University of Toronto Scarborough
Department of Computer and Mathematical Sciences
STAC32 (K. Butler), Final Exam
December 19, 2016 2:00–5:00pm

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

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 80 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.

Figure 1 of the booklet of code and output shows the R packages that I loaded in preparing this exam. You may assume that these packages have been loaded, and your code can use anything within them without further comment.

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.

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	3	
3	2	
4	2	
10	2	
12	3	
14	3	
16	5	
18	3	
20	2	
24	4	
25	6	
26	3	
29	5	
34	2	
36	5	
37	3	
39	4	
44	3	
45	2	
46	7	
48	4	
49	4	
53	9	
55	4	
56	3	

1. The data shown in Figure 2 of the booklet of code and output come from a survey. Ten individuals are listed here; the columns respectively indicate a numerical ID, gender, age, income (on some scale), and the scores on three tests, which will be labelled R1, R2 and R3.

   Each of the question parts below can be answered with three lines of code or less (2017: except for (a), which might be longer).

   (a) (3 marks) The data have been stored in a file `survey.txt` in your folder on SAS Studio. Give SAS code to read the data in with suitable variable names. (You may lose marks for any unnecessary code.) Extra for 2017: why can I not use last year's data file?

Solution: With your username rather than mine, or anything that looks like a username:

2016 (2017 version below):

```
data survey;
  infile '/home/ken/survey.txt';
  input id gender $ age income r1 r2 r3;
```

and to convince myself that it worked:

```
proc print;
```

Obs	id	gender	age	income	r1	r2	r3
1	1	F	35	17	7	2	2
2	17	M	50	14	5	5	3
3	33	F	45	6	7	2	7
4	49	M	24	14	7	5	7
5	65	F	52	9	4	7	7
6	81	M	44	11	7	7	7
7	2	F	34	17	6	5	3
8	18	M	40	14	7	5	2
9	34	F	47	6	6	5	6
10	50	M	35	17	5	7	5

Question 1 continues... This page: _____ of possible 3 points.

This was about the easiest possible, since there are no lines to skip. The test names can be in uppercase or lowercase, both when you read them in and when you use them.

The virtual-machine way would have `/folders/myfolders/survey.txt` on the `infile` line. If you started with `/folders` you needed to follow it with `myfolders`, but if you started with `/home`, I was willing to tolerate `folders` coming next, since that could conceivably be a username.

There ought to be *no* `firstobs`, but I was just about willing to tolerate it if you said `firstobs=1` (this is very close to being unnecessary code).

There's nothing special about reading in the gender (beyond it being text, so needing the dollar sign), since it is less than 8 characters long. If you insist on reading it in with something like `input ... gender: $1.`, that's OK (because it will work), but not obligatory. (I won't call *that* "unnecessary".)

2017: `proc import`, which makes it longer (but arguably easier) than three lines. This version of the data file is (for me) stored in `survey2.txt`:

```
proc import
  datafile='/home/ken/survey2.txt'
  out=survey
  dbms=dlm
  replace;
  getnames=no;
  delimiter=' ';
```

(note that there are *no* variable names: did you look carefully enough at Figure 2?)

Did it work?

```
proc print;
```

Obs	VAR1	VAR2	VAR3	VAR4
1	1	F	35	17
2	17	M	50	14
3	33	F	45	6
4	49	M	24	14
5	65	F	52	9
6	81	M	44	11
7	2	F	34	17
8	18	M	40	14
9	34	F	47	6
10	50	M	35	17

Obs	VAR5	VAR6	VAR7
1	7	2	2
2	5	5	3
3	7	2	7
4	7	5	7
5	4	7	7
6	7	7	7
7	6	5	3
8	7	5	2
9	6	5	6
10	5	7	5

Note that we have variable names `VAR1, VAR2` etc., rather than the more descriptive ones that we would have gotten last year.

Last year's data file wouldn't have worked because it has *more than one* space between data values. (Reading it as shown above works, but we didn't do that this year. `proc import` is picky as `read_delim` is: it wants *exactly one* space between values.

Since I want to use this data set, but I don't want to mess around with the names, let's do a little reorganizing. SAS has a `rename` that you haven't seen before, but it should be fairly obvious how it works:

```
data survey2;
  set survey;
  rename var1=id var2=gender var3=age var4=income var5=r1 var6=r2 var7=r3;

proc print;
```

Obs	id	gender	age	income
1	1	F	35	17
2	17	M	50	14
3	33	F	45	6
4	49	M	24	14
5	65	F	52	9
6	81	M	44	11
7	2	F	34	17
8	18	M	40	14
9	34	F	47	6
10	50	M	35	17

Obs	r1	r2	r3
1	7	2	2
2	5	5	3
3	7	2	7
4	7	5	7
5	4	7	7
6	7	7	7
7	6	5	3
8	7	5	2
9	6	5	6
10	5	7	5

That looks better. We'll use that from here.

Don't worry, you won't see this on the exam.

From here, we'll use this data set.

(b) (2 marks) Give SAS code to calculate the mean scores on each of the tests R1, R2 and R3 for each gender.

Solution: A gentle start:

```
proc means;
```

```
var r1 r2 r3;
class gender;
```

with output

```
                          The MEANS Procedure

            N
gender    Obs    Variable     N            Mean         Std Dev         Minimum
-------------------------------------------------------------------------------
F           5    r1           5       6.0000000       1.2247449       4.0000000
                 r2           5       4.2000000       2.1679483       2.0000000
                 r3           5       5.0000000       2.3452079       2.0000000

M           5    r1           5       6.2000000       1.0954451       5.0000000
                 r2           5       5.8000000       1.0954451       5.0000000
                 r3           5       4.8000000       2.2803509       2.0000000
-------------------------------------------------------------------------------

                          N
                gender    Obs    Variable         Maximum
                ----------------------------------------
                F           5    r1             7.0000000
                                 r2             7.0000000
                                 r3             7.0000000

                M           5    r1             7.0000000
                                 r2             7.0000000
                                 r3             7.0000000
                ----------------------------------------
```

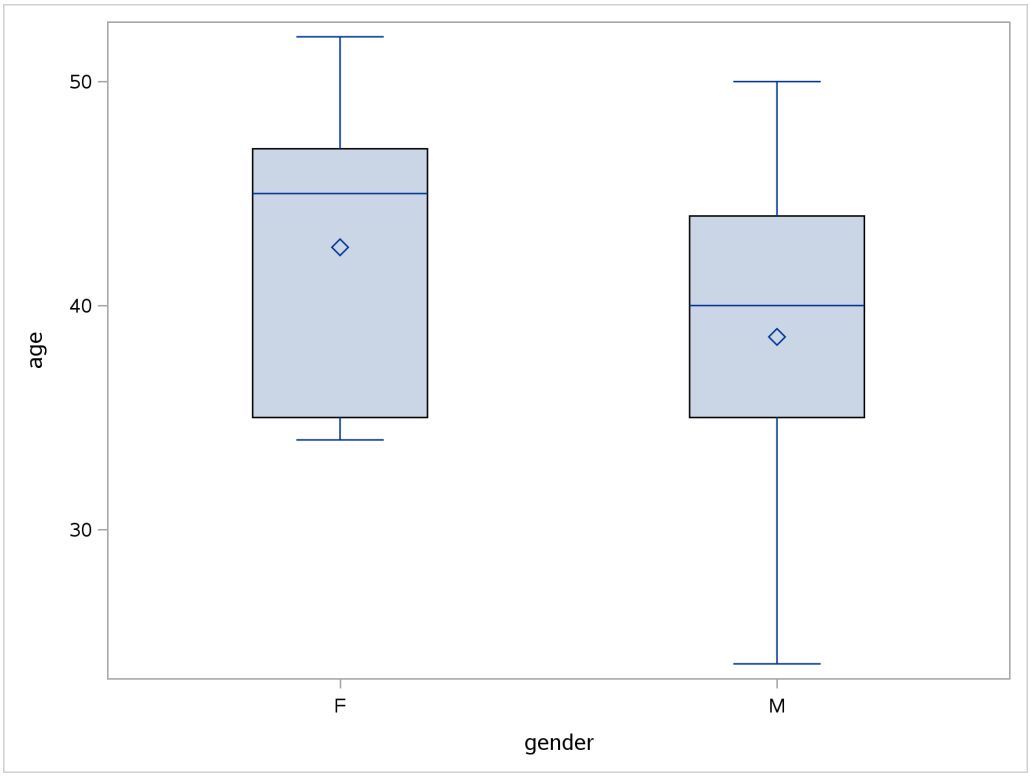
Doing three separate `proc means`, one for each test score, will work, but it's not the best answer (nor does it answer the whole question in three lines of code), so it's only worth one point.

It's not called `proc mean`, singular. If you think it is, that'll cost you a point.

(c) (2 marks) Give SAS code to obtain side-by-side boxplots of age for each gender.

Solution: Nothing difficult here:

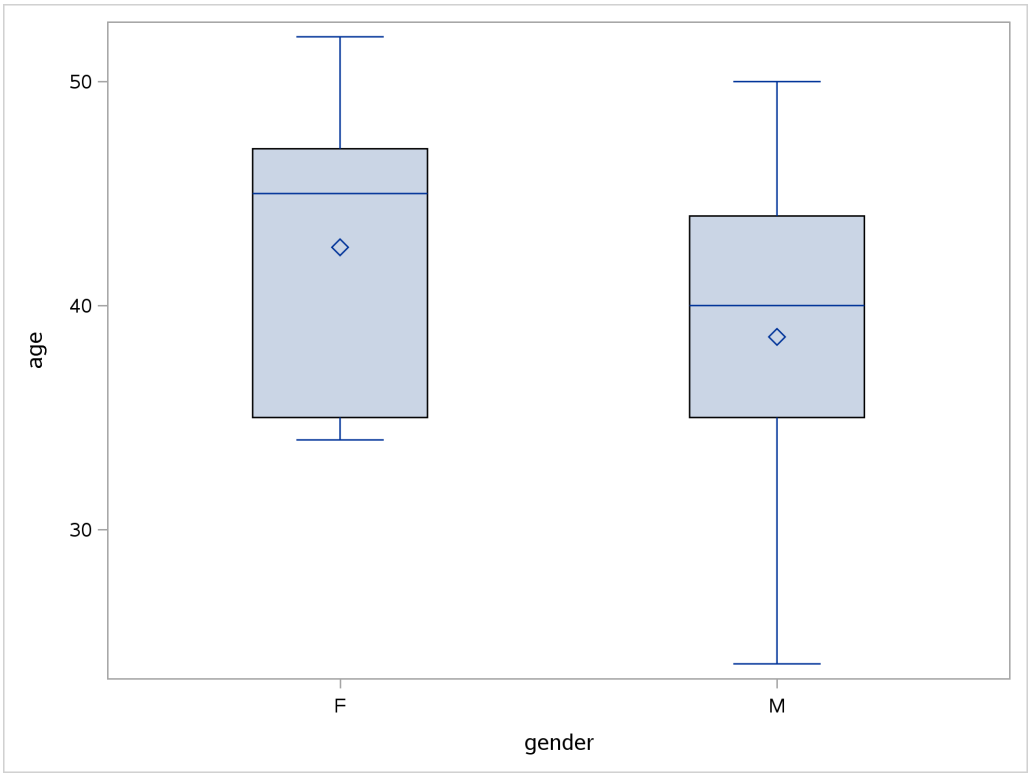
```
proc sgplot;
  vbox age / category=gender;
```


50
40
30
age
F
M
gender

Not very illuminating boxplots with only five observations per group, but here they are.

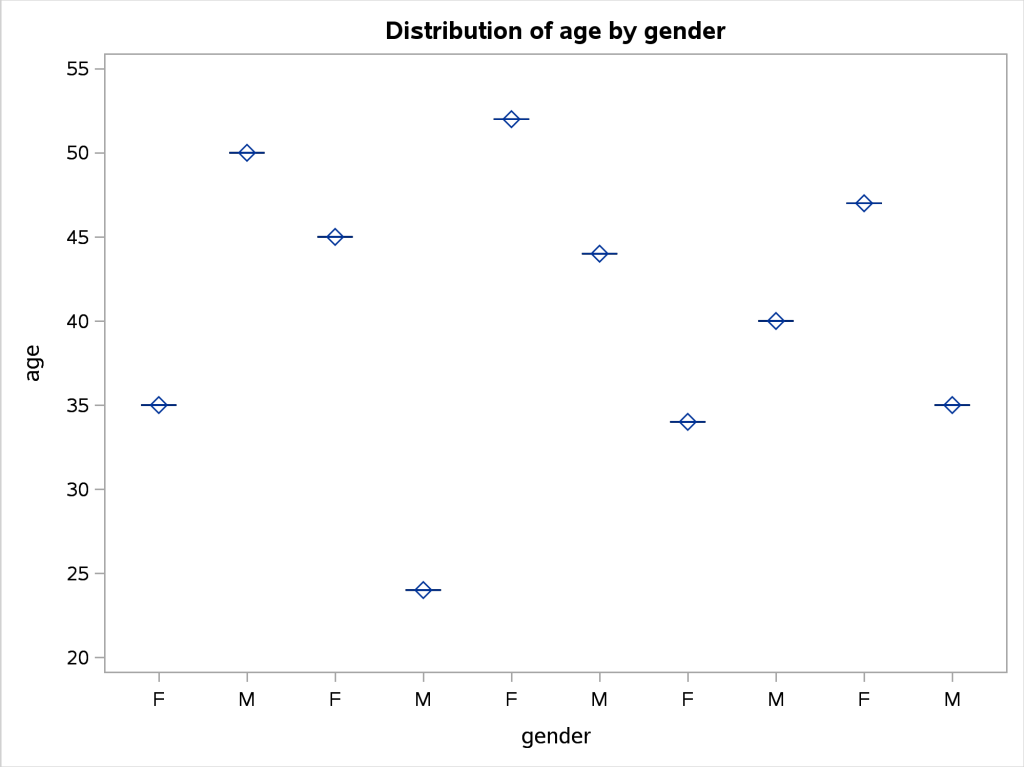
This also works:

```
proc sgplot;
  vbox age / group=gender;
```


50
40
30
age
F
M
gender

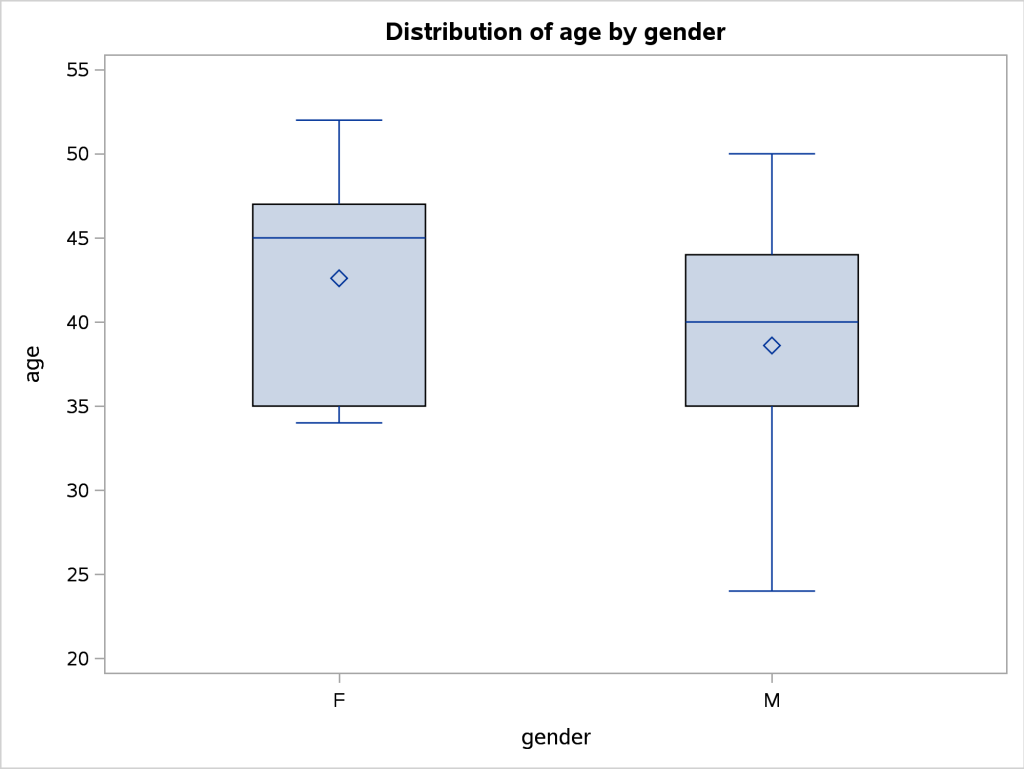
I didn't use `proc boxplot` in class (this year), and thus, if you use it, it rather demonstrates that you didn't come to lectures. This is not in itself a bad thing, and if your code will work, I'm good with it. However, this obvious code does *not* work:

```
proc boxplot;
  plot age*gender / boxstyle=schematic;
```


Distribution of age by gender
55
50
45
40
35
30
25
20
age
F
M
F
M
F
M
F
M
F
M
gender

If you read about `proc boxplot`, you'll find that the categorizing variable (`gender`) has to be *sorted into order*. Thus the code you need is all of this:

```
proc sort;
  by gender;
proc boxplot;
  plot age*gender / boxstyle=schematic;
```



The reason you need the `proc sort` is that the genders are all mixed up (as you can see in Figure 2), and if you leave it out, you'll get a separate "boxplot" for each run of people all of the same gender.

