38  15000     Y
##  4     4    49  35000     N
##  5     5    23  35000     N
##  6     6    28   7500     N
##  7     7    26  35000     N
##  8     8    24  15000     N
##  9     9    41  15000     U
## 10    10    41  15000     N
## # ... with 2,690 more rows
```

You have to have a collection of inputs to `select` (with, here, a minus sign on the front of each one), or a vector of names, formed with `c`, that you then "negate" the whole of.

Yet another way is to select the columns by *number*: columns 4, 6 and 8 out of the list in the question, or 5, 7 and 9 out of the data frame shown in the Figure (I'd accept either).

Another idea was to use `starts_with`, since `age`, `income` and `vote` are the only ones starting with those letters. This works:

```
chile %>% select(starts_with("a"),starts_with("i"),starts_with("v"))
```

```
## # A tibble: 2,700 x 3
##      age income  vote
##    <int>  <int> <chr>
##  1    65  35000     Y
##  2    29   7500     N
##  3    38  15000     Y
##  4    49  35000     N
##  5    23  35000     N
##  6    28   7500     N
##  7    26  35000     N
##  8    24  15000     N
##  9    41  15000     U
## 10    41  15000     N
## # ... with 2,690 more rows
```

(actually to my surprise: there were a couple of people who did that, and I now have to go back and mark them right). How about this?

```
chile %>% select(starts_with("a","i","v"))
```

```
## Error in if (ignore.case) match <- tolower(match):  argument is not interpretable
as logical
```

Nope. Only one input to `starts_with` at a time. But one point for a good try. Here's another one that works (if you are careful):

```
chile %>% select(matches("e$"))
```

```
## # A tibble: 2,700 x 3
##      age income  vote
##    <int>  <int> <chr>
##  1    65  35000     Y
##  2    29   7500     N
##  3    38  15000     Y
##  4    49  35000     N
##  5    23  35000     N
##  6    28   7500     N
##  7    26  35000     N
##  8    24  15000     N
##  9    41  15000     U
## 10    41  15000     N
## # ... with 2,690 more rows
```

The regular expression inside `matches` says "contains an `e` and then the end of the column name", which is the same thing as "ends with". The people who tried this got it slightly wrong. This also works:

```
chile %>% select(matches("^.*e$"))
```

```
## # A tibble: 2,700 x 3
##      age income  vote
##    <int>  <int> <chr>
##  1    65  35000     Y
##  2    29   7500     N
##  3    38  15000     Y
##  4    49  35000     N
##  5    23  35000     N
##  6    28   7500     N
##  7    26  35000     N
##  8    24  15000     N
##  9    41  15000     U
## 10    41  15000     N
## # ... with 2,690 more rows
```

This is read as "the start of the column name followed by any number of any character followed by an `e` followed by the end of the column name". But when it doesn't matter what the other characters are apart from the last one, there's no point including them in the regular expression; the could-be-anything is already implied.

This gets the wrong answer:

```
chile %>% select(starts_with("a")) %>%
    select(starts_with("i")) %>%
    select(starts_with("v"))
```

```
## # A tibble: 2,700 x 0
```

This one doesn't work because it looks for the columns that start with A *and* start with I *and* start with V; a column can't start with two different letters, so there are no columns left.

Somebody tried this:

```
chile %>% select(starts_with("a") | starts_with("i") | starts_with("v"))
```

```
## Error:  `starts_with("a") | starts_with("i") | starts_with("v")` must resolve to
integer column positions, not a logical vector
```

This doesn't work, but I think it deserves to succeed, since it shows the right logic: starts with A or starts with I or starts with V. I'm going to give the credit for this one. I think the problem is that `starts_with` produces all the *column numbers* of the columns that match, which is what `select` is expecting; when you put the logical-ORs inside the `select` it turns it into a true or a false, a "logical" rather than an "integer". The right way to handle this is to separate the `starts_with`s by commas, which does do logical-OR rather than the logical-AND I was expecting.

Including any of `age` or `income` or `vote` anywhere in your answer is a fast route to zero points.

Otherwise, a point off an otherwise correct answer for each error, and one point for a decent idea that turns out not to work.

(c) (2 marks) Display the first 15 rows of the data frame.

Solution: `print`:

```
chile %>% print(n=15)
```

```
## # A tibble: 2,700 x 9
##       X1 region population   sex   age education income statusquo  vote
##    <int>  <chr>      <int> <chr> <int>     <chr>  <int>     <dbl> <chr>
##  1     1      N     175000     M    65         P  35000   1.00820     Y
##  2     2      N     175000     M    29        PS   7500  -1.29617     N
##  3     3      N     175000     F    38         P  15000   1.23072     Y
##  4     4      N     175000     F    49         P  35000  -1.03163     N
##  5     5      N     175000     F    23         S  35000  -1.10496     N
##  6     6      N     175000     F    28         P   7500  -1.04685     N
##  7     7      N     175000     M    26        PS  35000  -0.78626     N
##  8     8      N     175000     F    24         S  15000  -1.11348     N
##  9     9      N     175000     F    41         P  15000  -1.01292     U
## 10    10      N     175000     M    41         P  15000  -1.29617     N
## 11    11      N     175000     M    64         P  15000   1.36566     Y
## 12    12      N     175000     M    19         S  35000   1.02791     U
## 13    13      N     175000     F    27        PS     NA   1.43448     Y
## 14    14      N     175000     F    46         S  75000   1.50684     Y
## 15    15      N     175000     M    36        PS  35000   1.49026  <NA>
## # ... with 2,685 more rows
```

Equally good is to use `slice`:

```
chile %>% slice(1:15)
```

```
## # A tibble: 15 x 9
##       X1 region population   sex   age education income statusquo  vote
##    <int>  <chr>      <int> <chr> <int>     <chr>  <int>     <dbl> <chr>
##  1     1      N     175000     M    65         P  35000   1.00820     Y
##  2     2      N     175000     M    29        PS   7500  -1.29617     N
##  3     3      N     175000     F    38         P  15000   1.23072     Y
##  4     4      N     175000     F    49         P  35000  -1.03163     N
##  5     5      N     175000     F    23         S  35000  -1.10496     N
##  6     6      N     175000     F    28         P   7500  -1.04685     N
##  7     7      N     175000     M    26        PS  35000  -0.78626     N
##  8     8      N     175000     F    24         S  15000  -1.11348     N
##  9     9      N     175000     F    41         P  15000  -1.01292     U
## 10    10      N     175000     M    41         P  15000  -1.29617     N
## 11    11      N     175000     M    64         P  15000   1.36566     Y
## 12    12      N     175000     M    19         S  35000   1.02791     U
## 13    13      N     175000     F    27        PS     NA   1.43448     Y
## 14    14      N     175000     F    46         S  75000   1.50684     Y
## 15    15      N     175000     M    36        PS  35000   1.49026  <NA>
```

Or even this:

```
head(chile, 15)
```

```
## # A tibble: 15 x 9
##       X1 region population   sex   age education income statusquo  vote
##    <int>  <chr>      <int> <chr> <int>     <chr>  <int>     <dbl> <chr>
##  1     1      N     175000     M    65         P  35000   1.00820     Y
##  2     2      N     175000     M    29        PS   7500  -1.29617     N
##  3     3      N     175000     F    38         P  15000   1.23072     Y
##  4     4      N     175000     F    49         P  35000  -1.03163     N
##  5     5      N     175000     F    23         S  35000  -1.10496     N
##  6     6      N     175000     F    28         P   7500  -1.04685     N
##  7     7      N     175000     M    26        PS  35000  -0.78626     N
##  8     8      N     175000     F    24         S  15000  -1.11348     N
##  9     9      N     175000     F    41         P  15000  -1.01292     U
## 10    10      N     175000     M    41         P  15000  -1.29617     N
## 11    11      N     175000     M    64         P  15000   1.36566     Y
## 12    12      N     175000     M    19         S  35000   1.02791     U
## 13    13      N     175000     F    27        PS     NA   1.43448     Y
## 14    14      N     175000     F    46         S  75000   1.50684     Y
## 15    15      N     175000     M    36        PS  35000   1.49026  <NA>
```

Use whichever one comes to your mind first. On each of these, you can either use the data frame as the first input, or say the data frame, then a pipe, and then `print` or `slice` or `head` *without* the data frame. (In a pipeline, the first input to a function is whatever came out of the previous pipe.)

This also works, *if you get it right*:

```
chile[1:15,]
```

```
## # A tibble: 15 x 9
##       X1 region population   sex   age education income statusquo  vote
##    <int>  <chr>      <int> <chr> <int>     <chr>  <int>     <dbl> <chr>
##  1     1      N     175000     M    65         P  35000   1.00820     Y
##  2     2      N     175000     M    29        PS   7500  -1.29617     N
##  3     3      N     175000     F    38         P  15000   1.23072     Y
##  4     4      N     175000     F    49         P  35000  -1.03163     N
##  5     5      N     175000     F    23         S  35000  -1.10496     N
##  6     6      N     175000     F    28         P   7500  -1.04685     N
##  7     7      N     175000     M    26        PS  35000  -0.78626     N
##  8     8      N     175000     F    24         S  15000  -1.11348     N
##  9     9      N     175000     F    41         P  15000  -1.01292     U
## 10    10      N     175000     M    41         P  15000  -1.29617     N
## 11    11      N     175000     M    64         P  15000   1.36566     Y
## 12    12      N     175000     M    19         S  35000   1.02791     U
## 13    13      N     175000     F    27        PS     NA   1.43448     Y
## 14    14      N     175000     F    46         S  75000   1.50684     Y
## 15    15      N     175000     M    36        PS  35000   1.49026  <NA>
```

This uses the old-fashioned way to get at rows 1 through 15 and all the columns, which we haven't done in this class.

(d) (2 marks) Display a random sample of 5 respondents.

Solution: `sample_n`:

```
chile %>% sample_n(5)
```

```
## # A tibble: 5 x 9
##      X1 region population   sex   age education income statusquo  vote
##   <int>  <chr>      <int> <chr> <int>     <chr>  <int>     <dbl> <chr>
## 1  1460      S     125000     F    19        PS 200000   1.54808     Y
## 2   435      C     250000     M    59         S  15000  -1.29617     N
## 3   272      N      15000     M    66         P  15000  -0.53327     U
## 4   156      N     250000     F    26        PS 125000   1.41620     Y
## 5   430      C     250000     M    39         P  15000   0.75644     Y
```

This is by far the easiest way. If you want to build it yourself, you can, and the function `sample` plays an important role, but the first roadblock is that you can't sample from a data frame directly:

```
sample(chile,5)
```

```
## # A tibble: 2,700 x 5
##    region  vote   age statusquo income
##     <chr> <chr> <int>     <dbl>  <int>
##  1      N     Y    65   1.00820  35000
##  2      N     N    29  -1.29617   7500
##  3      N     Y    38   1.23072  15000
##  4      N     N    49  -1.03163  35000
##  5      N     N    23  -1.10496  35000
##  6      N     N    28  -1.04685   7500
##  7      N     N    26  -0.78626  35000
##  8      N     N    24  -1.11348  15000
##  9      N     U    41  -1.01292  15000
## 10      N     N    41  -1.29617  15000
## # ... with 2,690 more rows
```

Well, you *can*, but what it does is to choose five random *columns*, not five random rows. (The reason for this is that internally a data frame is a `list` of columns, so that when `sample` looks at what `chile` has to sample from, columns are what it sees.)

`sample` will accept either a vector of numbers, or just a number, in which case it will sample randomly from 1 through that number. So what we have to do here is first to sample five row numbers at random out of the 2700 rows that there are:

```
rows=sample(2700,5)
rows
```

```
## [1] 1051  643 2321   94 1036
```

(sampling is by default without replacement, which is what we want). Then we pick out these rows of `chile` as our sample, which is a job for `slice`, since we are selecting rows by number; feeding `rows` into `slice` would do it:

```
chile %>% slice(rows)
```

```
## # A tibble: 5 x 9
##      X1 region population   sex   age education income statusquo  vote
##   <int>  <chr>      <int> <chr> <int>     <chr>  <int>     <dbl> <chr>
## 1  1051      S      25000     F    22        PS  35000  -0.71250     N
## 2   643      C      15000     M    25         P  15000  -0.27405     U
## 3  2321     SA     250000     F    62         P  35000   1.21020  <NA>
## 4    94      N     125000     F    19         S  35000   1.05618     Y
## 5  1036      S      25000     F    47         P   7500   0.14576     Y
```

If you go this way, one point for making a random selection of rows, and the second point for selecting those rows out of `chile` (though I don't think anyone got that far).

This is what `sample_n` does behind the scenes, however, so we may as well use it (it's already been written and tested).

In case you are wondering, another way to get a sample of "almost" the right size is to select a uniform random number for each row, and then pick the rows for which that random number is less than 5/2700 (since there are 2700 rows and we want to sample 5 of them):

```
chile %>% mutate(u=runif(2700)) %>%
    filter(u<5/2700)
```

```
## # A tibble: 3 x 10
##      X1 region population   sex   age education income statusquo  vote
##   <int>  <chr>      <int> <chr> <int>     <chr>  <int>     <dbl> <chr>
## 1   626      C     175000     M    18         S  15000  -0.79884     N
## 2  1126      S     250000     M    22         S  15000  -0.69824     N
## 3  1573      S     125000     F    31        PS 125000   1.27275  <NA>
## # ... with 1 more variables: u <dbl>
```

The problem with this is that the sample size is random; you'll get "about" five rows, but the actual number of rows could be more or less than that; here it was three. (The number of rows has a binomial distribution with $n = 2700, p = 5/2700$ so that the mean is 5 but there is a nonzero standard deviation.) This is like dividing subjects between a treatment and a control group by tossing a coin; there will be "about the same" number in each group, but the numbers will probably not be exactly even.

(e) (2 marks) Display the respondents who earn more than 70,000 pesos per month.

Solution: Use `filter` to pick out rows matching a condition:

```
chile %>% filter(income>70000)
```

```
## # A tibble: 433 x 9
##       X1 region population   sex   age education income statusquo  vote
##    <int>  <chr>      <int> <chr> <int>     <chr>  <int>     <dbl> <chr>
##  1    14      N     175000     F    46         S  75000   1.50684     Y
##  2    19      N     175000     M    67         P  75000   1.32279     Y
##  3    23      N     175000     M    18         S  75000   1.54808     Y
##  4    32      N     175000     F    37        PS 200000   0.96525     Y
##  5    54      N     125000     M    25         S  75000  -0.42448     A
##  6    56      N     125000     M    34        PS 125000  -1.10088     N
##  7    60      N     125000     M    51         S  75000   1.34999     Y
##  8    61      N     125000     M    33        PS  75000   1.33168     Y
##  9    70      N     125000     M    67        PS  75000   0.23095     A
## 10    73      N     125000     F    38         S  75000   1.06142     Y
## # ... with 423 more rows
```

Strictly, a `print(n=Inf)` is also required to display them all, but I didn't insist on that. Displaying the first ten of them is fine. (The original wording of the question has "display all" in it, but I took the "all" out. This would have made a more interesting question. Perhaps, now, it was too easy.)

(f) (3 marks) Display the twelve lowest `statusquo` scores, along with an indication of how each of these respondents intends to vote.

Solution: Arrange the data frame in order by `statusquo` (ascending), pick out the first 12 rows, then display only `statusquo` and `vote`, so three steps:

```
chile %>% arrange(statusquo) %>%
    slice(1:12) %>%
    select(statusquo, vote)
```

```
## # A tibble: 12 x 2
##    statusquo  vote
##        <dbl> <chr>
##  1  -1.80301     U
##  2  -1.74401  <NA>
##  3  -1.72594     N
##  4  -1.48144  <NA>
##  5  -1.34392     N
##  6  -1.33198     N
##  7  -1.33066     N
##  8  -1.31083  <NA>
##  9  -1.30809     A
## 10  -1.30713     N
## 11  -1.30485     N
## 12  -1.30482  <NA>
```

The `select` and `slice` can be the other way around, since the same first twelve rows are selected either way. You can even do the `select` *first,* since we are sorting by one of the things we are displaying. (If I

had asked for `vote` and `region`, say, of the people with the 12 lowest `statusquo` scores, then you would have to do the `select` after doing the sorting, since otherwise the column you want to sort by would have disappeared.

Since we're displaying the *first* 12 rows, we can also use `print` to accomplish this. Note that the `print` has to come *last*, or else it won't work:

```
chile %>% arrange(statusquo) %>%
    select(statusquo, vote) %>%
    print(n=12)
```

```
## # A tibble: 2,700 x 2
##    statusquo  vote
##        <dbl> <chr>
##  1  -1.80301     U
##  2  -1.74401  <NA>
##  3  -1.72594     N
##  4  -1.48144  <NA>
##  5  -1.34392     N
##  6  -1.33198     N
##  7  -1.33066     N
##  8  -1.31083  <NA>
##  9  -1.30809     A
## 10  -1.30713     N
## 11  -1.30485     N
## 12  -1.30482  <NA>
## # ... with 2,688 more rows
```

That works, but this doesn't:

```
chile %>% arrange(statusquo) %>%
    print(n=12) %>%
    select(statusquo, vote)
```

```
## # A tibble: 2,700 x 9
##       X1 region population   sex   age education income statusquo  vote
##    <int>  <chr>      <int> <chr> <int>     <chr>  <int>     <dbl> <chr>
##  1   788      C     175000     F    64         P     NA  -1.80301     U
##  2  2660      M      25000     M    27         S   7500  -1.74401  <NA>
##  3  2074     SA     250000     F    21        PS  35000  -1.72594     N
##  4  2445     SA     250000     F    42         S  35000  -1.48144  <NA>
##  5  1956     SA     250000     M    27         S   7500  -1.34392     N
##  6   464      C     250000     F    22         S   7500  -1.33198     N
##  7  2681      M      15000     M    34         P  15000  -1.33066     N
##  8  1994     SA     250000     M    66         S  35000  -1.31083  <NA>
##  9  1846     SA     250000     F    25         S  15000  -1.30809     A
## 10  1106      S     250000     F    39         S  15000  -1.30713     N
## 11   693      C      15000     F    30         S  35000  -1.30485     N
## 12  1762     SA     250000     F    23         S   7500  -1.30482  <NA>
## # ... with 2,688 more rows
## # A tibble: 2,700 x 2
##    statusquo  vote
##        <dbl> <chr>
##  1  -1.80301     U
##  2  -1.74401  <NA>
##  3  -1.72594     N
##  4  -1.48144  <NA>
##  5  -1.34392     N
##  6  -1.33198     N
##  7  -1.33066     N
##  8  -1.31083  <NA>
##  9  -1.30809     A
## 10  -1.30713     N
## # ... with 2,690 more rows
```

Well, it sort of works, but it prints out the first 12 rows of the *whole* data frame, then the first *ten* rows with the right columns, so not really.

A negative score on `statusquo` means that the respondent didn't like Pinochet, so it is perhaps not surprising that most of the people who actually expressed a voting intention said they were going to vote "no".

Marking: minus one per error, with the usual 1 (that I tried hard to find for you) if you got *something* right.

Note that `arrange` does *not* have "ascending", so `arrange(statusquo)` will sort the values in ascending order by default, which is what you wanted. You can sort them in descending order if you want, but then you have to pick out the *bottom* 12 values, which requires some careful counting (it's easy to be off by one):

```
chile %>% arrange(desc(statusquo)) %>% select(statusquo, vote) %>%
    slice(2689:2700)
```

```
## # A tibble: 12 x 2
##    statusquo  vote
##        <dbl> <chr>
##  1        NA     U
##  2        NA     Y
##  3        NA  <NA>
##  4        NA     U
##  5        NA     A
##  6        NA     N
##  7        NA     Y
##  8        NA     U
##  9        NA  <NA>
## 10        NA     U
## 11        NA     A
## 12        NA     U
```

This doesn't actually work, because missing values always sort at the end, and there are some. If you got this code, though, I was happy.

What we ought to do is get rid of the missing `statusquo` values first:

```
chile %>% filter(!is.na(statusquo)) %>%
          arrange(desc(statusquo)) %>%
          select(statusquo, vote) %>%
          tail(12)
```

```
## # A tibble: 12 x 2
##    statusquo  vote
##        <dbl> <chr>
##  1  -1.30482  <NA>
##  2  -1.30485     N
##  3  -1.30713     N
##  4  -1.30809     A
##  5  -1.31083  <NA>
##  6  -1.33066     N
##  7  -1.33198     N
##  8  -1.34392     N
##  9  -1.48144  <NA>
## 10  -1.72594     N
## 11  -1.74401  <NA>
## 12  -1.80301     U
```

`tail` is the opposite of `head`, if you know about that: it displays the bottom (here) 12 rows, however many there are altogether. The problem is that we don't know how many rows are left after we've gotten rid of the missings, so going down to 2700 won't work.

The way you used to have to do this, in pre-Tidyverse days, was rather fiddly. R has `sort` (that takes a column to put in order), but this is usually not what you want, because you want to carry along some other columns and display them as well. The thing you use instead is `order`:

```
ord=with(chile,order(statusquo))
ord[1:20]
```

```
##  [1]  788 2660 2074 2445 1956  464 2681 1994 1846 1106  693 1762 1910  108
## [15]   74  503 1876 1944 1849 2641
```

This shows you which rows have the lowest `statusquo` values (20 of them), and then you display the `statusquo` and `vote` of the first twelve of those rows like this:

```
chile[ord[1:12],c(8,9)]
```

```
## # A tibble: 12 x 2
##    statusquo  vote
##        <dbl> <chr>
##  1  -1.80301     U
##  2  -1.74401  <NA>
##  3  -1.72594     N
##  4  -1.48144  <NA>
##  5  -1.34392     N
##  6  -1.33198     N
##  7  -1.33066     N
##  8  -1.31083  <NA>
##  9  -1.30809     A
## 10  -1.30713     N
## 11  -1.30485     N
## 12  -1.30482  <NA>
```

That's a very dense notation. Tidyverse FTW, I say.

In case you were wondering what happened in the end: in the plebiscite, 56% of people voted "no", and perhaps even more surprising, Gen. Pinochet (who had seized power in a military coup in 1973) agreed to step down as President. (There was, I think, some pressure from the international community.) An election was held the following year for a new President.

Post scriptum: all this talk of `statusquo` reminds me that there was (still is) a British rock band of that name. The trouble is, they've been around so long (like so many of the bands from my youth) that a lot of what you find on YouTube shows them much older than I remember. This is about how I remember them (the hair!):

`http://www.dailymotion.com/video/x3aasy`

7. A marketing researcher studied the sales of a product that was introduced ten years ago. The data are shown in Figure 11. The `year` is the number of years since the product was introduced, and the `sales` are annual sales, so that for example the sales for year 0 are the sales for the first year after the product was introduced (between 0 years and 1 year after introduction). The researcher is interested in modelling the relationship between sales and time.

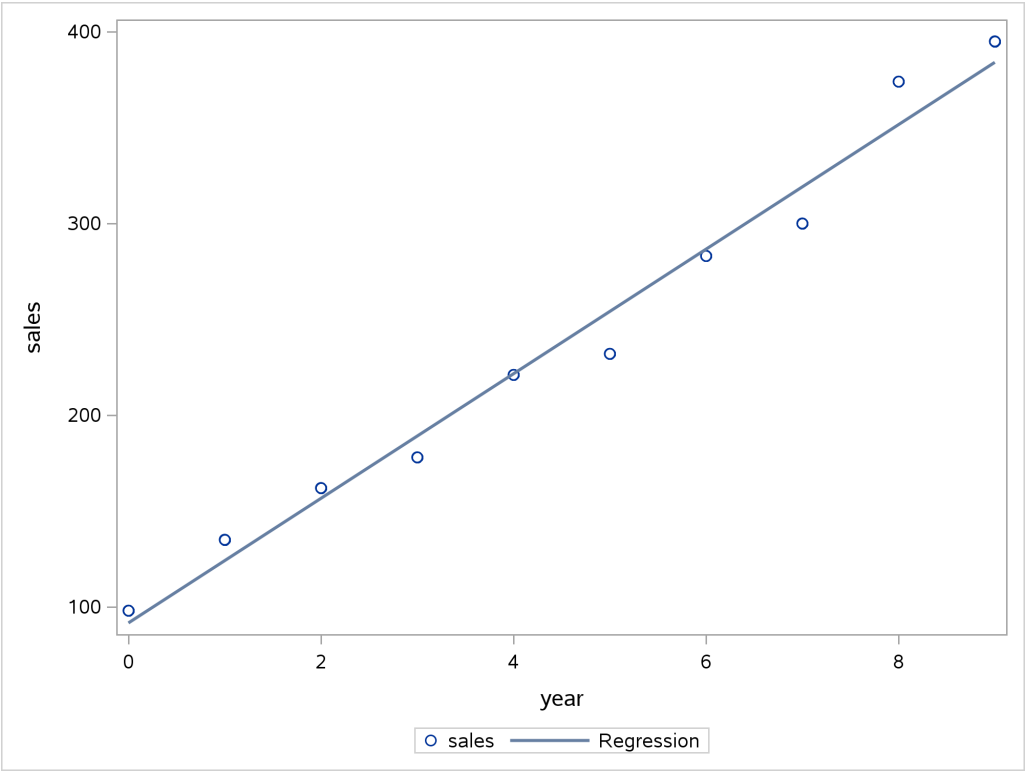
   (a) (3 marks) A scatterplot is shown in Figure 12. Give the SAS code that was used to create this plot.

Solution: Importation first, for me (not for you):

Obs	year	sales
1	0	98
2	1	135
3	2	162
4	3	178
5	4	221
6	5	232
7	6	283
8	7	300
9	8	374
10	9	395

Three lines, these three:

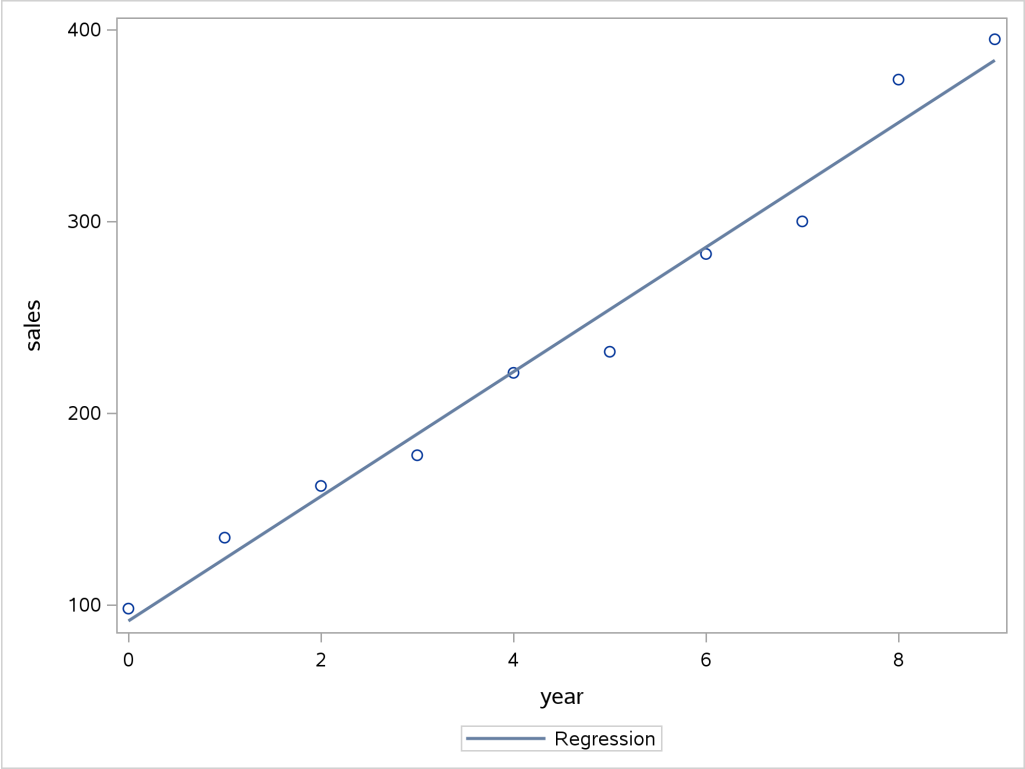
```
proc sgplot;
  scatter x=year y=sales;
  reg x=year y=sales;
```


400
300
200
100
sales
0
2
4
6
8
year
sales
Regression

The second line of code plots the points, the third one the regression line.

I was wondering whether this also works:

```
proc sgplot;
  reg x=year y=sales;
```



It does, and so is also good. The first one is the one I thought of first, so I won't count the "scatter" line as being unnecessary (even though, strictly, it is). To my mind, the plot has two separate things: a scatterplot of the data, and also a regression line, so I would expect *two* things under `proc sgplot`. I think that's also a reasonable assumption for you to make, along the lines of `+geom_point()+geom_smooth()` in R.

Minus one per error. Leaving out the `scatter`, as I did just above, does *not* count as an error. But if you put it in, you have to get it right. Missing the `reg` line is an error. Putting in anything to do with `proc reg` is also an error, since the plot has nothing to do with `proc reg`. (This resembles the "fit plot" that is part of the graphics output from a regression, but it is not the same.) Using `loess` rather than `reg` is an error (the clue is "Regression" next to the line in the legend under the plot). A loess here would probably look at least a bit curved (see later parts of this question).

(b) (2 marks) Figures 13 and 14 show the output from fitting a regression predicting sales from year. Would you say that the regression model fits well or badly? Cite a *number* from the output to support your answer, and explain briefly how it does that.

Solution: The obvious thing to look at is R-squared, which here is 0.9798, very high. So the model fits very well.

Looking at something like the P-value for the slope is not the right thing. This would tell you whether there really is a relationship between year and sales, but not how strong it is. A statistically significant relationship could be quite weak (for example, if you have lots of data).

Just giving me a number with no explanation is not an answer, and is probably not even worth any points. Make a call as to whether you think the regression fits well, and tell me what number you are using, its value, and how it supports your answer. You can describe your R-squared as "high" or "close to 1", or you can say something like "almost all of the variation in sales is explained by (the regression with) year", since that is what R-squared actually is.

Be wary of making your answer too long. This is only two points, so you can reasonably expect that a number and a short explanation is what is needed for full credit. If you write too much (eg. if you over-fill the space), I may take the view that you are writing things down in case there is credit in them, and deduct points for things that are irrelevant to answering the question. I said "a number", so don't give me two numbers. If you lead off with R-squared, I might take the view that the other stuff is supporting evidence (in which case two points), but if you lead off with the P-value, I may take that as your main point, which it should not be (see above): one point.

(c) (3 marks) Look at Figure 14. Is there any evidence that the regression is unsatisfactory? You should assess at least two of the graphs in the Figure. If you think the regression is unsatisfactory, suggest a way of improving it.

Solution: This is a matter of opinion, and I think you could justify either "the regression is satisfactory" or "the regression could be improved", as long as you do it properly.

The distinction depends on how you read the top left plot, of residuals against fitted values. This could be seen as a random collection of points, or as a down-and-up curve. Either is valid. If you think this plot is a curve, this indicates that the actual relationship is a curve, which is what we should fit. (I don't need you to say more at this point. We investigate a transformation below.)

I think the other easiest plot to consider is the normal quantile plot of residuals. This, I think, is completely satisfactory: there are no obvious deviations from the line. So this doesn't change our decision about whether the regression is satisfactory. I could *possibly* be swayed by a claim that there is also a curved pattern here, indicating that the residuals have a skewed distribution, but your argument needs to be good.

Since I won't necessarily agree with what you say here, the marks are not for whether your points agree with mine, but for the quality and clarity of your reasoning. I want to see exactly what graph you are looking at, what you see, and what you conclude from it. If it's a logical thought process clearly expressed, I like it; if it's fuzzy or unclear, I don't. As a result, the marking was mainly by my feeling of how clearly you made your points, and how logically your conclusions came from what you saw. But you can do yourself a favour by remembering to suggest a way to improve the model based on what you saw. (The part down below about Box-Cox was a bit of a giveaway here.)

(d) (3 marks) Figure 15 shows some SAS code and its output. What do you learn from this Figure, and what does it suggest to do next? Explain briefly.

Solution: What `proc transreg` does is to suggest a transformation of the response variable (at least when run like this, which is what we have done). The bottom part of the graph suggests that we should take `lambda` equal to about 0.5: that is, instead of predicting sales, we should predict sales to the power 0.5 (that is, the square root of sales).

If you saw a curve in the residual plot, this should straighten it out.

Roughly speaking, one point each for mentioning $\lambda = 0.5$, "transformation" and "square root of sales". If you say "square root of y", you need to tell me what y is. I thought "the response variable" was close enough, though really you can show me that you know what the response variable is *here* by talking about `sales`. As ever, copying stuff out of your notes will only get you so far. (I saw a lot of my words in answers, which suggests to me that you don't really understand what my words are saying.) *Application* of your knowledge is what I'm after.

(e) (2 marks) Compare your answers to parts (c) and (d). Are they consistent or inconsistent? Explain briefly.

Solution: This depends on what you concluded above; as ever, the key is for you to be logical in your explanation. Either answer, "consistent" or "inconsistent" could be worth full marks, *if supported by appropriate reasons.*

Part (d) says that we should do the regression predicting the square root of sales from year. It also says that "no transformation" is not plausible since `lambda=1` is outside the 95% confidence interval for `lambda` (this goes from about 0.25 to 0.75). Using the square root of sales instead of sales itself is a

non-linear transformation, and so would fit a curve instead of a straight line.

What you say here depends on what you thought in part (c). Probably you will take one of these directions:

- The residual plots indicated that the regression was satisfactory, but the Box-Cox method indicates that we do better to predict the square root of sales. This is inconsistent because suggesting a transformation indicates that the previous regression was *not* satisfactory.
- The residual plot indicated that the actual relationship was curved rather than linear. The Box-Cox transformation indicates that the particular curve given by predicting the square root of sales is a sensible one to use. The two parts are therefore consistent.
- The original regression was good, but there was still scope to improve it, so apparently inconsistent results in (c) and (d) are really consistent. (See discussion of this below.)

What I was guided by was your conclusion from (c) about whether the regression was OK or not (the latter implying that something needed to be done about it) and your conclusion from (d) (about some kind of transformation being necessary, or not, if that's what you thought, or some other kind of comment about the regression being acceptable or not). Then, if you made some kind of relevant comment about how your (c) and (d) said the same or different things, then you were good for two points here. I tried very hard to give you two points if I could. If I didn't, I found something in your *reasoning* that didn't make sense.

There is another exam strategy thing here: if you gave *no* answer to (d), then you couldn't receive any points here either (sorry). In an exam like this, giving a wrong answer is *always* better than giving no answer, because there might be some points in a wrong answer, but a blank answer is an automatic zero. (The only thing writing something costs you is the time to write it down.) Thus, if you said *something* in (d), no matter how off-base (and even if it didn't get any points in (d)), I am obliged to check it for consistency with your conclusion from (c) and grade you in (e) accordingly. In that case, you can still get something for (e). Maybe not full marks, particularly if you have made (e) easier to answer by saying something absurd in (d), but something.

The only exam situation where giving a wrong answer is worse than giving no answer is a multiple-choice exam where points are deducted for wrong answers. Even then, the deduction is usually structured so that if you have any idea at all, it's better to guess and risk being wrong than not to guess at all.

To explore this one further: what is perhaps surprising is that the original R-squared, 0.98, is very high and there doesn't seem to be much room to improve it. Is the square-root-transformed sales really a better response variable to use? Perhaps we should investigate:

```
proc import
  datafile='/home/ken/sales.txt'
  out=salesdata
  dbms=dlm
  replace;
  getnames=yes;
  delimiter=' ';

data salesdata2;
  set salesdata;
  rtsales=sqrt(sales);

proc reg;
  model rtsales=year;
```

The REG Procedure
Model: MODEL1
Dependent Variable: rtsales

Number of Observations Read 10
Number of Observations Used 10

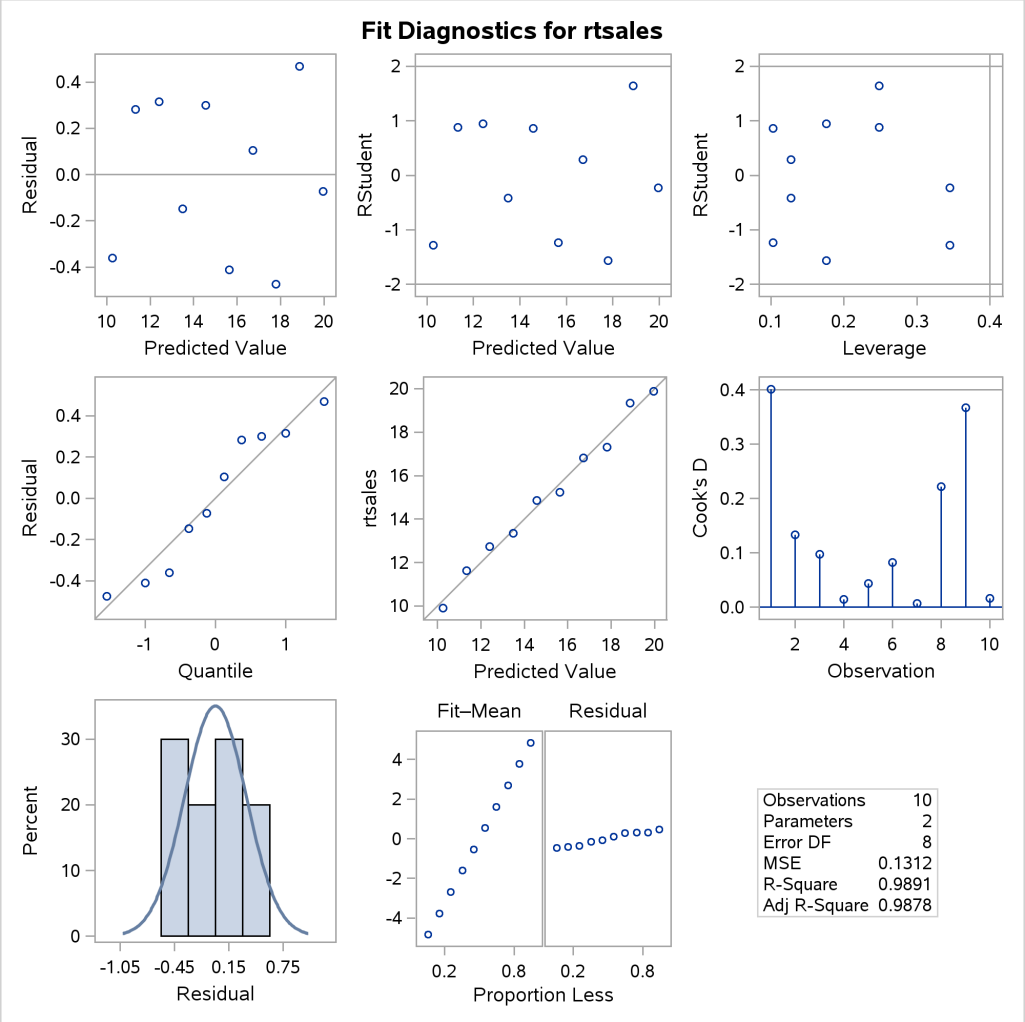
Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	95.56833	95.56833	728.37	<.0001
Error	8	1.04967	0.13121		
Corrected Total	9	96.61800			

Root MSE	0.36223	R-Square	0.9891
Dependent Mean	15.10424	Adj R-Sq	0.9878
Coeff Var	2.39818		

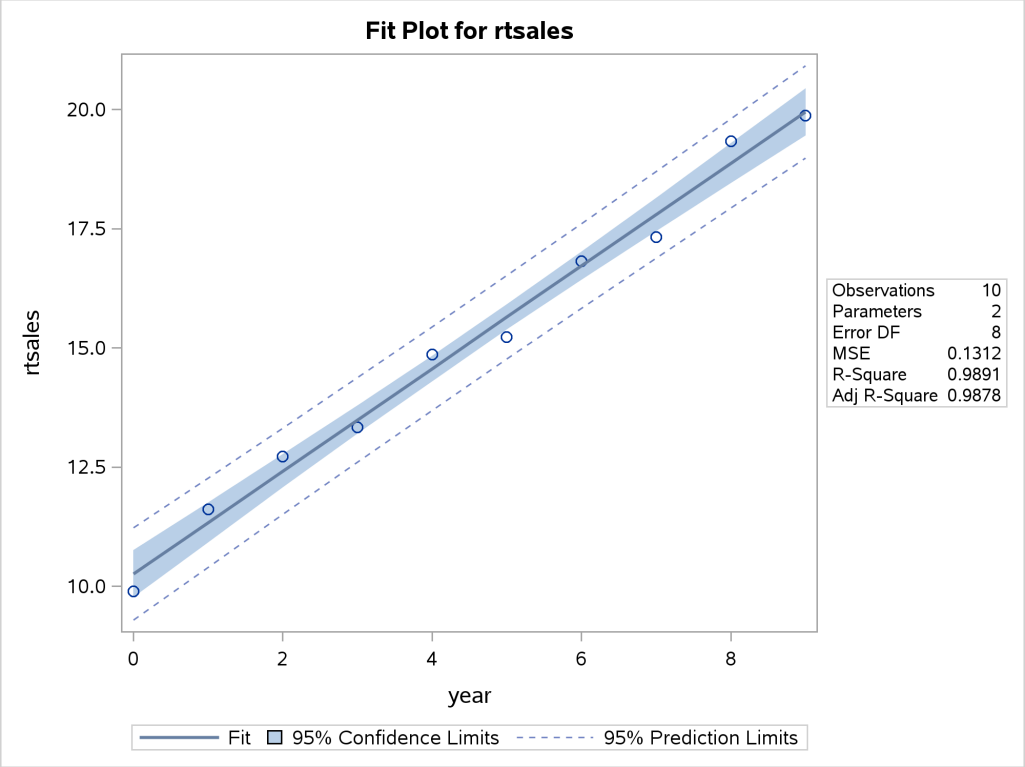
Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	10.26093	0.21290	48.20	<.0001
year	1	1.07629	0.03988	26.99	<.0001


Fit Diagnostics for rtsales
Residual
Predicted Value
RStudent
Predicted Value
RStudent
Leverage
Residual
Quantile
rtsales
Predicted Value
Cook's D
Observation
Percent
Residual
Fit–Mean
Residual
Proportion Less
Observations 10
Parameters 2
Error DF 8
MSE 0.1312
R-Square 0.9891
Adj R-Square 0.9878

Well, R-squared has gone up to 99%, and the plot of residuals against fitted values definitely looks random now. (Even if you thought it was acceptable before, I would say it's better now.) Also, the normal quantile plot of residuals is still normal enough. I'd say, therefore, that even though there wasn't much room to improve before, we have nonetheless managed to improve things with the transformation.

One of the other graphs is the "fit plot", which is worth looking at now:


Fit Plot for rtsales
rtsales
20.0
17.5
15.0
12.5
10.0
0
2
4
6
8
year
Observations 10
Parameters 2
Error DF 8
MSE 0.1312
R-Square 0.9891
Adj R-Square 0.9878
Fit
95% Confidence Limits
95% Prediction Limits

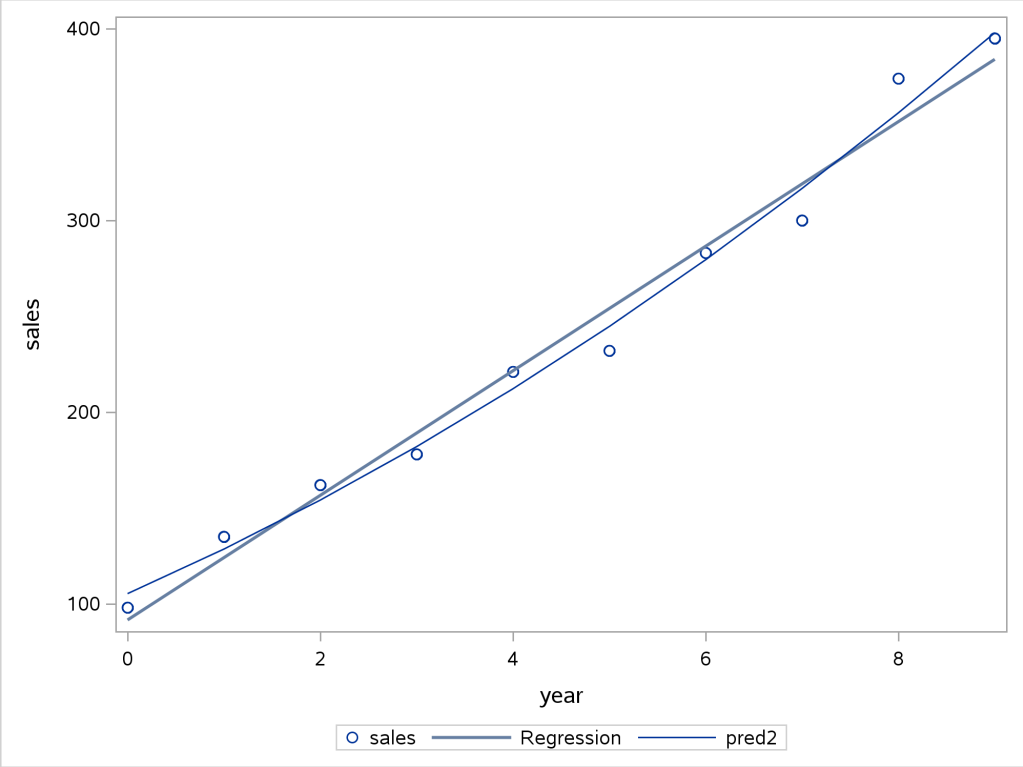
This shows that (i) the relationship with square root of sales is very straight, and (ii) that the deviations from straightness are more random than they were before. (Previously, there was a tiny curve if you looked closely enough.)

This wasn't quite the right plot. I want the predicted *sales* from the regression that uses *square root* of sales as response. This means saving the predictions from the regression and then building our own plot:

```
proc reg;
  model rtsales=year;
  output out=xx p=predrtsales;

data xx2;
  set xx;
  pred2=predrtsales*predrtsales;


proc sgplot;
  scatter x=year y=sales;
  reg x=year y=sales;
  series x=year y=pred2;
```



OK, that's what I wanted. This is like the plot for the windmill example from class, only this time it is less clear: the curve is maybe a teeny bit better than the line, but it is hard to choose.

It was rather a lot of work to get that.

- First we had to save the predicted square-root-sales by redoing the regression with an `output` line to save the predictions. This creates an output data set with all the original variables plus the "predicted values" (which is what the `p=` saves). In the old days you had to do this to get a residual plot: save the predicted values and residuals, and then make a plot of them.
- Next we had to get the predicted actual sales (rather than the predicted square-root-sales). This means undoing the transformation: squaring instead of square-rooting. To create a new variable with those values, we had to create a new data set, bringing in all the values from the old one (the output data set from the regression).
- Last, we made the plot. This is a scatterplot: first the original data, then the regression (straight) line, and finally the predicted values from the transformed regression. I plotted these using `series`: this joins the predictions by lines, without plotting the points themselves (by default), which is what I wanted. (As in the examples in class, `pred2` looks like a curve, but each prediction is actually joined to the next one by a straight line (segment).)

That was a lot of work to get that plot, but it's nice to see because it shows that the curve does fit a little bit better (maybe), and also that the curve is not very curved: a little bit lower in the middle and a little bit higher at the ends. You could say that going to the trouble of finding and doing the transformation and then making sense of the results was a lot of work for not much gain, but the Box-Cox plot shows that this really is better than not doing the transformation, and maybe this plot shows it too.

8. The "performance IQ" or PIQ is one aspect of intelligence. This is understood to depend on brain size (measured from MRI scans) and body size (height, in inches; weight, in pounds). Data on these variables were collected for a sample of 38 individuals. Part of the dataset is shown in Figure 16. **Use $\alpha = 0.01$ for all the hypothesis tests in this question.**

   (a) (2 marks) A regression predicting performance IQ from the three explanatory variables is shown in Figure 17. Even at $\alpha = 0.01$, explain briefly why this output *does not* permit us to remove both `Height` and `Weight` from this regression.

Solution: The tests on the `summary` output from the regression are for the removal of *just that one* explanatory variable. (That's what I wanted for the two points). So what I should do is to remove, say, `Weight`, because its P-value is largest, fit again, and *then* make a decision about `Height` (which might then have become significant at $\alpha = 0.01$).

This is motivating Figures 18 and 19, which we get to in a moment.

There are other ways to say this. A common (good) one is "if we remove `Weight`, we don't know what will happen to the P-value of `Height` (it may become significant)." Or you can talk about multicollinearity (correlated explanatory variables), which seems especially likely here since a person who is big on one measurement is likely to be big on the others as well. Multicollinearity is the biggest driver of big changes in P-values as variables are added or removed.

I thought "we have to remove explanatory variables one at a time" was a weaker answer (1 point) if you didn't say anything about why that is (or what effect it might have if you don't). Likewise, I was prepared to give one point for a tangentially relevant comment that didn't really answer the question.

Comments about R-squared don't help here, especially as taking *anything* out, even `Weight`, must mathematically lower the R-squared. You can make a point about *adjusted* R-squared, though, since that will tell you whether removing anything was a good idea or not (adjusted R-squared can go up or down).

Apart from a claim that `Height` is significant (not at $\alpha = 0.01$!), I was prepared to overlook things if a sensible answer was in there somewhere.

   (b) (2 marks) Give R code to fit the model of Figure 18, *without* using `lm` and without a `data=`. You may assume that the model `piq.1` shown in Figure 17 has already been fitted. (A correct answer can be obtained in one line of code.)

Solution:

The key is to use `update`, which requires starting from model `piq.1` and describing in code how it changes: "remove `Height` and `Weight` from it":

```
piq.2=update(piq.1,.~.-Height-Weight)
```

(that's the one line of code), and to check:

```
summary(piq.2)
```

```
##
## Call:
## lm(formula = PIQ ~ Brain, data = perf)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -40.077 -17.508  -2.095  17.097  41.571
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.6519    43.7118   0.106   0.9158
## Brain         1.1766     0.4806   2.448   0.0194 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.21 on 36 degrees of freedom
## Multiple R-squared:  0.1427,Adjusted R-squared:  0.1189
## F-statistic: 5.994 on 1 and 36 DF,  p-value: 0.01935
```

If you want to do this in two stages, thus:

```
piq.1a=update(piq.1,.~.-Height)
piq.2=update(piq.1a,.~.-Weight)
piq.2
```

```
##
## Call:
## lm(formula = PIQ ~ Brain, data = perf)
##
## Coefficients:
## (Intercept)        Brain
##       4.652        1.177
```

that's fine by me, since it gets the job done. Two rather inelegant points, but two points nonetheless. (Just `print`ing the fitted model object displays the intercept and slope, so we can see it's the same.)

`update` plus something plausible but incorrect, or with any errors in the code, is worth one point. (In fact, `update` plus anything not obviously silly was one point, so that one point covered a wide range. If I had made this out of 3, I would have distinguished more finely between 1 and 2.)

I had some sympathy with `drop1` and `step` (or with the one student who suggested `drop2`, which doesn't exist!) since these will at least be a way of working towards the desired model, even if they are not guaranteed to get there. One point for these. Most likely zero for anything else.

(c) (4 marks) A second regression is shown in Figure 18, and another test is shown in Figure 19. What do you conclude, at $\alpha = 0.01$, from the test in Figure 19, in the context of the data? What does that tell you about the appropriate model to use to predict performance IQ? Explain briefly.

Solution: The `anova` is comparing the fit of two models: the one with all three explanatory variables, and the one with just `Brain` size. The null hypothesis is that the two models fit equally well, and the alternative is that the bigger model, the one with `Height` and `Weight` as well, is better.

The P-value is 0.036, so at $\alpha = 0.01$, the null hypothesis is *not* rejected. That means that the two models fit equally well, and therefore (this is the 4th point) that we should go with the model containing only `Brain` because it is simpler.

I thought this was going to take a long time to mark, but I had a good marking scheme, so it was not too bad. One point for saying or implying each of these:

- the P-value is 0.036. (That was an easy first point.)
- don't reject the null hypothesis at $\alpha = 0.01$ (a pretty easy second point)
- the two models `piq.1` and `piq.2` fit equally well
- therefore go with the smaller model `piq.2` because it is simpler. (The logic is that you have no proof, by way of a small enough P-value, of the necessity of the bigger model with `Height` and `Weight`, so you conclude that you *don't* need it. The burden of proof is on the bigger model to justify that it helps.) I wanted you to say something beyond "the smaller model" to convince me that you knew which model we were talking about: the one with just `Brain` in it.

I made $\alpha = 0.01$ in this question to make it more interesting, and perhaps, to test the flexibility of your thinking: are you aware enough to compare the P-values to the α you are using, rather than automatically with 0.05?

If you incorrectly chose to reject the null hypothesis but correctly followed the logic through (thus concluding that you *did* need the bigger model with `Height` and `Weight` in it), I gave you two points. There were a few people who ended up in the right place by making two errors. I didn't have much sympathy there.

Concluding correctly that there was nothing to choose between the two models, and then deciding between them by R-squared, is exactly the *wrong* way to go. What the non-significant test tells you is that the decrease in R-squared between `piq.1` and `piq.2` is *just chance*, not meaningful at all, so that there is no justification at $\alpha = 0.01$ for using the bigger model `piq.1`. What happens is that removing variables from a regression *always* decreases R-squared, *even if those variables are useless*. The appropriate comparison between models of different sizes is *adjusted* R-squared (which is what that is for). This tends to favour keeping marginal variables. It decreases here between `piq.1` and `piq.2`, so that adjusted R-squared and Figure 19 are pointing to opposite conclusions. (I would have some sympathy for someone who chooses between `piq.1` and `piq.2` on this basis.) Another way of comparing models of different sizes is something like AIC (if you have run into that before): that does a proper assessment of the fit of a model, taking into account how complicated it is. A more complicated model has to fit better to be deemed worth using. This is what `step` uses, by default.

You see that we *were* justified in removing both `Height` and `Weight`, in the end, but Figure 17 was not the correct justification for doing so: we had to fit both models and go the `anova` way. It could have been the case that by removing `Weight`, the P-value for `Height` dropped below 0.01. If that had happened, we would have been alerted by the `anova` (its P-value would have dropped below 0.01 as well), but by blindly removing both variables on the basis of Figure 17, we would have missed it. As it happened, we would have gotten lucky here. But we won't always.

9. In 2008, a major college in the US was monitoring salaries to see whether there were systematic differences in salary between male and female faculty members. Part of the data set is shown in Figure 20. The variables shown are:

   - `rank`: Assistant Professor (lowest), Associate Professor, Professor (highest)
   - `discipline`: classified as `A` ("theoretical") or `B` ("applied")
   - `yrs.since.phd`: number of years since Ph. D. earned
   - `yrs.service`: number of years since hired (at this college)
   - `sex`: Female or Male
   - `salary`: in (US) dollars (response).

   (a) (2 marks) Figure 21 shows the results of a t-test for comparing salaries of male and female faculty members. The P-value is very small. However, despite this, this t-test is not the strongest evidence that females are being discriminated against. Explain briefly why not. (You may assume that the distributions of salaries within each gender are sufficiently close to being normal.)

Solution: The t-test shows *only* that the mean salary for males is larger than that of females. It says nothing about *why*. For example, the females could be different from males in other ways as well, such as not having achieved as high a rank, or being in departments that typically pay less, or having received their Ph. D.s more recently, or anything like that. This would not be evidence of discrimination.

What constitutes evidence of discrimination would be finding that, with a male and a female faculty member who are the same on all the other variables, the male one gets paid more. Or, to say it in a more multiple-regression way, if you find that males get paid more than females *after allowing for other variables* that might also be different for males and females, *then* you have evidence of discrimination, but not otherwise.

I added the last sentence to the question to prevent you claiming that there's something wrong with the residuals that would invalidate the conclusions here. I wanted to make sure that you engaged with the issues I raised above.

The t-test in Figure 21 is one-sided because it is trying to prove that females get paid *less* than males, and this is the correct side because `Female` is before `Male` alphabetically. The unfortunate part is that you don't get a nice interpretable confidence interval, but I decided that doing the test two-sided, though it would give us a CI, would confuse the interpretation of the test: we were not looking for evidence of *any* difference, but of the particular one where the females got paid less. The "interpretation" of the one-sided CI would be that the mean salary of females is at least \$6,600 less than males.

There are, as a number of people noticed, a lot fewer females than males. This in itself is not a problem. The t-test can handle differing sample sizes with no problem, and since means are being compared, the imbalance in sample sizes is being taken care of. (This might be evidence of systematic differences in *hiring*, which is another issue, not our focus here.)

   (b) (3 marks) Look at Figure 22. Some of the explanatory variables here are categorical rather than quantitative. Explain briefly why the two coefficient estimates that start with `rank` have values that make sense, given what the data represents.

Solution: There are three ranks, `AsstProf`, `AssocProf` and `Prof`. These are in ascending order, the first one being the most junior and the last being the most senior. So we'd expect salaries for the faculty members in the first group to be lowest and in the last group to be highest, all else equal. `AssocProf` is the first one alphabetically, so it is the baseline. It doesn't appear in Figure 22, and its coefficient is

zero. `AsstProf` is junior to `AssocProf`, so we'd expect the predicted salary to be less; this is the case because the coefficient for `rankAsstProf` is about $-\$13,000$, negative.

With that in mind, we'd expect the `rankProf` coefficient to be positive, since we expect Professors to earn more, and so it proves: it's about $\$32,000$, very clearly positive.

Another way of looking at that is to say that, relative to the middle-rank Associate Professors, Assistant Professors earn about $13,000 less and Professors about $32,000 more, even if everything else is the same.

Of course, everything else is *not* likely to be the same; people promoted to Professor will typically have done a lot of high-quality research, and to do that their Ph. D. was probably earned a long time ago. Likewise, a Professor may well have a lot of years of service, since (if they started their career at this college) they would probably have been hired at the Assistant Professor level and had to work their way up. So the actual differences in salary between ranks might be bigger than these values, because a higher-ranking faculty member might be higher on the other variables as well and *they* are associated with a higher salary in addition.

One mark for saying something (implicitly or explicitly) about the Associate Professors being the baseline, and one each for saying that Assistant Professors typically earn less than them and Professors earn more.

I don't want to know about P-values. I want to know about those values $-13,000$ and 32,000 and what they mean.

There were a lot of answers that didn't really get at this, but if something sensible was said (for example that higher-ranking people get paid more) without providing the evidence for it, I found 1 point for it.

(c) (3 marks) What precisely does Figure 22 tell you about the evidence for gender discrimination in terms of salary? Explain briefly.

Solution: The P-value of 0.216 is not small, which means that, *allowing for all the other variables,* there is *no* significant difference between male and female faculty members' salaries, so that there is *no* evidence of discrimination. Another way to say this is that there is no evidence of a gender difference in salary, other things being equal (that is, comparing males and females with the same values on the other variables).

I want to see words like "allowing for the effects of the other variables" or "all else being equal" or "`sex` has nothing to add in predicting salary over and above the other variables", something like that. The "precisely" in the question is a hint that you need to say something beyond "there is no significant `sex` effect", which in itself makes no sense given what you observed in (a).

So my marking scale here is something like:

- Proper interpretation of the slope of `SexMale` (that the salary for males is a bit higher than for females, all else being equal): 1 point.
- Proper interpretation of the P-value for `SexMale` (it is not significant, so there is no evidence of gender discrimination in salary): 2 points.
- Adding something that suggests that other factors are responsible for the variation in salary, or that gender has no effect, all else being equal: 3 points. This can be as simple as "gender is not a factor in predicting salary", implying that other things are. Or you can remark that gender is in fact the *least* significant predictor of salary. I would prefer that you say something like "gender is the only non-significant predictor of salary", but anything of this flavour will net you the third point.

This all means that the significant difference we saw in part (a) was actually explained by other differences between males and females, and so the explanation is *not* that males get paid more because they are male; they get paid more because they are of higher professorial rank, typically in an applied rather than theoretical discipline, have more years since their Ph. D., or something like that.

We can explore these, but first I need to read in the data:

```
salaries=read_csv("salaries.csv")
```

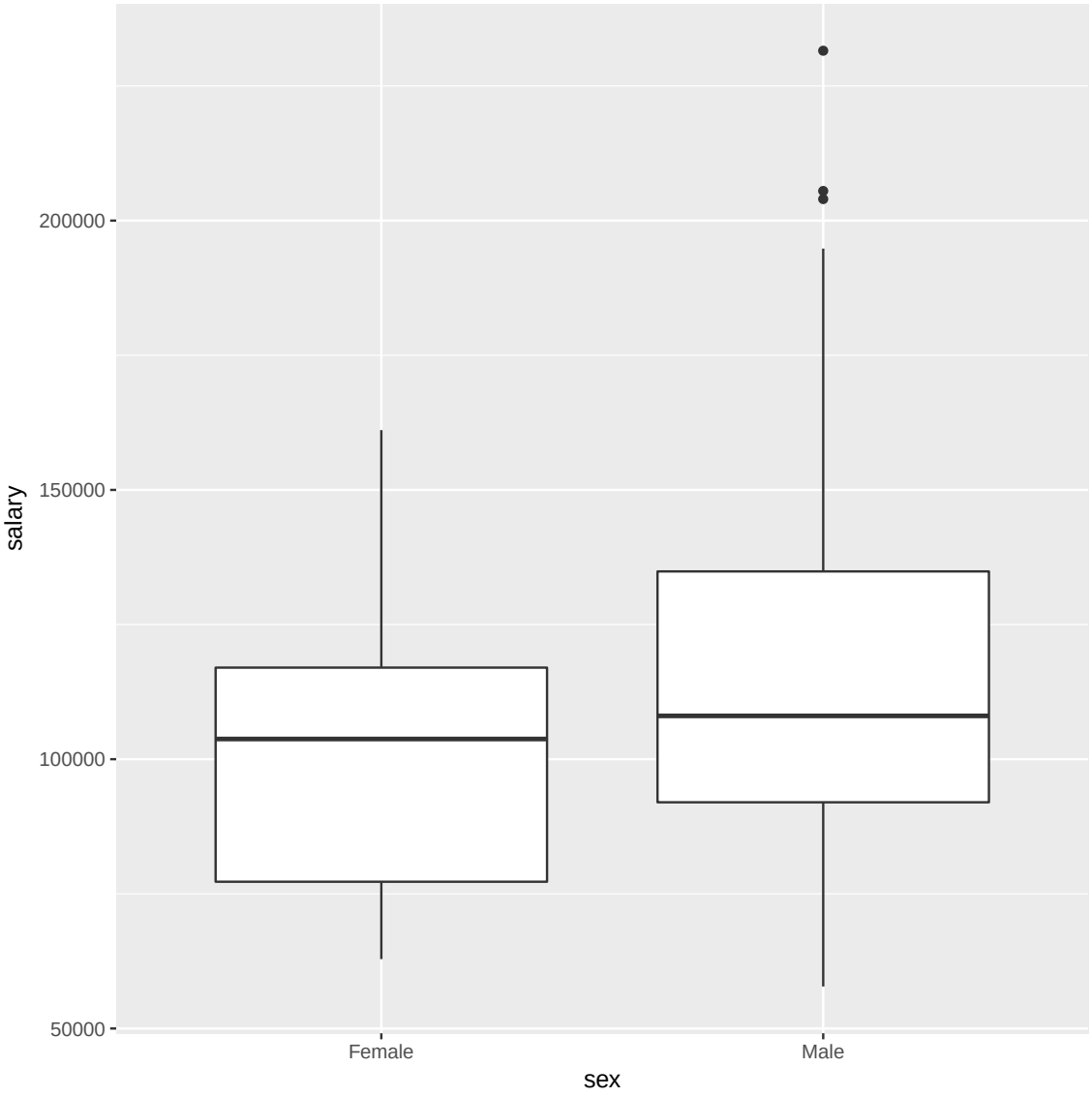
```
## Parsed with column specification:
## cols(
##   rank = col_character(),
##   discipline = col_character(),
##   yrs.since.phd = col_integer(),
##   yrs.service = col_integer(),
##   sex = col_character(),
##   salary = col_integer()
## )
```

```
salaries
```

```
## # A tibble: 397 x 6
##         rank discipline yrs.since.phd yrs.service    sex salary
##        <chr>      <chr>         <int>       <int>  <chr>  <int>
##  1      Prof          B            19          18   Male 139750
##  2      Prof          B            20          16   Male 173200
##  3  AsstProf          B             4           3   Male  79750
##  4      Prof          B            45          39   Male 115000
##  5      Prof          B            40          41   Male 141500
##  6 AssocProf          B             6           6   Male  97000
##  7      Prof          B            30          23   Male 175000
##  8      Prof          B            45          45   Male 147765
##  9      Prof          B            21          20   Male 119250
## 10      Prof          B            18          18 Female 129000
## # ... with 387 more rows
```

We want to consider graphs of `sex` along with the other variables. When those variables are quantitative, this means a boxplot. For example, `salary`:

```
ggplot(salaries,aes(x=sex,y=salary))+geom_boxplot()
```

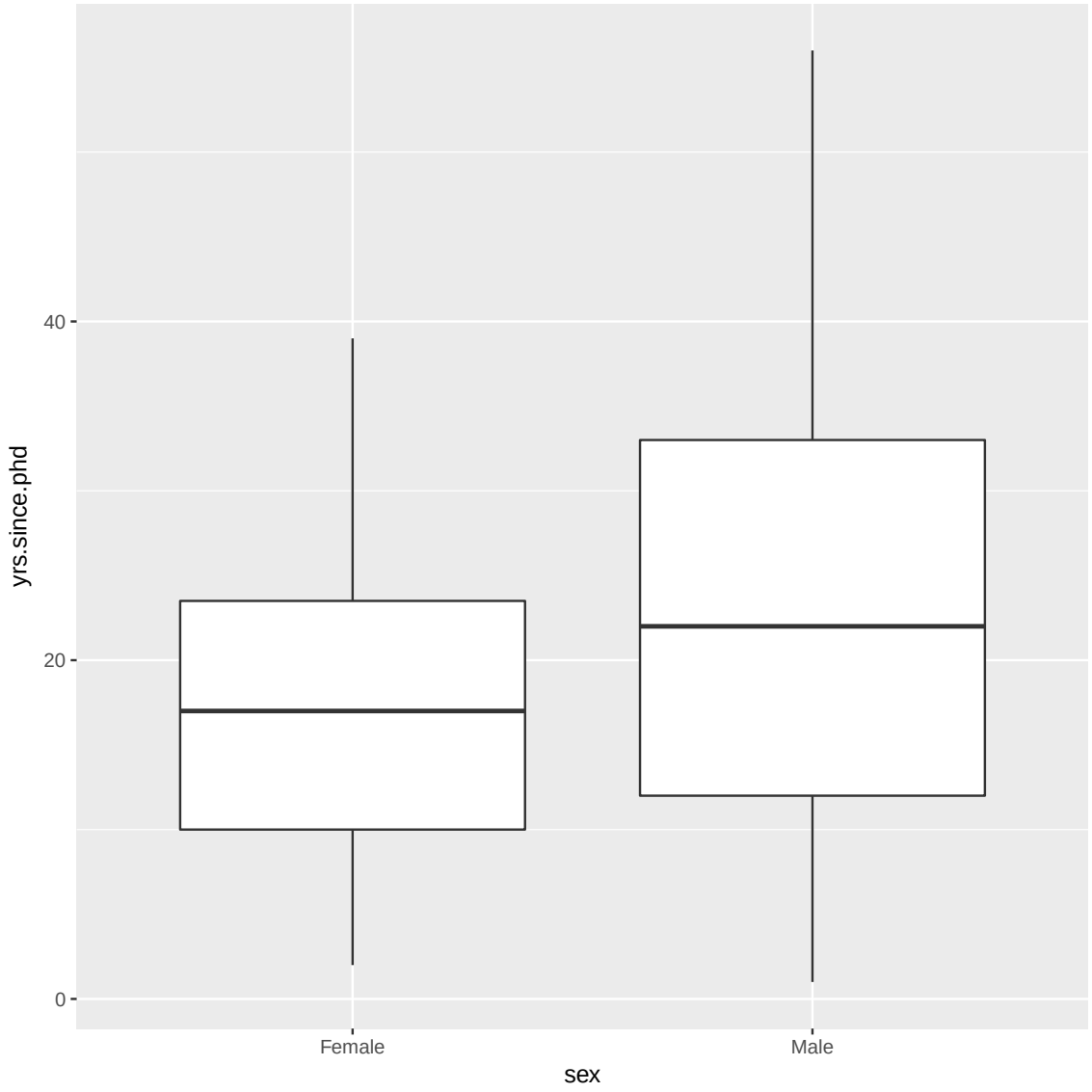


I'm not too worried about those outliers since we have a huge sample size, $n = 397$.

The P-value for the two-sample t-test came out very small, which is perhaps surprising, considering that the medians don't look very different (and we have a lot of variability), but the sample size is driving that as well. The outliers are pulling the male mean salary up a little, but not too much, since they are only three values out of nearly 400.

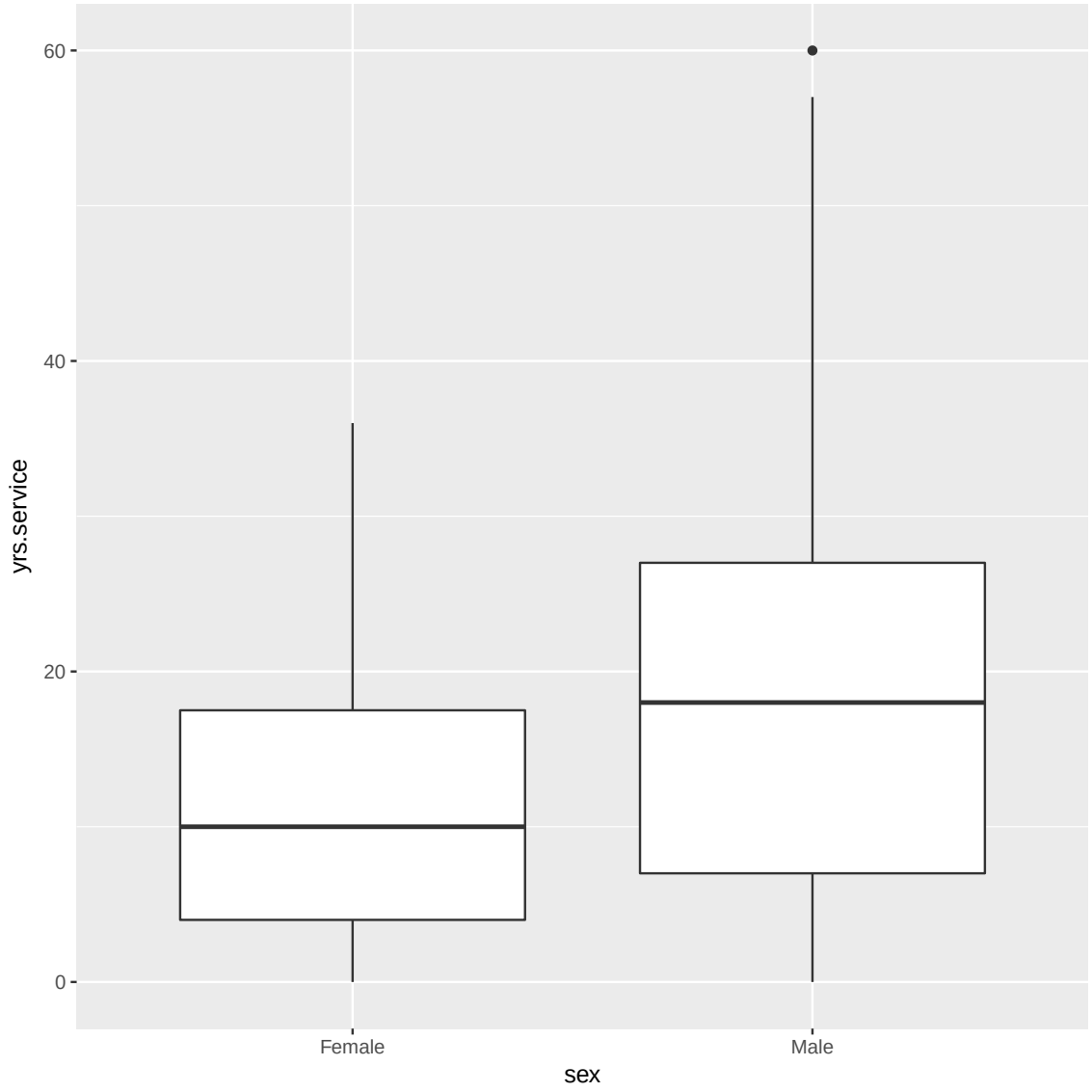
All right, why are the male and female mean salaries different? In the light of what we've seen, we should look at pictures of `sex` and the other variables. The two quantitative variables are easiest since they're boxplots again:

```
ggplot(salaries,aes(x=sex,y=yrs.since.phd))+geom_boxplot()
```



and

```
ggplot(salaries,aes(x=sex,y=yrs.service))+geom_boxplot()
```



The females have noticeably fewer years since Ph. D. and years of service:

```
salaries %>% select(starts_with("yrs"),salary) %>% cor()
```

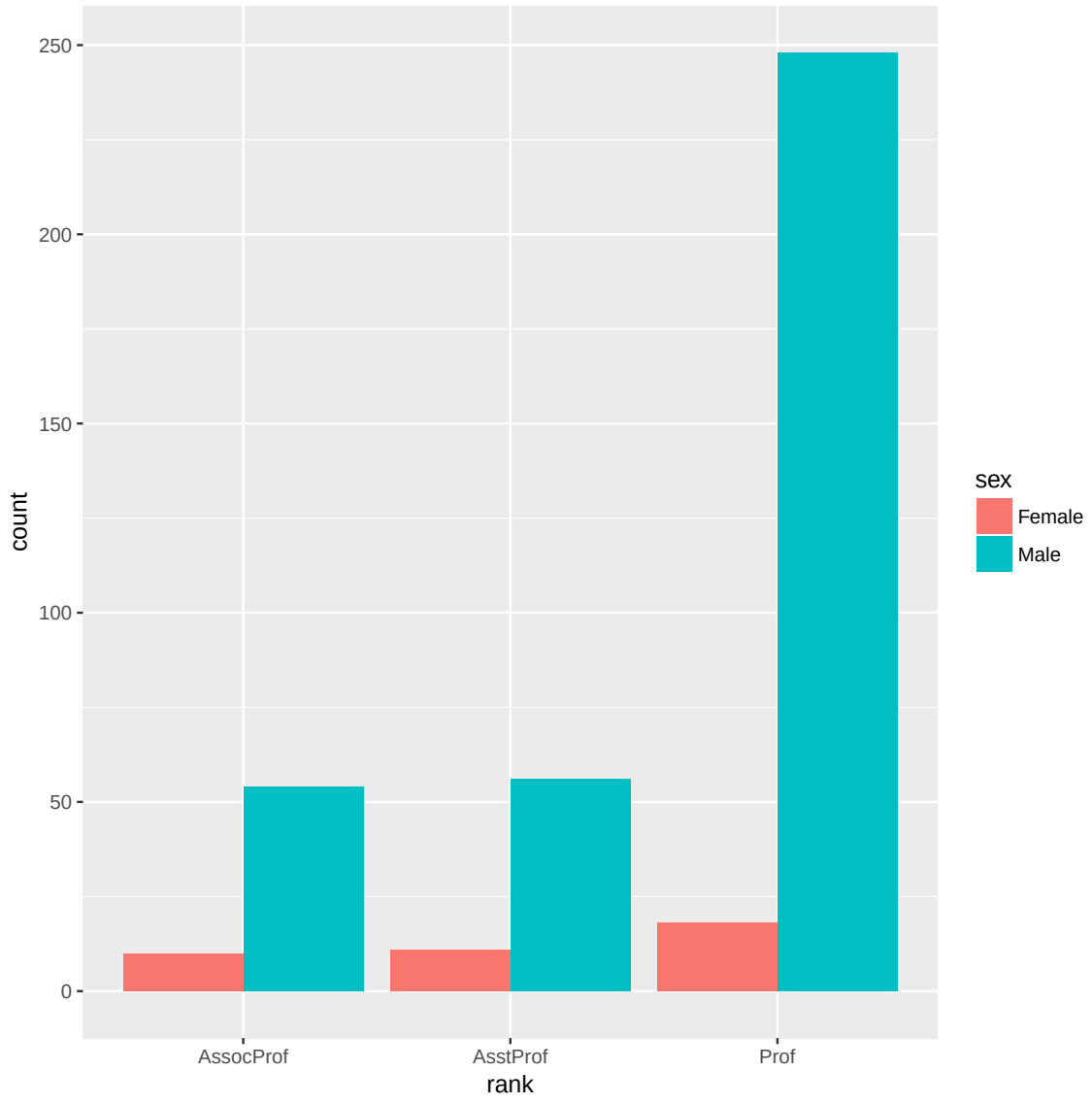
```
##               yrs.since.phd yrs.service    salary
## yrs.since.phd     1.0000000   0.9096491 0.4192311
## yrs.service       0.9096491   1.0000000 0.3347447
## salary            0.4192311   0.3347447 1.0000000
```

and these both have positive correlation with salary. This is something that cannot be changed quickly; even if the college hires a lot of females now, the effect won't be seen until those old males retire.

Now, `sex` and the other variables (`rank` and `discipline`) are all categorical, and the right thing to do with pairs of categorical variables is a grouped bar chart. The last thing we want to decide is which variable is `x` and which is `fill`; I think this is just like the Australian athletes one from the lecture notes in that we want to compare gender within each group made by the other thing, so `sex` should be `fill`

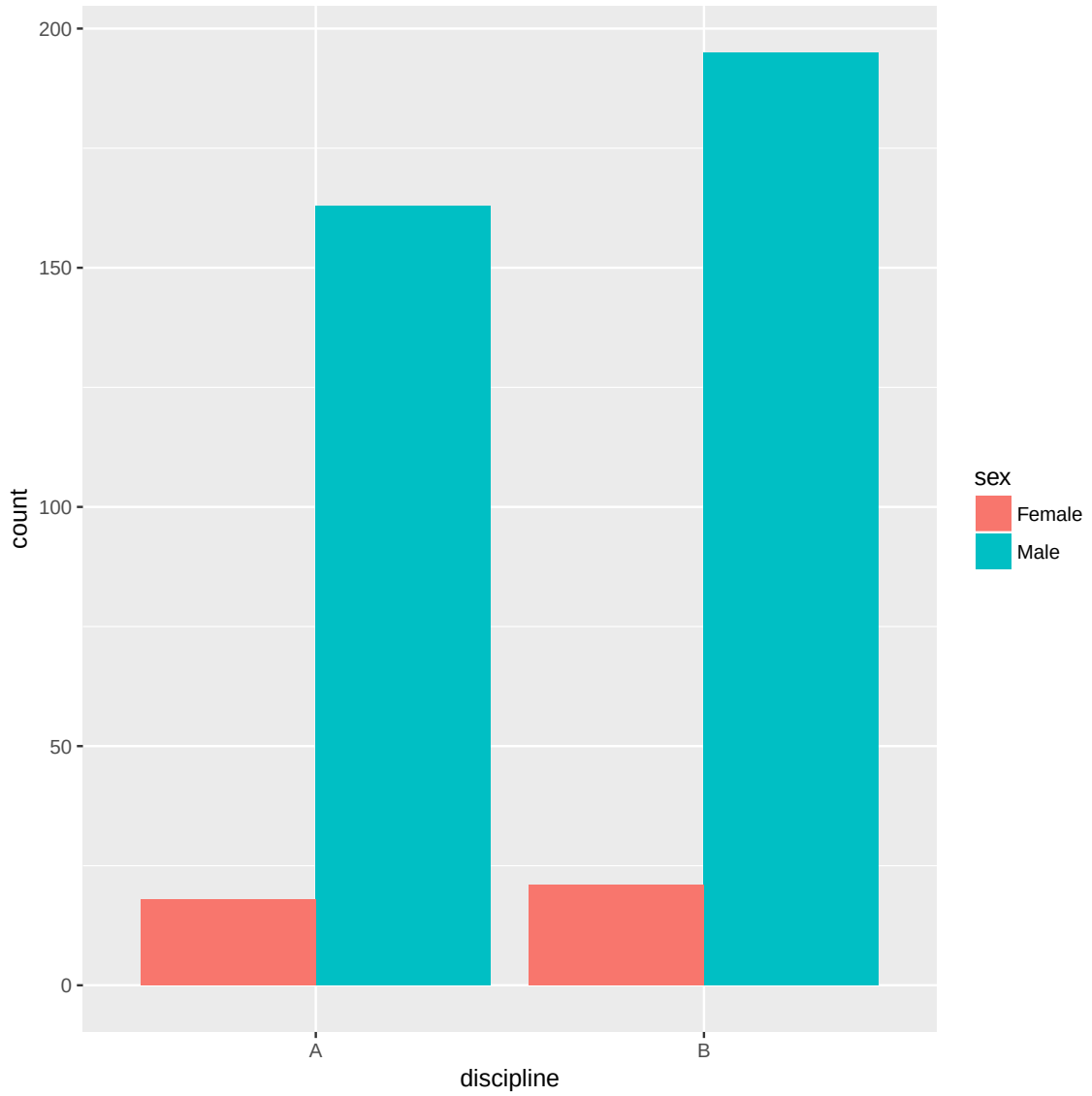
both times:

```
ggplot(salaries,aes(x=rank,fill=sex))+
  geom_bar(position="dodge")
```



Perhaps I should have compared proportions rather than actual numbers on this one, since there are more males at each rank, but there are overwhelmingly more males and fewer females (proportion-wise) at the Professor rank, and these are the people who really bring in the big bucks.