Also, if you leave out the `boxstyle=schematic`, you won't get the outliers plotted separately as per all our other types of boxplot.

With all that in mind, an otherwise correct `proc boxplot` gets 1 mark only, and if you're missing the `boxtype` thing, it's not worth anything.

Live by old exams, die by old exams.

(d) (2 marks) The same data set, now in a file `survey.txt` (2017: `survey2.txt`) in your current R working folder, can be read into a data frame called `survey2` (2016: `survey`) in R, using `read.table` with a suitable value for `header`. What value? (2017: using `read_delim`, somehow. Show how.) And what names will the columns have?

Solution:

I didn't seem to have one of these yet:

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.0.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.6
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

2017: like this:

```
survey2=read_delim("survey2.txt"," ",col_names=F)
```

```
## Parsed with column specification:
## cols(
##   X1 = col_integer(),
##   X2 = col_character(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_integer(),
##   X6 = col_integer(),
##   X7 = col_integer()
## )
```

```
survey2
```

```
## # A tibble: 10 x 7
##       X1 X2       X3    X4    X5    X6    X7
##    <int> <chr> <int> <int> <int> <int> <int>
##  1     1 F        35    17     7     2     2
##  2    17 M        50    14     5     5     3
##  3    33 F        45     6     7     2     7
##  4    49 M        24    14     7     5     7
##  5    65 F        52     9     4     7     7
##  6    81 M        44    11     7     7     7
##  7     2 F        34    17     6     5     3
##  8    18 M        40    14     7     5     2
##  9    34 F        47     6     6     5     6
## 10    50 M        35    17     5     7     5
```

The columns get names `X1` through `X7`, which you ought to have been able to figure out, since you have seen it before.

2016: `header=F`, and the columns will get names `V1` through `V7`:

```
survey=read.table("survey.txt",header=F)
survey
```

```
##    V1 V2 V3 V4 V5 V6 V7
## 1   1  F 35 17  7  2  2
## 2  17  M 50 14  5  5  3
## 3  33  F 45  6  7  2  7
## 4  49  M 24 14  7  5  7
## 5  65  F 52  9  4  7  7
## 6  81  M 44 11  7  7  7
## 7   2  F 34 17  6  5  3
## 8  18  M 40 14  7  5  2
## 9  34  F 47  6  6  5  6
## 10 50  M 35 17  5  7  5
```

I need to keep a copy of this for later (but you don't):

```
survey.old=survey
```

Reading with `header=T` is *wrong*, because there is no top line that is headers. If you try, this is what will happen:

```
survey.wrong=read.table("survey.txt",header=T)
survey.wrong
```

```
##   X1 F X35 X17 X7 X2 X2.1
## 1 17 M  50  14  5  5    3
## 2 33 F  45   6  7  2    7
## 3 49 M  24  14  7  5    7
## 4 65 F  52   9  4  7    7
## 5 81 M  44  11  7  7    7
## 6  2 F  34  17  6  5    3
## 7 18 M  40  14  7  5    2
## 8 34 F  47   6  6  5    6
## 9 50 M  35  17  5  7    5
```

R tries to fashion variable names out of the top row of data, but since things starting with a number are not legal variable names, it puts an X in front of them (and you have lost the top row of the data).

If you thought `header` should be `T`, you need to say that these are the variable names you would get.

I had some sympathy for people who said `header=F` and that the columns had "no names" (since this is the right kind of thinking), but I decided this was only 1 out of 2.

(e) (3 marks) What R code will give the columns names that describe what the columns contain?

Solution: Something like this. Make a vector of column names, and then set `names` of the data frame. The exact names you use don't matter.

```
mycols=c("id","gender","age","income","r1","r2","r3")
names(survey2)=mycols
survey2
```

```
## # A tibble: 10 x 7
##       id gender   age income    r1    r2    r3
##    <int> <chr>  <int>  <int> <int> <int> <int>
##  1     1 F         35     17     7     2     2
##  2    17 M         50     14     5     5     3
##  3    33 F         45      6     7     2     7
##  4    49 M         24     14     7     5     7
##  5    65 F         52      9     4     7     7
##  6    81 M         44     11     7     7     7
##  7     2 F         34     17     6     5     3
##  8    18 M         40     14     7     5     2
##  9    34 F         47      6     6     5     6
## 10    50 M         35     17     5     7     5
```

Or put the vector of names directly on the right side of `names(survey)`. If you said this in (c) (or some of it), it earns you marks for (d). Though you probably lost something in (c), since you likely didn't say "`V1` through `V7`" there.

Or, set the column names directly in the `read_delim`, thus:

```
survey2=read_delim("survey2.txt"," ",col_names=mycols)
```

```
## Parsed with column specification:
## cols(
##   id = col_integer(),
##   gender = col_character(),
##   age = col_integer(),
##   income = col_integer(),
##   r1 = col_integer(),
##   r2 = col_integer(),
##   r3 = col_integer()
## )
```

```
survey2
```

```
## # A tibble: 10 x 7
##       id gender   age income    r1    r2    r3
##    <int> <chr>  <int>  <int> <int> <int> <int>
##  1     1 F         35     17     7     2     2
##  2    17 M         50     14     5     5     3
##  3    33 F         45      6     7     2     7
##  4    49 M         24     14     7     5     7
##  5    65 F         52      9     4     7     7
##  6    81 M         44     11     7     7     7
##  7     2 F         34     17     6     5     3
##  8    18 M         40     14     7     5     2
##  9    34 F         47      6     6     5     6
## 10    50 M         35     17     5     7     5
```

The other way of doing this (admittedly a rather inelegant way, but if it's the best you can do

it'll work) is to do a large-scale `mutate`, renaming the columns one by one. A couple of people tried this, but they had a superfluous `paste`. Here's how I think it should go. (I'm using my copy `survey.old`, since the original `survey` already got renamed columns.)

```
library(tidyverse)
survey.old %>% mutate(id=V1, gender=V2, age=V3, income=V4,
    r1=V5, r2=V6, r3=V7) %>%
  select(id:r3)
```

```
##    id gender age income r1 r2 r3
## 1   1      F  35     17  7  2  2
## 2  17      M  50     14  5  5  3
## 3  33      F  45      6  7  2  7
## 4  49      M  24     14  7  5  7
## 5  65      F  52      9  4  7  7
## 6  81      M  44     11  7  7  7
## 7   2      F  34     17  6  5  3
## 8  18      M  40     14  7  5  2
## 9  34      F  47      6  6  5  6
## 10 50      M  35     17  5  7  5
```

(taking out the old `V1` through `V7`).

This, I think, is the most elegant way along these lines, but I didn't show you this in class, so I couldn't expect you to come up with it on an exam:

```
survey.old %>% rename(id=V1, gender=V2, age=V3, income=V4,
    r1=V5, r2=V6, r3=V7)
```

```
##    id gender age income r1 r2 r3
## 1   1      F  35     17  7  2  2
## 2  17      M  50     14  5  5  3
## 3  33      F  45      6  7  2  7
## 4  49      M  24     14  7  5  7
## 5  65      F  52      9  4  7  7
## 6  81      M  44     11  7  7  7
## 7   2      F  34     17  6  5  3
## 8  18      M  40     14  7  5  2
## 9  34      F  47      6  6  5  6
## 10 50      M  35     17  5  7  5
```

This idea is rather like the one I showed you in SAS above.

(f) (3 marks) Use something from `dplyr`/`tidyverse` to find the mean income for each gender. Give the code you would use. (You may assume that `dplyr` has already been loaded.)

Solution: `group_by` and `summarize`:

```
survey2 %>% group_by(gender) %>%
  summarize(m=mean(income))
```

```
## # A tibble: 2 x 2
##   gender     m
##   <chr>  <dbl>
## 1 F         11
## 2 M         14
```

Not `aggregate`, because I said "something from `dplyr`/`tidyverse`". If that was the best you could think of, though, this would get you one point (on the basis that getting an answer is better than getting no answer):

```
aggregate(income~gender,survey2,mean)
```

```
##   gender income
## 1      F     11
## 2      M     14
```

2. A clinic provides a program which is supposed to help their clients lose weight. They take a sample of 15 clients in the program. They weigh each client before the program begins, and again three months later. The data are shown in Figure 3. The clinic has just hired a Statistics graduate student to help them determine whether the program is effective or not.

   (a) (2 marks) Why are these data matched pairs rather than two independent samples? Explain briefly.

   Solution: There are 15 clients that each give *two* measurements, before and after, rather than 30 clients that give one measurement each (as would be the case with two independent samples). Or, each before and after measurement is linked by having been on the *same* client. Anything equivalent to that. This should be an easy two marks for you.

   The word "same" or "each" is what I was looking for. I liked the idea that there were two measurements on the same *individual* more than the two *groups* were the same, though the latter is not wrong and I couldn't give it less than two points. But "so we can assess the effect of the program" is not precise enough (because that could have been done using an experimental design that led to a two-sample t-test).

   (b) (3 marks) The graduate student has just learned about the spaghetti plot, and wants to draw one for the clinic. Figure 4 shows a line of code that the graduate student used in preparation for making the spaghetti plot. Describe briefly what columns the data frame `wtloss2` contains, and what values are in each column. (If you want to list the values, list only a few, enough to show that you know what they are.)

   Solution: This is what comes out, using the 2016 way of reading the data, which you can ignore:

```
weightloss=read.table("weightloss.txt",header=T)
wtloss2=weightloss %>% gather(when,weight,before:after)
wtloss2
```

```
##    client   when weight
## 1       1 before    210
## 2       2 before    205
## 3       3 before    193
## 4       4 before    182
## 5       5 before    259
## 6       6 before    239
## 7       7 before    164
## 8       8 before    197
## 9       9 before    222
## 10     10 before    211
## 11     11 before    187
## 12     12 before    175
## 13     13 before    186
## 14     14 before    243
## 15     15 before    246
## 16      1  after    197
## 17      2  after    195
## 18      3  after    191
## 19      4  after    174
## 20      5  after    236
## 21      6  after    226
## 22      7  after    157
## 23      8  after    196
## 24      9  after    201
## 25     10  after    196
## 26     11  after    181
## 27     12  after    164
## 28     13  after    181
## 29     14  after    229
## 30     15  after    231
```

The data frame contains three columns: a column `client` that is the numbers 1 through 15 (repeated), a column called `when` that contains the text `before` or `after`, and a column called `weight` that has all the weights, whether they were from before or after the program (which one they were is in `when`). A number of people forgot about `client`: the clue that it appears in the output is that is was *not* part of the columns that were `gather`ed together, so it must appear somehow in the output.

The data frame *does not* contain columns called `before` or `after`, since they were "gathered up" by the `gather`.

I did *not* ask "what does `gather` do?". I wanted you to take *your* knowledge of what `gather` would do here, and use it to tell me what the output would look like. (There is no column in the output called `before:after` or anything like that.)

If you want to list the values rather than describing them, list enough values so that I can see you have the right ones, but not so many that it takes you all exam to write them out (and takes me all day to check them). `when` contains something like

`before ... before after ... after,`

and weight contains something like

$$210, 205, 193, 182, \ldots 181, 229, 231.$$

There were some really nice answers. The best ones showed something like the output from `head(wtloss2)`, which was very easy for me to check. The *actual* order in which the observations will come out is as shown above, but if you thought that the two observations for client 1 would come out first, followed by the two for client 2, I was fine with that.

(c) (3 marks) What do you learn from the spaghetti plot, shown in Figure 36? Explain briefly. (Note that this plot is shown at the *end* of the booklet of code and output.)

Solution: All the lines are going uphill, which means that for every single client, the weight was higher before than after: that is to say, *everybody* lost weight, which indicates that the program is working consistently.

You can (optionally) add that the bigger weight losses appear to come from the people who were heaviest to start with. That would suggest that *percent* of weight lost is the thing to measure: that is, we could have taken logs of the before and after measurements. For a change, I'm not exploring that. In the source from which I got these data, they did a matched-pairs t-test, and got a strongly significant result. The weight loss is sometimes not all that big, which suggests that a confidence interval for the mean weight loss would be interesting. You can tell from the spaghetti plot what would happen in the sign test: all fifteen clients lost weight, so all fifteen before minus after differences would be positive and none negative. A 15-0 split is very unlikely if the program had no effect, so the sign test's P-value will be very small.

What, you want proof? SAS, for fun:

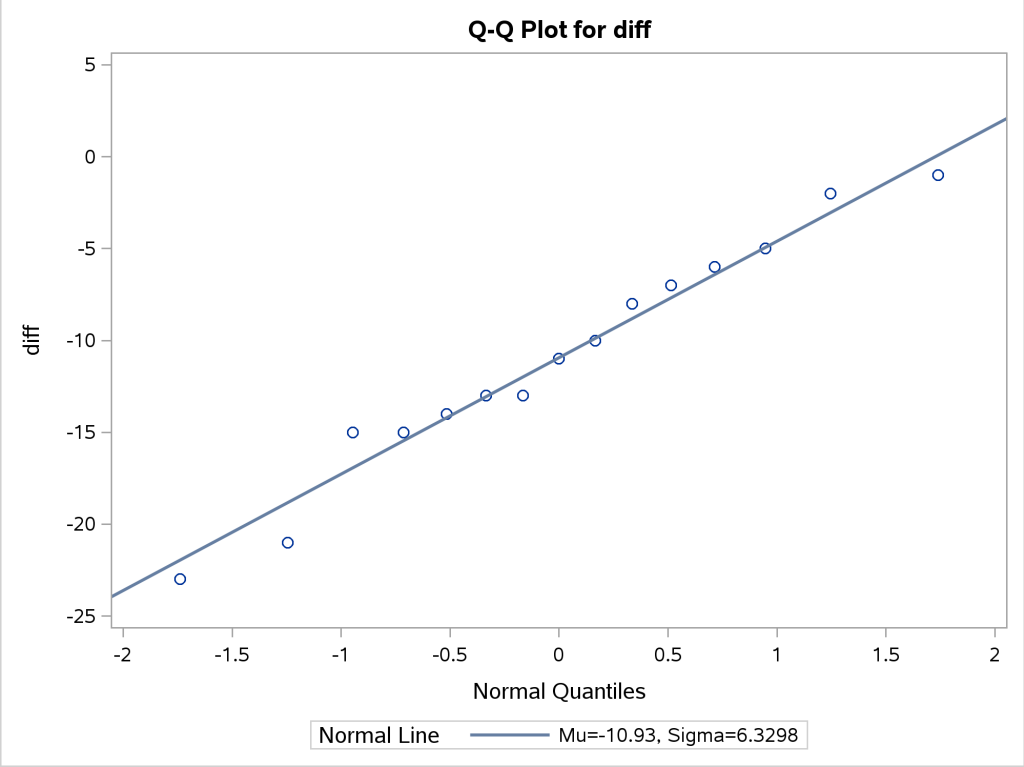
```
data weightloss;
  infile '/home/ken/weightloss.txt' firstobs=2;
  input client before after;
  diff=after-before;


proc univariate;
  var diff;
  qqplot diff / normal(mu=est sigma=est);
```

The text output:

```
                          The UNIVARIATE Procedure
                              Variable:  diff

                        Tests for Location: Mu0=0

               Test           -Statistic-    -----p Value------

               Student's t    t   -6.6897    Pr > |t|    <.0001
               Sign           M      -7.5    Pr >= |M|   <.0001
               Signed Rank    S       -60    Pr >= |S|   <.0001
```

We are testing a mean/median difference of 0, so we calculate the differences in the `data` step, and then throw those differences into `proc univariate`, pulling out the tests for location. The P-values for testing that the mean (t) and median (sign) differences are 0 are both roundly rejected. Both test statistics are negative, so we are "on the correct side" (I calculated the differences as after minus before), so we can also reject a mean/median difference of zero in favour of the *one*-sided alternative that the program is helping reduce weight. It doesn't make any difference whether we test the mean or the median; the conclusion is the same.

I should have looked at a normal quantile plot (earlier) but since I went to the trouble of getting one, I can look at it now:



That looks as straight as you could wish for. The t-test would have been perfectly fine. So let's do it:

```
proc ttest;
  paired before*after;
```

```
          N        Mean     Std Dev     Std Err     Minimum     Maximum

         15     10.9333      6.3298      1.6344      1.0000     23.0000
            Mean        95% CL Mean        Std Dev       95% CL Std Dev

          10.9333      7.4280  14.4387      6.3298      4.6342   9.9828
                               DF    t Value    Pr > |t|

                               14       6.69      <.0001
```

The confidence interval for the mean (difference) is the interesting thing: a mean weight loss between 7 and 14 pounds. That is, we're sure there is a weight loss, but we're also sure it's not all that big.

(d) (2 marks) Why is the spaghetti plot more informative for these data than side-by-side boxplots of the before and after weights would be? Explain briefly.

Solution: Side-by-side boxplots would lose the linkage between the two measurements for the same person, which is exactly what the spaghetti plot shows, and which is the crucial thing with matched-pair data. The boxplots would show that the median weight after is slightly lower than before, but with a lot of overlap, which loses the story that *everybody* lost weight over the course of the program. The distinction is between *individual* changes (spaghetti plot) and whole-sample changes (boxplots); the latter could be hiding a small but consistent change for everyone (which is what actually happened), or something like a mixture of large losses in weight for some people and moderate weight *gains* for others.

I wanted to see something about why the spaghetti plot was helpful, and also something about why the boxplots were less helpful. I was fairly relaxed about what I would accept for the second one, but I wanted to see *something*.

I could do the boxplots in SAS, but the data are in wide format and I want long. It can be done, but I forget how (and I am currently sitting in a coffee shop whose wifi has a password that I don't know).

Much later: I figured out what to do. The solution is, inevitably, to create a new data set from the current "wide" one. Each line of the current data set has to produce *two* lines in the new long one. In fact, the process is exactly what `gather` does, except that we have to keep track of things ourselves: we have variables `client`, `before` and `after`; we want to keep `client`, but create new variables `weight` (the weight, regardless of time) and `when`, which is `before` or `after` as appropriate. Here's how it goes:

```
data weightloss_long;
  set weightloss;
  when="before";
  weight=before;
  output;
```

```
when="after";
weight=after;
output;
drop before after;
```

The idea is that we construct each of the two possible values of `when`, that is, `before` and `after`, and the value of `weight` that goes with each one is the *value* of the variable `before` or `after` as appropriate. This is rather confusing, but it's exactly the same thing as `gather` does. (I didn't say anything about `client`, but it should have been passed straight through unchanged to the new data set, two copies of 1–15.) The `output` thing says "create a row of the new data set containing the variables as they now stand"; there are two of these, since each row of the wide data set contains *two* rows of the long one. Finally, we can get rid of `before` and `after`, since they have served their purpose (namely, providing values to put into `weight`).

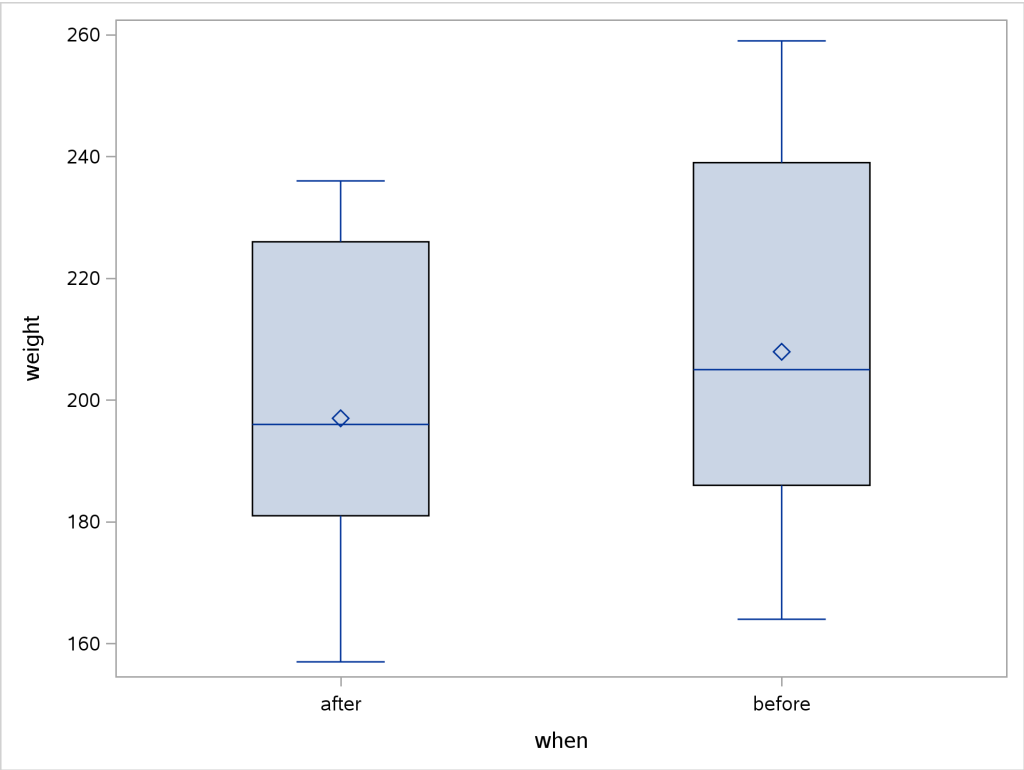
Did it work? Well, one way to find out:

```
proc print;
```

Obs	client	diff	when	weight
1	1	-13	before	210
2	1	-13	after	197
3	2	-10	before	205
4	2	-10	after	195
5	3	-2	before	193
6	3	-2	after	191
7	4	-8	before	182
8	4	-8	after	174
9	5	-23	before	259
10	5	-23	after	236
11	6	-13	before	239
12	6	-13	after	226
13	7	-7	before	164
14	7	-7	after	157
15	8	-1	before	197
16	8	-1	after	196
17	9	-21	before	222
18	9	-21	after	201
19	10	-15	before	211
20	10	-15	after	196
21	11	-6	before	187
22	11	-6	after	181
23	12	-11	before	175
24	12	-11	after	164
25	13	-5	before	186
26	13	-5	after	181
27	14	-14	before	243
28	14	-14	after	229
29	15	-15	before	246
30	15	-15	after	231

It did, except that I created a variable `diff` above which doesn't quite fit in the long data set (so I'll ignore it). The rows, as you can check, came out in a different order to the `gather` output, but they are the same 30 rows overall. So now we can make the boxplot that was the reason for wanting to do this:

```
proc sgplot;
  vbox weight / category=when;
```



As you see, not much apparent difference between before and after, but only because the boxplots are the wrong way of looking at this paired data.

Thanks to `http://www.ats.ucla.edu/stat/sas/modules/widetolong_data.htm` for the inspiration for this. I used their first example, but the "right" way is the way in their second example, which uses the same "array" idea as we had for indicator variables (in connection with leverages).

3. Some psychologists study "stereotype threat". Consider people who identify as members of a group characterized by the ability to perform some task at a high level. According to the theory, those people can feel threatened if they are told that some other group can perform that task better. This threat has the effect of lowering their performance at the task they are good at.

   To assess whether stereotype threat occurs in practice, 23 white male students, who were known to be good at math, were were recruited for a study. These students were randomly divided into two groups. The first group, a control group labelled `exam`, were asked to write a difficult math exam. The second group, before writing the same exam, were told "Asian students typically do better than other students in math exams". This group was labelled `threat`. For each student, the score on the math exam was recorded, a higher score being better. Our task is to investigate whether the threat did indeed lower the scores of the students in the second group.

   The data are shown in Figure 5.

   (a) (2 marks) Side-by-side boxplots of exam scores by group (exam only or exam plus threat) are shown in Figure 6. Do the boxplots appear to support the researchers' hypothesis? Explain briefly.

   Solution: According to the theory, exam scores should be *lower* for the threat group. If you compare the medians on the boxplots, they are about 9 for the control group and about 7 for the treatment group. So the researchers' hypothesis appears to be supported by the data.

   I was going after your intuition here, since we'll be doing a test later, and I wanted to get you thinking about what to expect. (If you want to say that the difference in medians is small and therefore the research hypothesis is not really supported, that's OK too.)

   The shape of the distributions doesn't matter (much) here, especially if you're comparing medians (the obvious thing on a boxplot).

   I *did* want you to think about the direction of the difference, and not just say "the medians are different", since the research hypothesis predicts that marks for the threat group should be specifically *lower*, which they are. Whether they are far enough lower to be *significantly* lower is something we tackle below.

   (b) (2 marks) By looking at the boxplots in Figure 6, give *two* reasons why you might be unwilling to run a two-sample t-test on these data. (The recruited white male students were divided into two groups of approximately equal size; a reason that applies to both groups counts as two reasons.)

   Solution: The t-test assumes that *both* groups have a normal distribution. With relatively small sample sizes, that normal assumption is more important (so we should be more critical in looking for trouble than if we had larger sample sizes).

   I see three things, of which you need to get two:

   (i) the threat group has an *outlier at the lower end*.

   (ii) even apart from the outlier, the scores in the threat group are *skewed to the left*, because of the long lower whisker (and thus taking the outlier into account, these scores are clearly skewed to the left).

   (iii) the scores in the exam group are also *skewed to the left*.

   If we had 60 students in each group, this skewness might not matter, but with only 23 students, actually split into groups of 11 and 12, I think we should be cautious. (I'll compare the t-test later.)

Note: observing that the sample sizes are small is not by itself a reason to be unwilling to run the t-test. With small samples from apparently normal distributions, there is no problem. You need to get at non-normality somehow. You can use the small sample sizes to support your argument (as I did just now), but your main point has to be the non-normality of at least one of the distributions. Justifying *one* group being non-normal is enough; you don't need to consider the exam group if you can make two points about the threat group.

I wanted to see something more than "non-normal" — how do you know the groups are not normal? The word "skewed" is enough here.

Remember that the two-sample t-test as we learned it does *not* assume equal spreads. That's what the Satterthwaite-Welch test takes care of. (If we had only learned the pooled two-sample t-test, this would matter, but we have a way of dealing with unequal spreads.) Likewise, there is no requirement for the sample sizes in the two groups to be equal. The test will work just fine if they are different. (When you are *designing* a study, you probably want to have your samples equal, or approximately so, so as to get as much power as you can, but having unequal sample sizes does not stop the test working.)

(c) (2 marks) Look at the computations in Figure 7. Explain briefly what `obs` and `omd` contain.

Solution: `obs` contains the median mark for each group; `omd` contains the difference in medians, threat group minus exam group (a difference that, according to the theory, should be negative). That easy. (But not means, medians.)

`omd` stands for "observed median difference", a reminder that I will be comparing it to the randomization median differences later. If you grew up in England at about the time I did, "OMD" stands for these guys: `https://en.wikipedia.org/wiki/Orchestral_Manoeuvres_in_the_Dark`, who are apparently still recording, 30 years on. I found a 2013 YouTube video of two old guys with no hair who nonetheless sounded the same as the OMD I remember.

2017: The rest of the question talks about a randomization test, which we didn't do this year. Think about how you would do it instead. There's some discussion in the solution to the final part of this question.

(d) (4 marks) Look at the function in Figure 8. Describe in words (i) what kind of input the function requires, (ii) what the function does (without using the word "sample"), (iii) what the function returns to the outside world.

Solution: The input is a data frame that contains columns called `group` and `mark` (such as are in the data frame `stereo`, but the data frame can be anything as long as it contains columns by those names). The function randomly shuffles the group membership and computes the median for each shuffled group. It returns the difference in medians between the two (shuffled) groups.

The way I was marking this, it was pretty easy to get 3 points (data frame as input, computing difference of medians, returning that). But to get the 4th point, you needed to express the idea that the function randomly reallocates the group membership and computes the medians of the reallocated groups. "Shuffling" or "randomizing" the group membership is a good way to express those. What `sample` is *not* doing in this case is taking a sample! Or, at least, it is, but is actually sampling *all* the values, which has the effect of randomly re-ordering them. Compare this:

```
sample(1:10)
```

```
##  [1]  2  7  5  8  9  6  4  1  3 10
```

with this:

```
sample(1:10,4)
```

```
## [1]  4  1 10  7
```

Only the second one is doing what you would think of as taking a sample (of 4 values out of the "population" 1 through 10). The first one is taking a "sample" of size 10 (without replacement), which is going to sample all the values, just in a different order. The clue is that the `sample` in my function does not have a second input (the number of values to sample), so that it will do the default thing, which is to "sample" all of the values.

(e) (3 marks) In Figure 9, a randomization test is carried out. What do you conclude from it, in the context of the data? Explain briefly. (2017 on: don't answer this question. Think about how you might analyze these data instead.)

Solution: Original solution is here. Solution for those reading in 2017 and on is further down:

First we simulate the randomization distribution of the difference between the group sample medians, if the two population medians are equal. Then we count how many of the observations in the randomization distribution are as least as negative (one-sided test) as we observed. There are 146 of these out of 1000 values in the randomization distribution, so our P-value is $146/1000 = 0.146$. (No doubling here, since the test is one-sided: if the stereotype threat exists, it *lowers* test scores.) This is not smaller than 0.05 (or any typical α), so we cannot reject the null that the two groups, the exam-only group and the threat group, have the same median. That is to say, there is no demonstrated evidence that the "stereotype threat" decreases the median exam score of the threatened group.

So, you need to pull out the right P-value, write down your null hypothesis (there is *no* difference in medians between exam and threat groups, or that there is no stereotype threat), decide whether to reject the null or not (here not), and make a statement about whether the stereotype threat exists (or whether there is a difference in median exam scores between the two groups).