```
ggplot(salaries,aes(x=discipline,fill=sex))+
  geom_bar(position="dodge")
```



Finally, there are again more males than females everywhere, but there don't seem to be proportionately more males in either discipline: the proportion of females is distressingly small in both cases, but it doesn't seem to differ much.

So what these pictures show is that males tend to outrank females at things that are associated with higher salaries, but there is no evidence of discrimination if you compare like with like.

10. A staff analyst at a manufacturer of microcomputer components has compiled monthly data for the past 16 months. The variables are as follows:

    - `month` and `year`: the month and year for which the data was collected. The month and the year are both numbers.
    - `comp`: the value of the manufacturer's components sold (in thousands of dollars)
    - `indproc`: the total value of all industry production that uses these components (in millions of dollars).

    The file has been read into a SAS data set called `micro`, with `month` and `year` stored as numbers. The data set as read in is displayed in Figure 23.

    (a) (3 marks) Give code to create a new data set that contains the month and year turned into an actual SAS date called `date`. The dates you create should be the first day of the appropriate month, and the columns `month` and `year` of the read-in data set should be removed.

Solution: I have to read in the original data first (which you don't have to):

```
proc import
  datafile='/home/ken/micro.txt'
  out=micro
  dbms=dlm
  replace;
  getnames=yes;
  delimiter=' ';

proc print;
```

Obs	month	year	comp	indproc
1	2	2012	102.9	2.052
2	3	2012	101.5	2.026
3	4	2012	100.8	2.002
4	5	2012	98	1.949
5	6	2012	97.3	1.942
6	7	2012	93.5	1.887
7	8	2012	97.5	1.986
8	9	2012	102.2	2.053
9	10	2012	105	2.102
10	11	2012	107.2	2.113
11	12	2012	105.1	2.058
12	1	2013	103.9	2.06
13	2	2013	103	2.035
14	3	2013	104.8	2.08
15	4	2013	105	2.102
16	5	2013	107.2	2.15

That's good. And now down to business. Call the new data set whatever you like:

```
data micro2;
  set micro;
  date=mdy(month,1,year);
  drop month year;
```

Another way to do this is to construct a piece of text out of the date, and then turn it into a date using `input`:

```
data micro3;
  set micro;
  date_text=cat('1 ',month,' ',year);
  date=input(date_text,anydtdte20.);
  drop month year date_text;
```

Two points for getting the date right (either via `mdy` or the `cat-input` thing, and one for the `set micro` at the front and the `drop` (or `keep`) at the end. Two marks if you made up to about two mistakes on getting the date right, or if you forgot to drop the variables you didn't want. (Two covers a wide range.) One mark if you got *something* right. I was rather generous on this one.

(b) (2 marks) If I run simply `proc print` on the new data set, how will the dates be displayed?

Solution: As days since Jan 1 1960 (that is, as numbers, not as anything that looks like a date.)

To illustrate:

```
proc print;
```

Obs	comp	indproc	date
1	102.9	2.052	19024
2	101.5	2.026	19053
3	100.8	2.002	19084
4	98	1.949	19114
5	97.3	1.942	19145
6	93.5	1.887	19175
7	97.5	1.986	19206
8	102.2	2.053	19237
9	105	2.102	19267
10	107.2	2.113	19298
11	105.1	2.058	19328
12	103.9	2.06	19359
13	103	2.035	19390
14	104.8	2.08	19418
15	105	2.102	19449
16	107.2	2.15	19479

These dates are in 2012 and 2013, a bit over 50 years since the zero date, so the dates-as-numbers should be a bit more than $365 \times 50 \simeq 18,000$, which they are.

"As numbers" is worth one point, but if you want the second point, you'll have to tell me *what* numbers.

The point of this one is to say that, in SAS, if you make dates this way, you *have* to supply a `format` (next part), or else you get something apparently meaningless. `proc import` guesses the date format (if you read the dates in from a file), but if you build them yourself as text, it doesn't.

(c) (2 marks) What code would I add to `proc print` to display the dates in UK format, that is, day-month-year with the month as a number and the year as 4 digits?

Solution: Add a `format`. The one that fits the specification is `ddmmyy10.` Don't forget the dot on the end! It's "10" because the day, the month and the 4-digit year together are 8 characters, but if you add the `-` between them, that's 10 characters altogether. (It's a little bit un-mnemonic in that you'd expect it to be `ddmmyyyy10.` instead, but it isn't.)

```
proc print;
  format date ddmmyy10.;
```

Obs	comp	indproc	date
1	102.9	2.052	01/02/2012
2	101.5	2.026	01/03/2012
3	100.8	2.002	01/04/2012
4	98	1.949	01/05/2012
5	97.3	1.942	01/06/2012
6	93.5	1.887	01/07/2012
7	97.5	1.986	01/08/2012
8	102.2	2.053	01/09/2012
9	105	2.102	01/10/2012
10	107.2	2.113	01/11/2012
11	105.1	2.058	01/12/2012
12	103.9	2.06	01/01/2013
13	103	2.035	01/02/2013
14	104.8	2.08	01/03/2013
15	105	2.102	01/04/2013
16	107.2	2.15	01/05/2013

If you called the new date variable something other than `date`, use your name here. I'm checking for consistency.

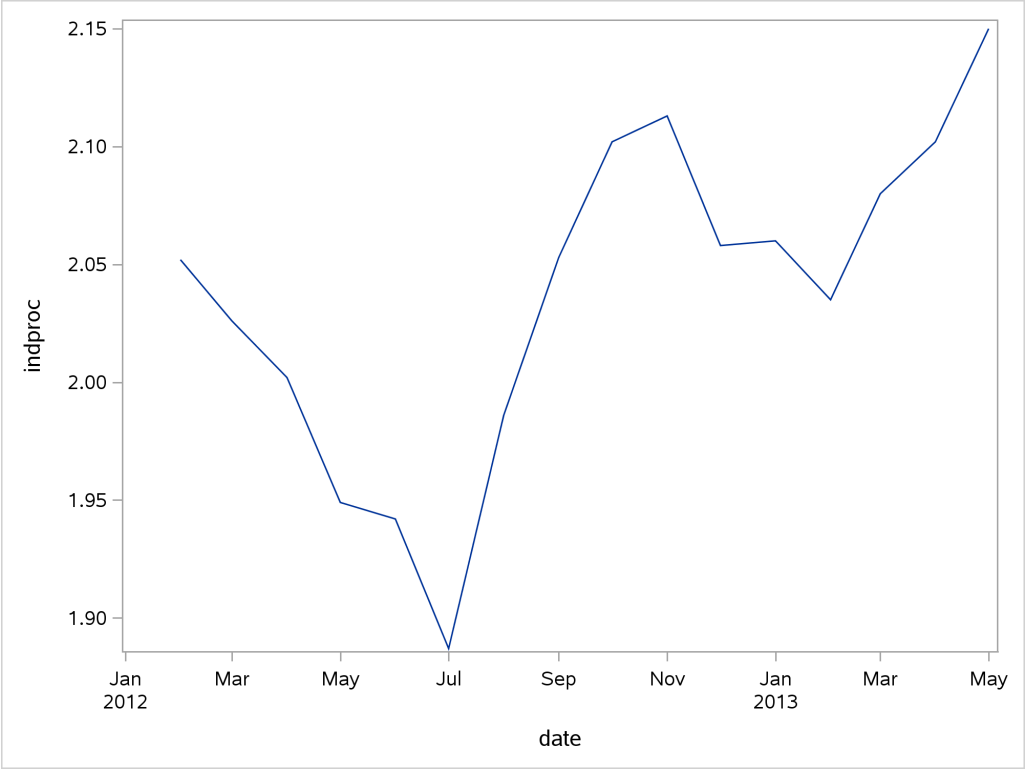
I felt extra-mean taking off a point if you missed the dot on the end of the format, but it's important; if you leave it out, SAS thinks you mean a *variable* called `ddmmyy10` and gets very confused (the error message doesn't help much, as I recall).

One point for a `format` with something else that looks plausibly like a date format.

(d) (3 marks) What code could I use to plot `indproc` against time, joining the points by lines, and making sure the time axis is appropriately labelled? (*You* have to decide what "appropriately" means here.)

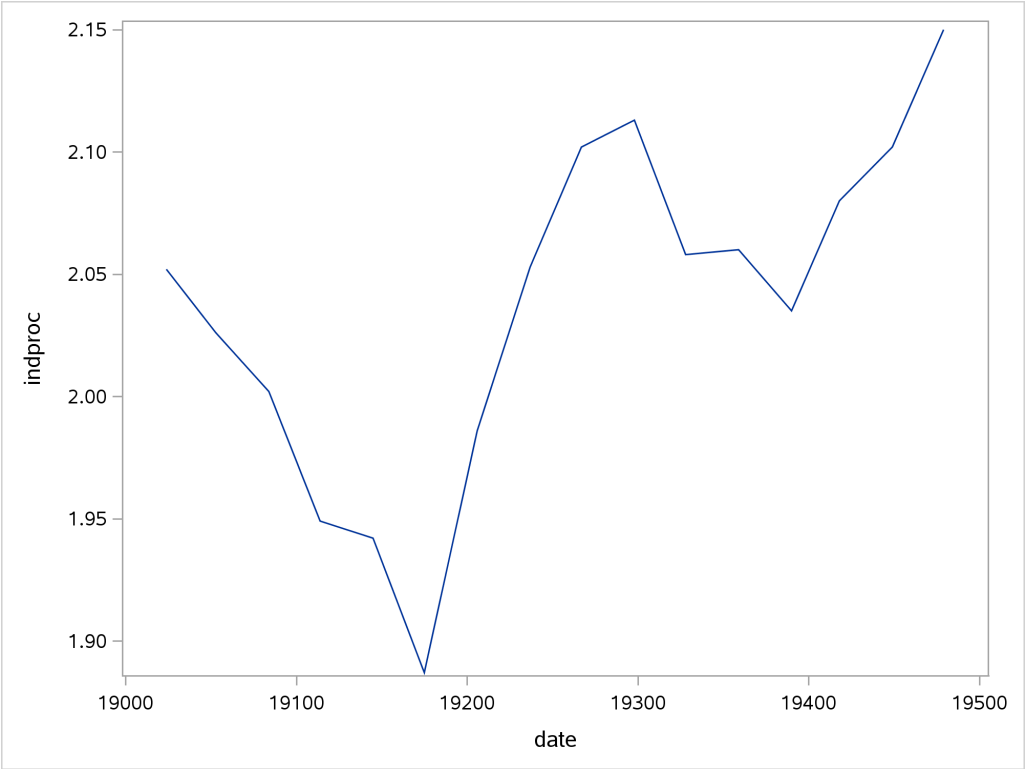
Solution: This is `proc sgplot`, but using `series` instead of `scatter` (to join the points by lines) and a `format` on the dates. I don't mind what format you use, as long as you use one that is appropriate for dates, eg.:

```
proc sgplot;
  series x=date y=indproc;
  format date yymmdd10.;
```



I'm not quite sure how the month *names* got in there, since `yymmdd10.` as a format is all numbers, but somehow SAS chose to display the dates as dates. If you forget the `format` it all goes astray:

```
proc sgplot;
  series x=date y=indproc;
```



The dates are days-since-Jan-1-1960 again. This is *not* an appropriate labelling of the time axis.

The previous part is a clue that you needed a `format` (since it's the same principle of displaying dates properly). This is what I meant by "time axis is appropriately labelled". If you want to add axis label text, be my guest (I might use this to give you an extra mark if you're on the edge), but the `format` is what I wanted.

Marking: 3 points if you got both the `series` and the `format` right (I forgave some tiny errors, and didn't punish you again for missing a dot in the `format`). 2 points if one of the `series` or `format` was wrong or missing, and 1 point if you plotted the data *somehow*.

`loess` or `reg` were not quite the right thing because they draw trends through the scatter of points, without necessarily going through all the points (going through all the points is what I meant by "joining the points with lines"). If you added one of those, or `scatter`, to `series`, I was fine with that, as long as `series` was in there somewhere.