A reminder that research hypothesis are *alternative* hypotheses, since they typically say that something is going to happen, while null hypotheses say that nothing is going to happen (that's why they're called "null").

I said that I would show you the t-test, for which I have to read in the data again:

```
stereo=read.table("stereotype-threat.txt",header=T)
t.test(mark~group,data=stereo,alternative="greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  mark by group
## t = 2.3565, df = 20.614, p-value = 0.01422
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.8217156       Inf
## sample estimates:
##   mean in group exam mean in group threat
##             9.636364             6.583333
```

This time we *reject* a hypothesis of equal means. Take a look at the sample means at the bottom of the output: they differ by 3, while the medians differ by only 2. Why would this be? My best guess is that the mean of the threat group is pulled down by that low outlier of 0:

```
stereo %>% group_by(group) %>%
    summarize(med=median(mark))
```

```
## # A tibble: 2 x 2
##   group    med
##   <fct>  <dbl>
## 1 exam       9
## 2 threat     7
```

This is true, but it's actually more than that: the mean is lower than the median in the `threat` group by about the same amount as the mean is *higher* than the median in the `exam` group. How can that be, if both groups are *left*-skewed? Well, the `exam`-group data are (using a pipe because I think it's clearer; `arrange` sorts things in order):

```
stereo %>% filter(group=="exam") %>% arrange(mark)
```

```
##    mark group
## 1     4  exam
## 2     6  exam
## 3     7  exam
## 4     8  exam
## 5     9  exam
## 6     9  exam
## 7    12  exam
## 8    12  exam
## 9    13  exam
## 10   13  exam
## 11   13  exam
```

The median is 9 (the 6th-lowest value), then there is a big gap to those 12s and 13s, which is the big upper part to the `exam` group box in the boxplot. So which do you think describes the centre of the exam group better, 9 (the median) or something a bit bigger than 9 (the mean)? It's really hard to say. With more data, there would probably be some exam marks in that gap, and the mean and median would not be so different.

For those reading this in 2017 and on, the obvious thing to try if the t-test doesn't work is

Mood's median test:

```
library(smmr)
median_test(stereo, mark, group)
```

```
## $table
##         above
## group    above below
##   exam       7     3
##   threat     3     7
##
## $test
##        what      value
## 1 statistic 3.20000000
## 2        df 1.00000000
## 3   P-value 0.07363827
```

The P-value is 0.0736. This, remember, is two-sided. The research (alternative) hypothesis was that the `threat` group would have a lower "typical" exam mark, and indeed most of them are below the overall median. Thus we are on the "correct side", and we are entitled to divide the P-value by two:

```
0.0736/2
```

```
## [1] 0.0368
```

This is now less than 0.05, so we can throw our hat across the room and declare that there *is* evidence that the threat reduces exam scores on average.

Final thought: one way in which "stereotype threat" might play out is that *for some people*, receiving the threat first would tend to lower their exam score, but for others, the exam score would be unaffected. This would mean a good deal more thinking for the statistician about an appropriate test.

4. A small New England college has historically admitted students with a mean SAT score of 520, with a standard deviation of 80. SAT scores are calibrated to have an approximately normal distribution. This year, the college admissions department has started a campaign with the aim of increasing the quality of the students admitted to the college. They are hoping that the mean SAT of students admitted next year will be 530. The plan is to admit 100 students next year.

   (a) (2 marks) The college administration plans to run a suitable t-test next year to determine whether the mean SAT score has increased. Why would the college admissions department be interested in the *power* of this test? Explain briefly.

   Solution: The admissions department expect that, next year, the mean SAT score will have increased (that is, the old null mean of 520 is no longer true). They want the administration's test to reject this no-longer-true null mean, at least most of the time, in favour of the alternative that the mean has increased; that is, they want the power of the test to be high. (If the power is not high, the administration may fail to get evidence that the mean SAT score has increased, which will make the admissions department's campaign look as if it has failed, even if it has actually succeeded.)

   The issue here is being *clear* about what power is: IF the mean SAT score next year is 530, THEN what is the chance of correctly rejecting $H_0 : \mu = 520$ in favour of $H_a : \mu > 520$? Power has nothing to do with the probability of observing a mean of 530, and note that 530 doesn't appear in the hypotheses at all.

   (b) (3 marks) If the college achieves what it hopes, how likely is it that they will be able to demonstrate a statistically significant increase in mean SAT score? Use the appropriate one of Figures 10 through 13 to obtain your answer, and explain briefly why your choice is correct. If none of the choices is appropriate, state this and explain briefly why.

   Solution: In `power.t.test`, the quantity `delta` is the *difference* between the null-hypothesis mean and true mean, thus `delta` should be $530 - 520 = 10$. Also, because the admissions department hopes to be able to demonstrate an *increased* mean, a one-sided test is appropriate. This means that power analysis 1 (in Figure 10) is the appropriate one, and thus the power is 0.34.

   This is a depressingly low power, so the admissions department won't be very happy.

   Make sure, after you've worked out which Figure you want, that you actually remember to give a numerical answer for the power!

   Another way to get hold of the power is by simulation: the idea is that you generate data from the *true* distribution, then run the test on the *old* null mean. Here, that would look like this:

```
set.seed(457299)
sat=rnorm(100,530,80)
tt=t.test(sat,mu=520,alternative="greater")
tt
```

```
##
##  One Sample t-test
##
## data:  sat
## t = 0.96102, df = 99, p-value = 0.1694
## alternative hypothesis: true mean is greater than 520
## 95 percent confidence interval:
##  514.0742      Inf
## sample estimates:
## mean of x
##  528.1429
```

```
tt$p.value
```

```
## [1] 0.1694408
```

I first set the seed of the random number generator, so if I run this document again, I'll get the same answer. Then I draw 100 values from a normal distribution with mean 530 and the same SD of 80 as before. These are 100 simulated incoming students. Then I test whether the mean SAT is 520 or greater than 520 (and I know that the null hypothesis is false because I "played God" in simulating data from a different distribution). This time, my P-value is 0.1694 so I fail to reject a mean of 520, and thus I make a type II error. (The sample mean is 528, which is higher than 520, but not enough higher to be able to reject 520.)

The last line shows that I could just pull out the P-value if I wanted.

Now, we're going to repeat this process a lot of times, so the idea is the same as for a randomization test: write a function to do it once, and then use `replicate` to run that function a lot of times.

I'm going to let the true mean be input to the function, for reasons I'll show you later, so I get this:

```
students=function(sat_mean) {
  sats=rnorm(100,sat_mean,80)
  tt=t.test(sats,mu=520,alternative="greater")
  tt$p.value
}
```

I can test this by running it, but I can only see if the results look "reasonable":

```
students(530)
```

```
## [1] 0.4286776
```

I guess so. This time the sample mean must have been closer to 520. So the idea is that we'll only reject a null mean of 520 if the sample mean is high enough, maybe something like 535 or 540.

Now, do it 1000 times:

```
pvals=replicate(1000,students(530))
head(pvals)
```

```
## [1] 0.53605492 0.03264173 0.02191674 0.04622838 0.06237580 0.28806933
```

and then see how many of those P-values would have led to rejection by being 0.05 or less:

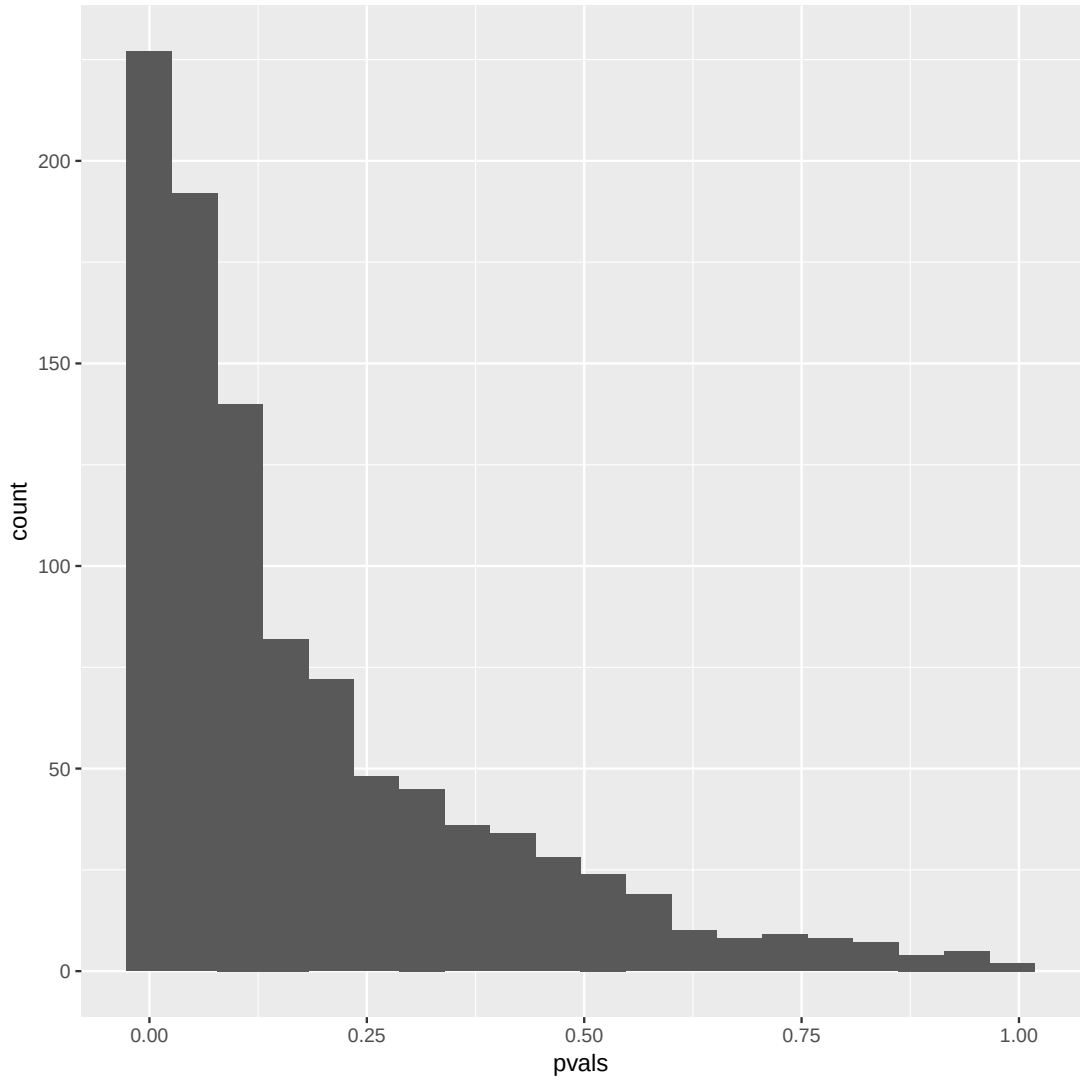
```
table(pvals<=0.05)
```

```
##
## FALSE  TRUE
##   664   336
```

We correctly rejected 336 times out of 1000, so the simulated power is 33.6%, very close to the correct answer of 34.3% that came out of `power.t.test`. But I like this way because you can see what's going on.

How about a histogram of those P-values?

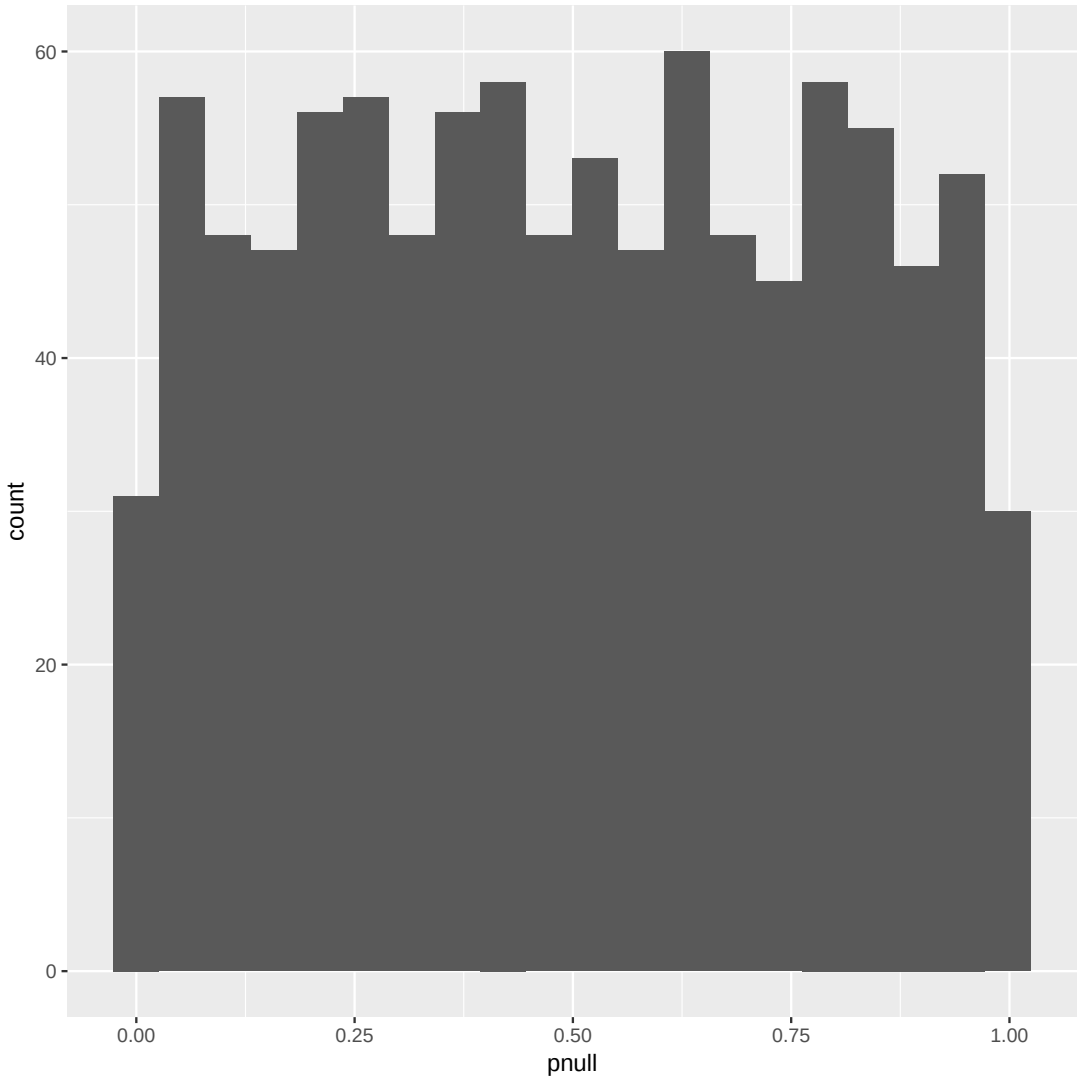
```
d=data.frame(pvals)
ggplot(d,aes(x=pvals))+geom_histogram(bins=20)
```



Most of the P-values are pretty small, so that even though we didn't reject $H_0 : \mu = 520$ as often as we wanted, the data we simulated usually suggested that the null wasn't true: we were often in that zone of "I don't think the population mean is 520, but I cannot prove it".

Now, what do the P-values look like if the null is actually true? This is why I wrote my function as I did, because I can answer the question this way:

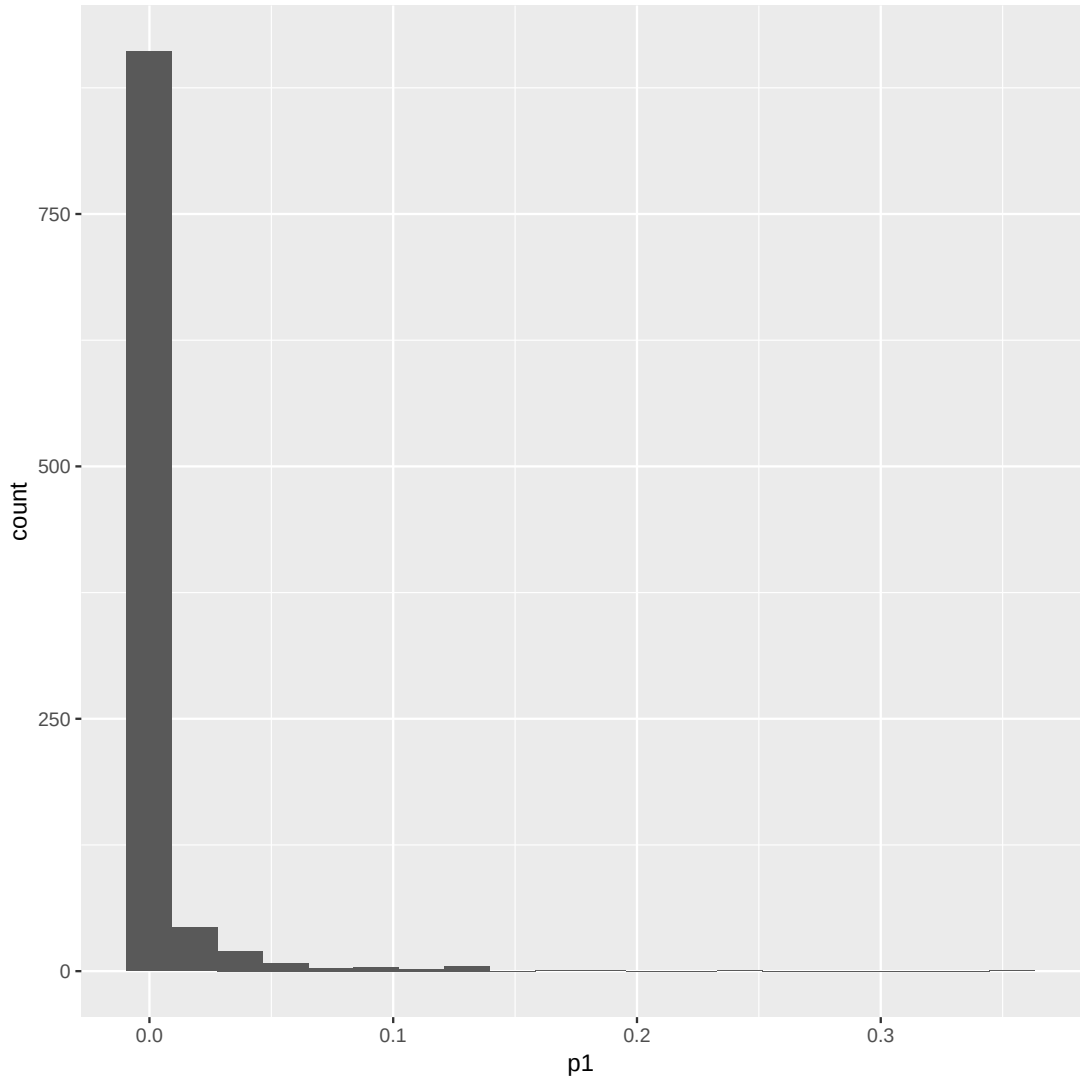
```
pnull=replicate(1000,students(520))
d=data.frame(pnull)
ggplot(d,aes(x=pnull))+geom_histogram(bins=20)
```



That looks pretty nearly uniform. (The two outermost bins are half as wide as the others, so they really ought to be twice as tall.) This is an illustration of a result from theoretical statistics: if the null hypothesis is true, the P-value has a *uniform distribution* on the interval $(0, 1)$. That is, for example, 5% of the time you would happen to get a P-value less than 0.05 and make a type I error.

But if the null hypothesis is actually *false*, the P-values are usually going to be closer to 0 than a uniform distribution would suggest, thus you will reject the null hypothesis (as you should) at least a bit *more* than 5% of the time, and thus you will have at least some power to detect that the null is wrong. The more wrong the null is (the bigger `delta` is, in the language of `power.t.test`), the bigger that power will be:

```
p1=replicate(1000,students(550))
d=data.frame(p1)
ggplot(d,aes(x=p1))+geom_histogram(bins=20)
```



Here, the P-value is more likely to be close to 0, and thus the power should be bigger:

```
table(p1<=0.05)
```

```
##
## FALSE  TRUE
##    25   975
```

0.975, in fact, with only 25 of the simulated P-values being bigger than 0.05.

(c) (2 marks) In each of Figures 10 through 13, `power.t.test` is used. The admissions department is curious about how many students it would have to admit to make the power of its test equal to 0.80. How would you change the code used in the appropriate one of these Figures to find this out?

Solution: Two things: omit the `n=100`, and include `power=0.80`. The answer is:

```
power.t.test(delta=10,sd=80,power=0.80,type="one.sample",
  alternative="one.sided")

##
##      One-sample t test power calculation
##
##               n = 397.0399
##           delta = 10
##              sd = 80
##       sig.level = 0.05
##           power = 0.8
##     alternative = one.sided
```

The college would have to admit nearly 400 students. (The power above was a fair bit less than 0.80, so the number of admitted students would have to increase a fair bit from 100.)

If you forget that you can include power, you can describe a procedure of modifying the sample size and recalculating the power, repeating until you get close enough to 0.80. But you have to describe the whole procedure.

Finally, remember to remove the `n=` completely. Having both an `n` and a `power` gives an error:

```
power.t.test(delta=10,sd=80,power=0.80,n=100,type="one.sample",
  alternative="one.sided")

## Error in power.t.test(delta = 10, sd = 80, power = 0.8, n = 100, type = "one.sample",
:  exactly one of 'n', 'delta', 'sd', 'power', and 'sig.level' must be NULL
```

You have to allow `power.t.test` to find one of those five things by specifying values for exactly *four* of the others listed. (`sig.level` defaults to 0.05, so you don't have to specify it if 0.05 is good for you.) Or you can specify the missing one like this:

```
power.t.test(delta=10,sd=80,power=0.80,n=NULL,type="one.sample",
  alternative="one.sided")

##
##      One-sample t test power calculation
##
##               n = 397.0399
##           delta = 10
##              sd = 80
##       sig.level = 0.05
##           power = 0.8
##     alternative = one.sided
```

5. Plants are much more sensitive to environmental light than humans. Experiments on plants can therefore be affected by using regular light. However, if the experimenter works in darkness, they cannot see what they are doing! Experiments on plants are usually performed in a dark room under "safelight", under which the experimenter can see, but where the plants are minimally affected.

   In an experiment, two different kinds of safelight (labelled A and B) were tested, each at high and low intensities (labelled H and L, so that for example BH means safelight B at high intensity). A control group, Darkness (labelled D) was also used. 40 seedlings were randomly allocated to one of the five groups (8 seedlings to each). The seedlings were allowed to grow for 20 days, and at the end of that time, their height was measured. The question of interest is "do either of the safelights, at either of the intensities, have an effect on plant height?"

   The structure of the data is shown in Figure 14.

   (a) (3 marks) The researcher intends to compare the heights for the different groups using analysis of variance. What two assumptions about ANOVA do the boxplots in Figure 15 enable you to assess? Do those assumptions appear to be satisfied? Explain briefly.

   Solution: The data need to be approximately normal, with equal spreads. These can be assessed using the boxplots. There are no outliers anywhere, and all the distributions are approximately symmetric. However, the spreads are not anywhere close to equal: treatments BH and D have much larger spreads than the others, and BL is much smaller than those.

   In summary, we should not be doing analysis of variance here, but rather something like Mood's Median Test (or a transformation, but that tends to be effective when larger spreads go with larger means, which they don't really here. Or see below for discussion about the Welch ANOVA, which is probably the best answer in 2018.)

   I am happy for you to disagree about the normality, but I think you really ought to conclude that the spreads are not close enough to be considered equal. If you name the two assumptions (normal and equal spread), and then go some sensible way towards assessing them, I am good.

   The other assumption, independent observations, cannot be assessed by looking at boxplots; you have to go back to the design of the experiment for that. (Since different seedlings are used in the different groups, it is reasonable to suppose that no observation is affected by another.)

   Assumptions are different from hypotheses. Hypotheses are what you are directly testing (that all the means are equal vs. not, in the case of ANOVA), while assumptions are other things about your data that will make your test reliable. We are not formally *testing* normality and equal variances here, but the closer they are to being satisfied, the more reliable/trustworthy the ANOVA will be. So talking about means or medians here is not addressing assumptions.

   Some people assess the assumptions by doing a test also. See, for example, Levene's test below. (There are *tests* for normality as well; for example, the Shapiro-Wilk test is equivalent to testing whether the normal quantile plot is straight enough. But these suffer from the usual trouble: smaller departures from normality or equal variances will be significant for larger samples, precisely the case where the normality especially matters less.)

   (b) (2 marks) Figure 16 and 17 show (respectively) an analysis of variance and Mood's median test for these data. Which test do you think is better? Explain briefly.

   Solution: I think the assumptions for the ANOVA fail (they aren't *both* good), so we should be doing the Mood's median test. (If your answer in the previous part said that ANOVA is all right, then you should choose ANOVA here. Consistency is all. And, in a situation where both tests are valid, the ANOVA will be more powerful and should be preferred.)

Some relevant discussion of the issue, in the light of what you previously said, can earn you something here. For example, if you didn't say anything in (a) about normality and equal spreads, but you say something here about how those issues would guide you in the choice of test, you will definitely get some points here. Again, though, thinking about means or medians (or about the kind of output the tests give) won't help you here. If one of the tests gives you more informative output, but a failure of assumptions indicates that that test should not be done, it doesn't matter *how* informative the output is.

What is definitely *wrong* here is to look at the two P-values and choose the test that has the smaller one. This is called "P-value hacking" or "going on a fishing expedition", because as soon as you choose a test by looking at its P-value, you *invalidate* that P-value. In this case, you have given yourself two chances to find a significant P-value instead of only one, so the probability of making a type I error is less than you think it is, and the probabilility of making a type II error is more. This is exactly the kind of issue that Tukey's method addresses. (Strictly speaking, this applies to variable selection in regression as well, when we look at the P-values for all the x-variables and remove the one with the biggest P-value. We get away with it in that case because, especially early in the process, the P-values of the variables we're removing are usually high, so even if we were able to adjust them for the multiple testing, the conclusion wouldn't change much. Looking at adjusted R-squared (or AIC or C_p, if you know about those) is not doing a test, so that avoids this issue.)

(c) (3 marks) Looking at the appropriate one of Figures 16 and 17, what P-value do you obtain? Express your P-value in scientific notation or as a decimal number. What, precisely, do you conclude? Explain briefly. (Mood's median test has a "warning: chi-squared approximation may be incorrect". You may ignore this.)

Solution: The right P-value is the one from the test you preferred: 7.8×10^{-6} (0.0000078) from Mood's median test, 1.2×10^{-9} (0.0000000012) from ANOVA. (There should be one fewer zero in the decimal P-values than in the negative power of 10.)

For Mood's median test, the null hypothesis is that the *median* height is the same for all treatments. This is rejected, so not all treatments have the same median height (or anything logically equivalent to that). You need to convince me that not all of the treatments have to be different; "there exist differences" or "at least one of the medians is different from the others" are ways to say that.

For the ANOVA, the null hypothesis is that the *mean* height is the same for all treatments. This, too, is resoundingly rejected, so not all treatments have the same mean height (or equivalent).

I do *not* want you to copy a P-value like `1.24e-09` as is; I need you to show that you know what it means.

All those words in the question were there for a purpose, to tell you what I was looking for. If you didn't answer the question as I asked it, don't expect to get all the points.

About that warning: if you have studied the chi-squared test, you will probably have run into a thing like "expected frequencies should all be bigger than 5", or possibly with "all" replaced by "mostly". The reason for that is not a failing of the test itself, but of using the chi-squared distribution to get a P-value for it. This is an "asymptotic approximation", meaning that as the number of observations gets bigger, the chi-squared distribution is a better and better approximation to the truth (which is complicated). People have found that the chi-squared distribution is acceptably good as an approximation if each of the expected frequencies in the cells is bigger than 5 (or, depending who you read, something a bit less stringent). Here, there

are 8 observations in each row of the table in Figure 17, because there were 8 observations in each treatment group, and if the group medians are all equal, there are expected to be 4 of them above and 4 below the overall median in each group. So the expected frequencies are all 4, which is less than 5. Hence the warning.

I am not worrying about it here, partly because I don't want to muddy the issue of doing a Mood's median test (which is what this question was all about), and partly because the P-value is so small and the pattern in the table so clear (see next part) that the chi-squared distribution can be a pretty awful approximation without changing what we conclude.

(d) (2 marks) Look again at Figure 17. What is it about the table `tab` that would make Mood's median test give the kind of P-value that it did? Explain briefly. (Answer this even if you think the ANOVA is the better test to use.)

Solution: If the treatments all have the same median, there should be about four observations in each treatment below the (overall) median and four above. But this very much does not happen: all 8 of the observations in groups AH and AL are below the overall median, and all but one of the observations in groups BH and BL are above. (This can also be seen from the boxplot; you can look back at the boxplot to help you figure out which values are above and which below.)

Unbalanced frequencies like this go with the medians being different from one group to another; in particular, it says that groups AH and AL have a low median, and groups BH and BL have a high one.

I need to see some clearly-explained insight here: telling me how the table is made is only worth one point. I want to see what it is about the numbers in the table that would make the test come out significant, so you need to make at least some comment about groups that are mostly or entirely above or below the overall median (and, ideally, that this implies that the medians are not all the same for the different treatments).

(e) (2 marks) What would be your recommendation for the kind of safelight to use, bearing in mind the purpose of a safelight? Explain briefly. (You might like to look back at the boxplots.)

Solution: The purpose of a safelight is to affect the plants as little as possible: that is, to make as small a difference from darkness as possible. This is so that an experimenter can do whatever needs to be done without otherwise affecting the plants. I think this is the best approach, but I also decided (eventually: I changed my mind on this) to accept "the light that makes the plants grow tallest". This is not the best, because you *don't* want a light that makes the plants grow dramatically taller than they would grow under darkness, but given the kind of results we had here, it seems a reasonable view to take.

Looking back at the boxplots in Figure 15, D, the control group (darkness) is over on the right. The typical heights of the plants in the BH and BL groups are very similar to D: that is, safelight B has a very small effect on the height compared to darkness. Safelight A, on the other hand, reduces the height noticeably and consistently (by 2 or 3 inches), and so this is having some kind of undesirable effect on height (the plants are being affected by safelight A, so that it is not really a "safelight" at all).

These conclusions are the same whether high or low intensity of the light is used, though light A at high intensity makes an especially big difference, and so is especially bad. I didn't ask you to choose an intensity, though I didn't penalize you if you did. You can make a case either for BL or BH: BL because it gives more consistent results (while being close to D), or BH because it gives results of about the same variability as D.

This kind of comparison, of treatments with a control rather than each other, is not something we have studied in this course. The logic of it should be clear enough, however. If we were doing an analysis of variance, we would follow up not with Tukey but with something like Dunnett's method. But that is another story (below). Usually when you're comparing something to a control, you're doing something like a drug trial where control is "placebo", and you want to be *significantly better* than the control. This one, though, is different in that you want to be *not significantly different* from darkness; you want to *not* reject the null hypothesis of being equal to control. (In the language of drug trials, this is called "equivalence testing".) Anyway,

the story I promised you:

Charles Dunnett was a Canadian statistician who worked at McMaster, and developed the test that bears his name in 1955. The idea is to compare each treatment with a control, while allowing for the fact that you are doing more than one test at once. This is a similar idea to Tukey, but instead of comparing each group with each other group, you *only* compare each treatment with the control, which is a smaller number of comparisons.

Dunnett's procedure can be done in R using the `multcomp` package, which also does Tukey and tests of a "general linear hypothesis" about means (including such things as contrasts, which those of you taking D29 will see later).

Let's grab the data so we can see how it goes (2018: note the old-fashioned code here):

```
safelight=read.table("safelight.txt",header=T)
str(safelight)
```

```
## 'data.frame': 40 obs. of  2 variables:
##  $ treatment: Factor w/ 5 levels "AH","AL","BH",..: 5 5 5 5 5 5 5 5 2 2 ...
##  $ height   : num  32.9 36 34.8 32.4 32.8 ...
```

The first thing to do is to see what levels our `treatment` factor has:

```
levels(safelight$treatment)
```

```
## [1] "AH" "AL" "BH" "BL" "D"
```

These are, as ever, alphabetical order, so that the first one alphabetically is the baseline, and will be treated (wrongly) as the control group unless we stop that from happening. We want Darkness or `D` to be the control group, so let's create a second Treatment variable with the right baseline:

```
safelight2=mutate(safelight, trt2=relevel(treatment,ref="D"))
levels(safelight2$trt2)
```

```
## [1] "D"  "AH" "AL" "BH" "BL"
```

What `relevel` does is to create a copy of the old factor, but with the factor levels shuffled so that the desired level is listed first and will be the reference or control group for Dunnett's test.

This is where a suspension of belief happens: we said that an ANOVA was a bad idea for these data, but we are going to do it anyway, purely so that we can illustrate how Dunnett's test works. So, `aov`, the same as appeared in the booklet of code and output:

```
height.1=aov(height~trt2,data=safelight2)
summary(height.1)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## trt2         4  78.94   19.73   24.07 1.24e-09 ***
## Residuals   35  28.69    0.82
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The groups are not all the same, so there are differences to be discovered. Normally, we'd follow up with Tukey here, since we are typically looking for any groups that differ from any other groups, but here we have a narrower focus: which of the four treatments differ from the control?

Here we go, since we organized things so that in `trt2` the desired control group came first:

```
library(multcomp)
```

```
## Loading required package:  mvtnorm
```

```
## Loading required package:  survival
```

```
## Loading required package:  TH.data
```

```
## Loading required package:  MASS
```

```
##
## Attaching package:  'MASS'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##    survey
```

```
## The following object is masked from 'package:dplyr':
##
##    select
```

```
##
## Attaching package:  'TH.data'
```

```
## The following object is masked from 'package:MASS':
##
##    geyser
```

```
height.2=glht(height.1,linfct=mcp(trt2="Dunnett"))
summary(height.2)
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Dunnett Contrasts
##
##
## Fit: aov(formula = height ~ trt2, data = safelight2)
##
## Linear Hypotheses:
##             Estimate Std. Error t value Pr(>|t|)
## AH - D == 0  -3.1800     0.4527  -7.024   <0.001 ***
## AL - D == 0  -1.8700     0.4527  -4.131   <0.001 ***
## BH - D == 0   0.2725     0.4527   0.602    0.933
## BL - D == 0   0.3425     0.4527   0.757    0.863
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

This is saying "compare the levels of `trt2` using Dunnett's test". If you wanted to do Tukey you would replace `Dunnett` with `Tukey`.

The output compares each treatment group with the control group D, and reports a P-value (properly adjusted for having done 4 tests at once). The two safelight B groups are not significantly different from the control `D`, but the two safelight A groups very much are different. Since we were looking for something equivalent to darkness, safelight B is definitely the one to choose.

If you want confidence intervals for the differences in means, you can get those too. `confint` is a general-purpose getter of confidence intervals for almost anything:

```
confint(height.2)
```

```
##
##   Simultaneous Confidence Intervals
##
## Multiple Comparisons of Means: Dunnett Contrasts
##
##
## Fit: aov(formula = height ~ trt2, data = safelight2)
##
## Quantile = 2.5585
## 95% family-wise confidence level
##
##
## Linear Hypotheses:
##             Estimate lwr     upr
## AH - D == 0 -3.1800  -4.3383 -2.0217
## AL - D == 0 -1.8700  -3.0283 -0.7117
## BH - D == 0  0.2725  -0.8858  1.4308
## BL - D == 0  0.3425  -0.8158  1.5008
```

This shows how much safelight A decreases plant height on average: between 2 and 4 inches at high intensity and between 1 and 3 at low.

All of this is to be taken with a grain of salt, of course, because we shouldn't have done the ANOVA in the first place, never mind the Dunnett's test to follow it up. But this, at least, shows how it goes.

There is also a "Welch ANOVA", (2017: you might have thought of this) which is ANOVA where the groups are allowed to have different spreads (a generalization of the Welch-Satterthwaite t-test). The test is simplicity itself, with the same `var.equal` thing that `t.test` has:

```
oneway.test(height~treatment,data=safelight,var.equal=F)
```

```
##
##  One-way analysis of means (not assuming equal variances)
##
## data:  height and treatment
## F = 92.145, num df = 4.00, denom df = 16.93, p-value = 2.941e-11
```

This test is believable, because we have normal data and we are *not* assuming equal spreads any more. It has an even smaller P-value than the ANOVA we didn't trust. So there really are differences among the plant heights. (The denominator degrees of freedom is fractional in the same way that the DF for the Welch-Satterthwaite t-test are fractional: the value is chosen to make the variance of the F-statistic come out right.)

It seems to me that this flavour of ANOVA deserves to be a lot better known. (2018: and thus it appears in the course now.)

The standard Tukey-like followup to Welch ANOVA is called Games-Howell. This is not widely available, but there is a package called `userfriendlyscience` that has it. (2018: `userfriendlyscience` takes me a long time to install because there is a lot of C++ code to compile, so I have come to prefer the other package whose name I can never remember.)

The advantage to Games-Howell is that it does not assume equal variances.

```
library(userfriendlyscience)
attach(safelight)
oneway(height, treatment, corrections=TRUE,
  posthoc="games-howell", levene=TRUE)
```

```
## ### Oneway Anova for y=height and x=treatment (groups: AH, AL, BH, BL, D)
##
## Omega squared: 95% CI = [.52; .8], point estimate = .7
## Eta Squared: 95% CI = [.55; .79], point estimate = .73
##
##                                     SS Df    MS     F     p
## Between groups (error + effect) 78.94  4 19.73 24.07 <.001
## Within groups (error only)      28.69 35  0.82
##
##
## ### Levene's test for homogeneity of variance:
##
## F[4, 35] = 19.33, p < .001.
##
## ### Post hoc test: games-howell
##
##        diff ci.lo ci.hi     t    df     p
## AL-AH  1.31  0.68  1.94  6.55 13.47 <.001
## BH-AH  3.45  2.01  4.90  8.15  8.36 <.001
## BL-AH  3.52  2.96  4.09 19.48 14.00 <.001
## D-AH   3.18  1.25  5.11  5.72  7.77  .003
## BH-AL  2.14  0.69  3.60  4.95  9.01  .005
## BL-AL  2.21  1.58  2.84 10.98 13.56 <.001
## D-AL   1.87 -0.07  3.81  3.32  8.14  .059
## BL-BH  0.07 -1.38  1.52  0.17  8.41 1.000
## D-BH  -0.27 -2.40  1.85  0.40 12.96  .994
## D-BL  -0.34 -2.28  1.59  0.62  7.79  .968
##
## ### Welch correction for nonhomogeneous variances:
##
## F[4, 16.93] = 92.14, p < .001.
##
## ### Brown-Forsythe correction for nonhomogeneous variances:
##
## F[4, 16.22] = 24.07, p < .001.
```

```
detach(safelight)
```

This shows the original ANOVA. At the bottom, it shows the Welch ANOVA (the same results as `oneway.test`). Levene's test is a test of whether all the variances are equal (which, as you would expect, is roundly rejected and tells you that Welch is the right way to go). (There are different tests of whether the groups have the same variance; the one in `var.test` and Bartlett's test rely perhaps too much on the data being normal, and so Levene's test is the one that people like these days.)

Then comes the Games-Howell, that looks a lot like Tukey, with P-values for comparing each

pair of groups. The wisdom is that you should only use Games-Howell if you need it (that is, if the groups have the same spread, Tukey is better). Also, here, we really only wanted to compare with group D, so we would do better to use a more focused test that *only* compares with D (except that I don't know what that would be!). A rough-and-ready approach here would be to ignore the comparisons that don't include D. If you do that, the only significant difference is between D and AH; the D-AL difference is not quite large enough to be significant, in contrast to what the (wrong) Dunnett's test told us before.

You can see that this is a real rabbit-warren, but there are two major issues being confounded here:

1. With unequal spreads but acceptably normal data, use the Welch version of the ANOVA (eg. from `oneway.test`). To follow up, instead of Tukey, use Games-Howell.
2. To compare each treatment only with a control, use Dunnett's test rather than Tukey. But this only applies to the standard ANOVA where the groups have equal variance. If you want a Dunnett-like test to follow up the Welch ANOVA, you are (at the current state of my knowledge) stuck.

The more I write, the more I realize I still have to learn (and teach you guys!) 2018: note that this is how things find their way into the course, as Welch's ANOVA did this year. I realize that we need to learn things to answer questions properly, so I make sure that we do, which means finding out about them myself first.

6. The salaries of a number of employees at a company were recorded. Along with the employee's salary, three other variables were recorded: the level of education attained by the employee (1 is Bachelor's, 2 is Master's, 3 is PhD), the number of years of experience after completing their degree, and the number of employees supervised in the employee's current position. The aim is to see which, if any, of these three variables helps to predict salary.

   The data are read in and some of the values are shown in Figure 18.

   Note that throughout this question, the variable `degree` is treated as being numeric (so that each extra level of education is worth one extra "point", so to speak). You may assume that this is a reasonable way to handle `degree` for these data.

   (a) (3 marks) A regression is run for predicting salary from the other three variables. The output and graphs are shown in Figures 19 and 20. Do you think this regression is appropriate to describe the relationship between salary and the other variables? Explain briefly, referring to items in the output as necessary. You should assess all the relevant items in the output.

Solution: This means to assess the residual plots and look for any problems. Specifically:

- The plot of residuals against fitted values appears to show some fanning out. This effect also appears on the plot of residuals against experience.
- The normal quantile plot of residuals looks more or less normal.
- The plot of residuals against degree looks a bit less spread out for degree 1 than for the other degree values.
- The plot of residuals against supervised looks more or less random, though there are some very high numbers of employees supervised. (This is a feature of the data, not a problem with the regression.)

In summary, the regression is not appropriate because of the fanning out. This suggests a

transformation of the response variable `salary`. (If you read my assignment solutions, you'll have some sense of what might be a good transformation, even before you look at the next part.)

I didn't insist on all of these. For example, noting "fanning out" would get you 2 points, and making some other intelligent comment, like examining the normal quantile plot and commenting on it, or examining one of the other plots, or stating that a transformation would be good or that the regression could be improved, would get you the third one. Any evidence that you have looked at the plots in Figure 20 and said something vaguely sensible should get you something.

Looking at R-squared or the F-test or the t-tests is not what I wanted here. I wanted you to critique the model: to say whether this regression is good enough or should be improved. The model is good (as R-squared etc. would tell you), but the important point is that we should do better (as the residual plots tell you).

I think you need to be able to look at the top-left plot in Figure 20 and see "fanning out" (or some other equivalent words), but I was fairly relaxed about letting you conclude what you wanted from the other plots, since the most important thing for this question is that you *looked at* them.

(b) (2 marks) Some output from `proc transreg` is shown in Figure 21. Looking at this output, what do you conclude?

Solution: Looking at the bottom graph, not the rather confusing top one, we see that lambda is estimated to be in a small interval around zero: that is, we should use a log transformation of salary.

I wanted some kind of hint in your answer about what variable should have its log taken. "Salary" or "y" will do, as will the word "transformation" somewhere in your answer (since transformations, for us, are transformations of the response variable).

I figured out what the top graph is. It takes the explanatory variables that have P-values less than 0.05 in the log-transformed regression, which is all three of them here. It then makes a picture of how their significance changes as `lambda` changes. For example, the green curve is highest at `lambda=1`, which means that `supervised` is the most significant variable in the original regression. But when `lambda=0`, the best transformation, `supervised` is actually the *least* significant variable because its curve is the lowest (though still strongly significant). You can check this by looking at the size of the t-value for each explanatory variable in each regression. If you care to do so. There's not much point in doing so here, where everything is always strongly significant.

(c) (2 marks) The analyst decides to predict log of salary from the other variables. (This may or may not have been suggested by any of the previous output.) The output is shown in Figure 22 and Figure 23. Is there anything that would cause you to doubt the appropriateness of this regression? Explain briefly.

Solution: This invites you to check all the same graphs again on the new output. I would say that there is less evidence of fanning-out than there was before, on the plots of residuals against fitted values and against experience. The residuals look a lot more normally-distributed than they did before: the normal quantile plot is almost perfectly straight. The other plots look about as random as you would expect. So I think this regression is appropriate. You are welcome to disagree, if your disagreement is based on something reasonable.

You can compare the two R-squared values. It happens here, in fact, that the regression that is more appropriate from the point of view of the residuals happens to have a slightly *smaller* R-squared, but that is not an issue of whether or not you have the right form of relationship. The best use of R-squared here is to say that "R-squared has decreased, but not much", while saying something about the residual plots. It's perfectly fine to say that the residual plots are no better than they were before (I happen to disagree, but any justifiable opinion is fine), in which case your viewpoint is that this regression is no better than the untransformed one. Transformations usually help, but they are not a "magic bullet": sometimes there will still be obvious problems.

This part was pretty easy to get something for: all you had to do was to write something relevant and not obviously nonsense!

(d) (3 marks) Think about what you know or can guess about the type of relationship being investigated here. For the regression in Figure 22, would you expect the slopes to be positive or negative? Are they? Explain briefly.

Solution: It seems as if having a higher degree, having more work experience or supervising more employees should all be associated with a higher salary (or, as in this regression, a higher log-salary, but that comes to the same thing, since log is an increasing function). So we'd expect all three slopes to be positive, and they all are, significantly so.

Two marks for saying what you'd expect and why, one mark for saying whether the slopes actually were what you expected.

A few people appear to have gotten confused with what "employees supervised" meant. I meant "people that the employee in question is the boss of", so that having more of those ought to increase your salary. If you thought I meant "number of employees doing the same job as the employee in question", then you would expect the slope to be *negative* since the employee we are thinking about has more competition (or something like that). If your reasoning seemed sound (and you expressed surprise that the actual slope was positive), I was happy with that.

(e) (2 marks) What, if anything, would you do next, after the regression in Figure 22? Your choices are (i) to remove one or more explanatory variables, (ii) to use a different transformation than the analyst did, (iii) declare yourself satisfied with the regression. Explain briefly.

Solution: I'd declare myself satisfied with the regression in Figure 22. I didn't find any problems with it above. All of the explanatory variables are strongly significant, meaning that removing any of them would be a mistake (ruling out (i)), and the log transformation was the one suggested by Box-Cox, ruling out (ii). The Box-Cox suggested that no transformations

other than log were sensible.

If your answers were different from mine above, your answer here should be consistent with what you said before. For example, if you decided that the square root was the right transformation from the Box-Cox, you should disagree with the analyst's choice of log. Or if you thought the log transformation had not fixed the residual plots, you were entitled to try another transformation. Or, if you think "transformation" includes re-expressing the x-variables, you could suggest that too. I think `supervised` (see below) is the prime candidate for this treatment, since there are a few very large values.

In the textbook from which I got these data, degree was treated as a categorical variable (with two indicator variables, as in STAC67). That would probably be better than doing what I did, which was to assume that each "step up" in degree was worth the same increment in log-salary, but I didn't want to confuse the issue here with that. I stated this in the question to dissuade you from suggesting that as something to do next.

I also thought about transforming `supervised` because there were a few very big values, but the large number of zeroes meant that I couldn't take logs. So I didn't. (Taking log of `supervised` plus one is possible.) There is one high-leverage point (top right of the nine graphs in Figure 23), which I'm guessing is that employee who supervises over 40 other employees. Curiously, if I have that individual right, the residual is one of the more negative ones, not near zero as is usually the case for an influential point.

I am suspecting that there is some positive correlation among the explanatory variables: someone who supervises a lot of employees is probably also going to be someone with a lot of experience. But the tininess of the P-values indicates that each explanatory variable adds to the prediction, over and above the influence of the others, so that taking even one of them out is a bad idea.

As the afternoon wore on, I found myself getting more relaxed about what I would accept, so your best bet was to write *something* in response to this question, and there was likely some points in it.

7. Back in 2000, an instructor at the University of California Davis collected some information about the students in her class. There were a lot of variables, but we will focus on these:

   - `Height`: the student's height (inches)
   - `GPA`: the student's GPA (4-point scale)
   - `Sex`: whether the student identified as `Male` or `Female`
   - `Alchol`: the number of alcoholic drinks consumed in the past week
   - `momheight`: the height of the student's mother (inches)
   - `dadheight`: the height of the student's father (inches)

   The variable name `Alchol` is mis-spelled. That's how it was in the original data. I wonder if the instructor had been drinking too much alchol.

   Some of the relevant data are shown in Figure 24.

   (a) (2 marks) What does the code in Figure 25 do? Explain briefly.

   Solution: Keeps only the rows that have no missing data in any of the columns `GPA`, `Alchol`, `momheight`, `dadheight` and saves the result in `davis3`. "Gets rid of the missing values" is a just-about-acceptable answer to the first thing, since these are actually the only ones of our columns to have any missing values in them.

   The logic of the `filter` is "take the rows for which `GPA` is not missing, and take the rows for which `Alchol` is not missing", and so on. So none of the columns should have any missing values in them, in the data frame `davis3`. In `dplyr` terms, `filter` says which rows to *include*, so if you want to say which rows to *exclude*, you'll need to use words like "filter out".

   The printout of the first 20 lines of data above was meant to offer you a clue to this, since they have some missing values in them, and missing values and regressions don't mix. (The code to print out the first 20 lines was entirely separate from the pipe that you had to assess here. Did you look at the right Figure?)

   The first time I did this, I didn't worry about the missing values, and the two regressions (below) both ran, by taking out any rows having any missing values on any of the variables in the regression. But they ran on *different sets of observations*, since some of the observations have missing values only on `Alchol`, which was in the first regression but not the second. This made comparing the regressions problematic, so I figured I'd better take care of the missing values properly.

   (b) (2 marks) Figure 26 shows a regression predicting students' height from the other variables. Based on this output, what regression would you fit next? Explain briefly. (If you are happy to stop at this regression, say so, but you should also offer an explanation.)

   Solution: A regression with all the same explanatory variables except for `Alchol`, since that is the non-significant one with the highest P-value. The right thing to do is to take out only *one* x-variable, since you don't know what is going to happen to the P-value for `GPA` when you take out `Alchol`.

   If your answer is to take out *both* `Alchol` and `GPA`, since they're both non-significant, you should add that you will test whether you should have removed them both (that's what the `anova` is doing below). The P-values only relate to that *one* x-variable in the context of the regression

with that variable and the others in it, and don't tell you what's going to happen if you take out more than one.

The one way you can get away with taking out both is to assert that "obviously these have nothing to do with height". (This is "expert knowledge", of a sort.) But I think it's still smart to test whether you should have removed them both.

(c) (2 marks) Look at the regression in Figure 27, which may or may not be your preferred regression from part (b), and the output in Figure 28 below it. What do you conclude from Figure 28? Explain briefly.

Solution: This is comparing the two models, one with all five x-variables, and the one with three (having removed the non-significant ones). The null hypothesis is that the two models fit equally well, and that null is not rejected (1 point), so we should prefer the smaller model (predicting height from `Sex` and the two parents' heights) (the second point).

This is of course entirely reasonable, since I put `GPA` and `Alchol` in the first model precisely *because* I expected them to have no impact on height. We know that males are taller than females, and we'd expect a student with taller parents to be taller themselves, so there is no surprise here.

An additional clue is that the first regression has a larger R-squared (as it must), but a slightly *smaller* adjusted R-squared, also indicating that taking those two irrelevant variables out was a good idea.

To get the second point, you need to go beyond the null hypothesis (models equally good) not being rejected to say which model to go with (the one without GPA and alcohol, because it is simpler). You can short-cut this, but you have to be very careful, otherwise you might end up with a null hypothesis "the smaller model is better than the bigger one" that doesn't look like a null hypothesis. It's better to take all the steps, since that shows me that you know what you're talking about without any doubt (if you get it right, of course).

(d) (2 marks) Figure 29 shows a residual plot for the regression in Figure 27. Do you see (i) no problems, (ii) one problem, (iii) two or more problems? Explain briefly.

Solution: I see at least one problem: the point top left with a large (positive) residual and a small fitted value. (If you missed the outlier, you will be lucky to get any points!)

I think the rest of the points are randomly scattered, for reasons explained in a minute. They appear to be in the bottom right purely because of that outlier; if you take away the outlier, they would fill the plot and be more or less evenly scattered above and below zero.

I am also fine with you noting the diagonal downward-sloping lines of points as a problem also, since we haven't seen this before.

This occurs, not because of a problem with the regression, but because people usually give their heights as a whole number of inches, so that there is a smallish, discrete set of possible heights that people can have. The points follow a diagonal line because each observed height goes with a different combination of values of the other variables, and thus a different predicted height. (This is one of those things that, once you've seen it, is hard to "unsee", and interferes with your judgement about whether the points are randomly scattered.)

Another thing that happened is that there are two "clusters" of points on the right. One student

described them as "lungs". I didn't actually notice this at first, but they happen because the fitted heights for female students are almost all less than the fitted heights for males, so there appear to be two groups with, almost, a gap between. This, though, is a feature of the data rather than a problem with the regression; the residuals (apart from that top left one) appear properly scattered about 0 in both groups (see below), so that the heights for both genders are being properly predicted. If you saw a slight downward trend on the plot, it's because of that outlier.

Is there something funny about that observation with the large residual? Let's see if we can find out:

```
davis2=read.csv("davis2.csv",header=T)
davis2 %>% filter(!is.na(GPA),
                  !is.na(Alchol),
                  !is.na(momheight),
                  !is.na(dadheight)) -> davis3
height.1=lm(Height~Sex+GPA+Alchol+momheight+dadheight,data=davis3)
height.2=update(height.1,.~.-GPA-Alchol)
d=with(davis3,data.frame(Sex,momheight,dadheight,Height,
  r=resid(height.2),f=fitted(height.2)))
head(d)
```

```
##      Sex momheight dadheight Height         r        f
## 1 Female        64        70     64 -1.160163 65.16016
## 2   Male        67        68     69 -1.653528 70.65353
## 3 Female        62        68     63 -1.247470 64.24747
## 4   Male        66        69     72  1.297556 70.70244
## 5 Female        68        69     67  1.277608 65.72239
## 6   Male        67        69     69 -1.906159 70.90616
```

This reads in and processes the data, re-fits the two models, then creates a data frame `d` that has all the variables that were in the (smaller) regression, plus the residuals and fitted values from that regression.

Now we can plot the residuals against fitted values, colouring the points by `Sex`:

```
ggplot(d,aes(x=f,y=r,colour=Sex))+geom_point()
```

And so you see that the larger fitted values are all males, and the smaller ones are almost all females.

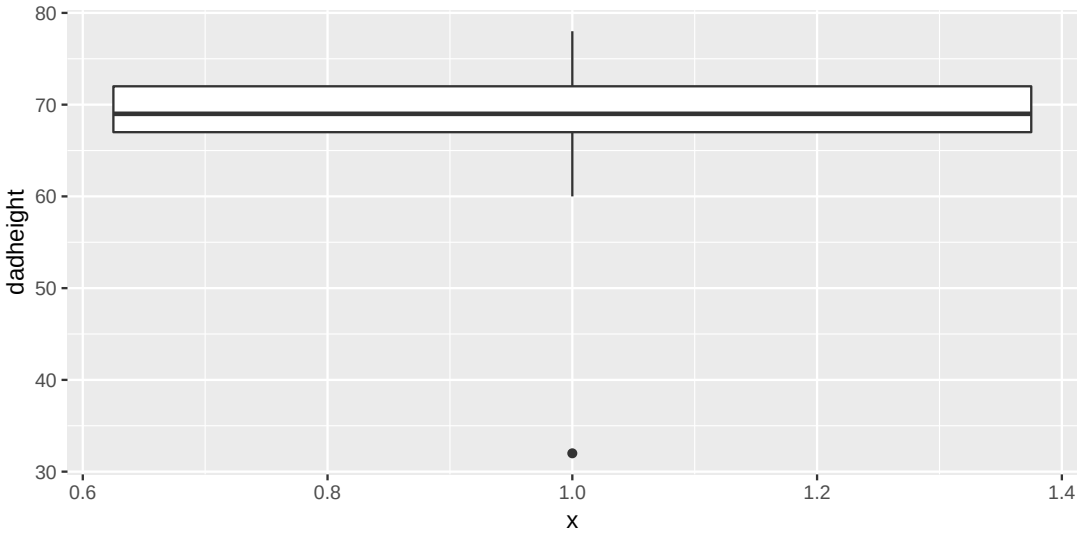
Then we can find the observation with residual greater than 10 thus:

```
d %>% filter(r>10)
```

```
##    Sex momheight dadheight Height        r        f
## 1 Male        42        32     69 12.53407 56.46593
```

This is a male student, nearly 6 feet tall, whose mother is claimed to be 3 ft 6 inches tall, and whose father is 2 ft 8 inches tall! No, I don't believe this either. This is an observation that should have been removed before we fit any models, because it cannot be correct. It's so crazy that something as simple as boxplots of the mothers' and fathers' heights would have found it quickly:

```
ggplot(d,aes(x=1,y=dadheight))+geom_boxplot()
```



There it is, down at the bottom. The other dads' heights look sensible, but that one is absurd.

8. A manager has data on sales of asphalt roofing shingles from data in 26 sales districts. For each district (numbered 1–26), the recorded variables are:

   - the annual sales (in thousands of dollars),
   - the promotional expenditures (thousands of dollars),
   - number of active accounts,
   - number of competing brands,
   - district potential (a coded value, but a higher value indicates more sales potential).

   The data are shown in Figure 30. The manager is interested in seeing whether any of the data values recorded are possible errors. 2018: skip this question.

   (a) (1 mark) What is the name of the quantity calculated in the first three lines of Figure 32?

   Solution: Leverages. Not "hatvalues", since that was already in the R code.

   (b) (2 marks) What is the *purpose* of the calculation in the last line of Figure 32?

   Solution: Any leverage bigger than the value calculated there indicates an outlier observation (one that might be an error).

   The way I phrased the question, "to find outliers" is a just-about-acceptable answer. I was more interested in what we will use this calculation for than what the things in the formula represent.

   (c) (2 marks) Which sales districts are possible errors, according to the criterion given here?

   Solution: None of them. (This was kind of a trick question, but I want you to have confidence in what you are doing.) This is because none of the leverages are bigger than 0.46.

   If you have said somewhere that a "leverage bigger than 0.46 indicates an outlier", you don't need to say it again here. But you should say it somewhere.

   An alternative (perfectly good) way to tackle this question is to say that districts 8 and/or 21 are close to the cutoff, and deserve examination (this is looking forward to the next part). Naming one or other of those districts without explanation, or doing something else sensible but not completely correct, is probably worth one point. (There are lots of ways to get one point out of me in general: if I would describe what you wrote as "vaguely plausible and relevant", I try to find one point for it. But if you leave a question blank, I cannot give you anything other than zero.)

   (d) (4 marks) For the observation with the most extreme value from the output in Figure 32, describe what made it come out that way, referring to Figure 31 as necessary, and using what you know or can guess about sales of a product.

   Solution: "Most extreme" means the *largest* leverage. This is district 8, with a leverage of 0.44. (This is not quite large enough to be deemed an outlier, but we proceed anyway.)

   What you do is to compare each of the variable values for this observation with the summary in Figure 31, and see which values are unusually high or low:

Question 8 continues... This page: _____ of possible 9 points.

- `sales` 291.9 higher than third quartile
- `promotion` 9 is highest
- `active` 56 is between median and Q3, high but not unusually high
- `competing` 5 is *less* than Q1, almost the minimum
- `potential` 4 is likewise less than Q1, close to minimum.

This is a high-sales district. I would have expected `promotion` and `active` to be high along with that (given what I know about sales, which is not much), which they are. I would expect the number of competing brands to be small (it is), because if you have fewer brands competing with yours, you would expect to sell more. I would expect the potential for sales to be *large*, but it is actually very small. If this is correct, the salespeople are making sales against the odds!

I think having a small potential with the other combination of variable values is what makes this observation the most unusual one, even though it is not quite an outlier. This, to my mind, (using some knowledge or guesswork about what would make sales high) is the best answer, though the hard work of comparing district 8's variable values with the summary is worth most of the points.

Marking here: 1 point for picking out an unusual district (8 or 21), 2 points for looking at each of the variables and comparing with the summary so that you know whether they are high or low, and 1 for making an intelligent and supported sales-related comment about why this district is unusual. You can short-cut this by picking out the right combination of variables off the top, but you are taking a large risk: if you get that wrong, you'll lose some of the credit for examining the variables as well as for your sales-related comment. (I was OK with picking district 21, the second-highest leverage, since it's hard to scan that list under exam pressure. That district has low sales and potential, high competing brands, average active accounts but, oddly, *high* promotion.)

I should come clean here and say that, in the original data, `potential` for district 8 was actually 10 rather than 4; in other words, it was above average as you would expect rather than below. (This district still had the largest leverage, though less than what you see here.) I thought that introducing this error would make this district come out with a leverage higher than the cutoff, but it didn't quite. (This is a kind of type II error for outlierness: the data value was actually wrong, but it wasn't flagged as wrong, as it should have been.)

9. The data in Figure 33 are (some of the) results from the English Premier soccer league for the 2016-2017 season, which started in August 2016. The columns are:

   - The date of the match, year first, then month, then day.
   - The name of the home team (text, 27 characters long including the trailing spaces)
   - The name of the away (road) team (likewise, text, 27 characters long including the trailing spaces)
   - The number of goals scored by the home team
   - The number of goals scored by the away team

   There are 130 lines of data altogether.

   The columns are separated by commas. Imagine that the (entire) data set has been uploaded as `england.csv` to your file storage on SAS Studio, and also to your working folder on R Studio.

   (a) (4 marks) 2016: Give a SAS data step to read the data into a SAS data set, so that the variables are read in properly and are of the appropriate types. (By "data step", I mean SAS code starting with `data` and the name of a data set.) 2017: skip this; see the next part.

Solution: 2016: You need appropriate informats to read in the data: a date one for the dates, and text ones of the right length to read the team names. The variable and its informat need to be separated by colons (or else the data won't get read properly: I tried). Plus, the data start on the second line of the file and are separated by commas, so the `infile` has to account for this:

```
data england;
  infile '/home/ken/england.csv' firstobs=2 dlm=',';
  input date: yymmdd10. team1: $27. team2: $27. s1 s2;
```

As ever, the names you use for dataset and variables don't matter, but it is helpful to use names that remind you of what they contain.

Did it work?

```
proc print data=england(obs=10);
  format date date8.;
```

Obs	date	team1	team2	s1	s2
1	13AUG16	Southampton	Watford	1	1
2	13AUG16	Middlesbrough	Stoke City	1	1
3	13AUG16	Everton	Tottenham Hotspur	1	1
4	13AUG16	Manchester City	Sunderland	2	1
5	13AUG16	Crystal Palace	West Bromwich Albion	0	1
6	13AUG16	Burnley	Swansea City	0	1
7	13AUG16	Hull City	Leicester City	2	1
8	14AUG16	Arsenal	Liverpool	3	4
9	14AUG16	AFC Bournemouth	Manchester United	1	3
10	15AUG16	Chelsea	West Ham United	2	1

It did. I used a different output date format from the input one because, well, I could.

The colons are necessary in the input line because:

```
data england2;
  infile '/home/ken/england.csv' firstobs=2 dlm=',';
  input date yymmdd10. team1 $27. team2 $27. s1 s2;
```

The output is now:

```
proc print data=england2(obs=10);
  format date date8.;
```

Obs	date	team1	team2	s1	s2
1	13AUG16	,Southampton	,Watford	.	1
2	13AUG16	,Middlesbrough	,Stoke City	.	1
3	13AUG16	,Everton	,Tottenham Hotspur	.	1
4	13AUG16	,Manchester City	,Sunderland	.	2
5	13AUG16	,Crystal Palace	,West Bromwich Albion	.	0
6	13AUG16	,Burnley	,Swansea City	.	0
7	13AUG16	,Hull City	,Leicester City	.	2
8	14AUG16	,Arsenal	,Liverpool	.	3
9	14AUG16	,AFC Bournemouth	,Manchester United	.	1
10	15AUG16	,Chelsea	,West Ham United	.	2

That didn't work: commas in the names, and the second score column got missed off entirely. (The value in `s2` is actually the *home* team's score, as you can check.)

Marking: basically minus one per mistake, with proviso that having one correct informat plus the `dlm` and the `firstobs` guarantees you 2, and having otherwise sound code but with no informats at all gets you 1. (The same mistake in more than one place only counts once — for example, missing a colon counts as one mistake no matter how many colons you miss, even though there should be three altogether.) Since the informats were the point of this question, I needed to insist on them being correct.

(b) (3 marks) Write a `proc import` statement to read in the same data.

Solution: I have red pens again now.

This is easier because you can let SAS guess what everything is. I've added a `proc print` for checking, but I didn't ask you to:

```
proc import datafile='/home/ken/england.csv' out=england3 dbms=csv replace;
  getnames=yes;

proc print data=england3(obs=10);
```

```
Obs          date    team1                  team2

  1    2016-08-13    Southampton            Watford
  2    2016-08-13    Middlesbrough          Stoke City
  3    2016-08-13    Everton                Tottenham Hotspur
  4    2016-08-13    Manchester City        Sunderland
  5    2016-08-13    Crystal Palace         West Bromwich Albion
  6    2016-08-13    Burnley                Swansea City
  7    2016-08-13    Hull City              Leicester City
  8    2016-08-14    Arsenal                Liverpool
  9    2016-08-14    AFC Bournemouth        Manchester United
 10    2016-08-15    Chelsea                West Ham United

Obs             s1             s2

  1              1              1
  2              1              1
  3              1              1
  4              2              1
  5              0              1
  6              0              1
  7              2              1
  8              3              4
  9              1              3
 10              2              1
```

Everything is as it should be. The dates have been printed out in the same format as they were read in (we discovered on one of the assignments that these really are dates in `yymmdd10.` format, and not just text that happened to look like dates).

It was pretty evident that you either knew how `proc import` worked, in which case you would get 2 (if you messed up one of the inputs) or 3, or you didn't (a fast zero).

(c) (2 marks) Starting from the data set you read in with `proc import`, what code would save it as a permanent data set (on disk)?

Solution: 2017 and on: I may not get to this; if so, don't worry about it.

2016: We have to clone the data set (using a data step), creating a permanent data set as we do:

```
data '/home/ken/england';
  set england3;
```

with a name of your choosing, but the permanent data set must look like a filename with the quotes (and no extension, because SAS will supply one).

If you insist, you can have a `data` line like mine followed by the `infile` and `input` that you had before (this is not really using the data set you read in with `proc import`, but I was prepared to let that go). Or you can use all your `proc import` code again but replace the `out=` line with something like `out='/home/ken/england'` (this is where the "thing in quotes" has to go). In either case, you don't have to write everything out again; the easiest is to say "repeat (a) with `data=soccer` replaced by `data='home/ken/soccer'`" or "repeat (b) with `out=soccer` replaced

by `out='/home/ken/soccer'`" as appropriate. I never have any issues with you referring me back to something you've already done and telling me how to change it.

The `replace` thing in `proc import` does *not* create a permanent data set; the data set you created with `out=` in (b) disappears just the same as any other SAS data set when you exit SAS. (Which makes me think: if you actually wrote (b) to create a permanent data set, that would answer both (b) and (c), and your answer to (c) could be just "see (b)".)

There was a point available here for any inkling of relevant code, for example something in quotes that looked like a permanent data set even if the rest of the code to create it wouldn't work.

(d) (2 marks) 2017: we might not have done this either. 2016: Imagine you have closed down SAS and started it up again. What code would display the first 8 rows of the permanent data set that you created in the last part, *without* using a `data` step or `proc import`?

Solution:

`proc print` with a data set and an `obs`:

```
proc print data='/home/ken/england'(obs=8);
```

with output

Obs	date	team1	team2
1	2016-08-13	Southampton	Watford
2	2016-08-13	Middlesbrough	Stoke City
3	2016-08-13	Everton	Tottenham Hotspur
4	2016-08-13	Manchester City	Sunderland
5	2016-08-13	Crystal Palace	West Bromwich Albion
6	2016-08-13	Burnley	Swansea City
7	2016-08-13	Hull City	Leicester City
8	2016-08-14	Arsenal	Liverpool

Obs	s1	s2
1	1	1
2	1	1
3	1	1
4	2	1
5	0	1
6	0	1
7	2	1
8	3	4

This has rather an unfamiliar look, since we haven't displayed the first few lines of a permanent data set before. But there's no surprise here: a permanent data set name in quotes, followed by the right `obs` in brackets. (I only asked you to display the first 8 lines to see if you were paying attention!)

Now, here is a point of exam technique (and an example of *me* being consistent): if you didn't know how to create a permanent data set above, you might think you are stuck for this part as well. But you are not: what you do is to ask yourself "how would I do this if I *had* been able to do the previous part?" The answer to that is that you would have a permanent data set with some name, `permo` say, and you might remember how to display the first 8 lines of that, like this: `proc print data=permo(obs=8)`. So you write that down and get 2 points. Or you use the name of a dataset you created earlier. The point is to show me that you *would* know what to do. (Of course, if you *could* create a permanent data set, you need to put *its* name after the `data=`. This is advice for what to do if you could not.)

Otherwise, if you printed the first 8 lines of the wrong data set, or forgot to display its first 8 lines of the right data set, you got 1 point.

(e) (3 marks) Now the same data in R. Give code that will read in the data of Figure 33 *as text rather than factors*, and will also create a data frame where the dates are stored as R Dates.

Solution: 2016: This goes in two steps: read in the data with `stringsAsFactors=F`, which will read in all the non-numbers as text, and then create a new data frame in which the dates-as-text have been converted to `Date`s:

```
england=read.csv("england.csv",header=T,stringsAsFactors=F)
str(england)
```

```
## 'data.frame': 130 obs. of  5 variables:
##  $ date : chr  "2016-08-13   " "2016-08-13   " "2016-08-13   " "2016-08-13   " ...
##  $ team1: chr  "Southampton               " "Middlesbrough             " "Everton
##  $ team2: chr  "Watford                   " "Stoke City                " "Tottenham Hotsp
##  $ s1   : num  1 1 1 2 0 0 2 3 1 2 ...
##  $ s2   : num  1 1 1 1 1 1 1 4 3 1 ...
```

```
england2=mutate(england,thedate=as.Date(date))
str(england2)
```

```
## 'data.frame': 130 obs. of  6 variables:
##  $ date   : chr  "2016-08-13   " "2016-08-13   " "2016-08-13   " "2016-08-13   " ...
##  $ team1  : chr  "Southampton               " "Middlesbrough             " "Everton
##  $ team2  : chr  "Watford                   " "Stoke City                " "Tottenham Hot
##  $ s1     : num  1 1 1 2 0 0 2 3 1 2 ...
##  $ s2     : num  1 1 1 1 1 1 1 4 3 1 ...
##  $ thedate: Date, format: "2016-08-13" "2016-08-13" ...
```

This is the easiest way, since the dates are year-month-day which is what `as.Date` expects, but you can also use `ymd` from `lubridate`:

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
england3=mutate(england,thedate=ymd(date))
str(england3)
```

```
## 'data.frame': 130 obs. of  6 variables:
##  $ date   : chr  "2016-08-13   " "2016-08-13   " "2016-08-13   " "2016-08-13   " ...
##  $ team1  : chr  "Southampton               " "Middlesbrough             " "Everton
##  $ team2  : chr  "Watford                   " "Stoke City                " "Tottenham Hot
##  $ s1     : num  1 1 1 2 0 0 2 3 1 2 ...
##  $ s2     : num  1 1 1 1 1 1 1 4 3 1 ...
##  $ thedate: Date, format: "2016-08-13" "2016-08-13" ...
```

Either of these work with pipes.

If you forgot `stringsAsFactors`, can you construct the text using `as.character`? Let's find out:

```
read.csv("england.csv",header=T) %>%
  mutate( t1=as.character(team1),
          t2=as.character(team2),
          d=as.Date(date)
        ) %>% str()
```

```
## 'data.frame': 130 obs. of  8 variables:
##  $ date : Factor w/ 35 levels "2016-08-13   ",..: 1 1 1 1 1 1 1 2 2 3 ...
##  $ team1: Factor w/ 20 levels "AFC Bournemouth           ",..: 13 12 6 10 5 3 7 2 1 4 ...
##  $ team2: Factor w/ 20 levels "AFC Bournemouth           ",..: 18 14 17 15 19 16 8 9 11 20
##  $ s1   : num  1 1 1 2 0 0 2 3 1 2 ...
##  $ s2   : num  1 1 1 1 1 1 1 4 3 1 ...
##  $ t1   : chr  "Southampton                " "Middlesbrough              " "Everton
##  $ t2   : chr  "Watford                    " "Stoke City                 " "Tottenham Hotsp
##  $ d    : Date, format: "2016-08-13" "2016-08-13" ...
```

Yes, that works. You'd probably want to get rid of the original `date, team1, team2` this way, but the variables we created are fine.

I also note that the team names have been read in with trailing blanks (all the blanks up to the comma in the original data file). The place to handle that is when we read in the data:

```
england.1a=read.csv("england.csv",header=T,stringsAsFactors=F,
  strip.white=T)
england2=mutate(england.1a,thedate=as.Date(date))
str(england2)
```

```
## 'data.frame': 130 obs. of  6 variables:
##  $ date   : chr  "2016-08-13" "2016-08-13" "2016-08-13" "2016-08-13" ...
##  $ team1  : chr  "Southampton" "Middlesbrough" "Everton" "Manchester City" ...
##  $ team2  : chr  "Watford" "Stoke City" "Tottenham Hotspur" "Sunderland" ...
##  $ s1     : int  1 1 1 2 0 0 2 3 1 2 ...
##  $ s2     : int  1 1 1 1 1 1 1 4 3 1 ...
##  $ thedate: Date, format: "2016-08-13" "2016-08-13" ...
```

This looks better.

There is also a function `str_trim` that is part of `stringr` (which in turn is part of the `tidyverse`), that you could also use in a `mutate` to create new team names from the old ones. Note, however, that `stringr` *does not* get loaded with the `tidyverse`, so you have to load it separately:

```
library(stringr)
england %>% mutate(team1=str_trim(team1),
                   team2=str_trim(team2)
                  ) %>% str()
```

```
## 'data.frame': 130 obs. of  5 variables:
##  $ date : chr  "2016-08-13   " "2016-08-13   " "2016-08-13   " "2016-08-13   " ...
##  $ team1: chr  "Southampton" "Middlesbrough" "Everton" "Manchester City" ...
##  $ team2: chr  "Watford" "Stoke City" "Tottenham Hotspur" "Sunderland" ...
##  $ s1   : num  1 1 1 2 0 0 2 3 1 2 ...
##  $ s2   : num  1 1 1 1 1 1 1 4 3 1 ...
```

That works. (I overwrote the blank-padded team names with the trimmed ones, since I couldn't

imagine that I would need the blank-padded ones for anything, and I couldn't imagine that `str_trim` was going to give me any grief.)

If you forget to load `stringr`, trying to use `str_trim` will give you an incomprehensible error. This happened to me. I eventually got around to asking myself "well, what does the `tidyverse` *actually* load, anyway?" and that's when I found out that I had to load `stringr` myself.

Marking: 1 point for `stringsAsFactors` (or equivalent with `as.character`) properly done, 2 points for correct `as.Date` or `ymd`. I was willing to forgive *one* small blemish (eg. `as.Dates` or `as.date`). If you made an attempt at getting the dates but what you had wouldn't quite work, there was a point for that. You have to read in the dates with `ymd` (if you went that way), since they are years then months then days in that order, so a permutation like `dmy` will fail. No points for reading in the data file without `stringsAsFactors` (unless you later fixed it up), since at this point it is taken for granted that you can do *that*.

2017: This ought to be easier, with `read_csv`, since we expect the dates to be guessed as such (and we *will* get text rather than factors, actually, since that is one of the things the `read_` functions do):

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

```
## Parsed with column specification:
## cols(
##   date = col_date(format = ""),
##   team1 = col_character(),
##   team2 = col_character(),
##   s1 = col_integer(),
##   s2 = col_integer()
## )
```

```
england
```

```
## # A tibble: 130 x 5
##    date       team1           team2                   s1    s2
##    <date>     <chr>           <chr>                <int> <int>
##  1 2016-08-13 Southampton     Watford                  1     1
##  2 2016-08-13 Middlesbrough   Stoke City               1     1
##  3 2016-08-13 Everton         Tottenham Hotspur        1     1
##  4 2016-08-13 Manchester City Sunderland               2     1
##  5 2016-08-13 Crystal Palace  West Bromwich Albion     0     1
##  6 2016-08-13 Burnley         Swansea City             0     1
##  7 2016-08-13 Hull City       Leicester City           2     1
##  8 2016-08-14 Arsenal         Liverpool                3     4
##  9 2016-08-14 AFC Bournemouth Manchester United        1     3
## 10 2016-08-15 Chelsea         West Ham United          2     1
## # ... with 120 more rows
```

Piece of cake. Date is date, the long team names got trimmed, and the scores are properly integers.

(f) (2 marks) What R `dplyr` code would display all the games that the team Liverpool played (both home games and away games)?

Solution: This is a `filter` to pick out rows, but the condition is a bit more complicated than usual: it has an "or" in it. So it would look something like this:

```
england2 %>%
  filter(team1=="Liverpool" | team2=="Liverpool")
```

```
##          date              team1                team2 s1 s2    thedate
## 1  2016-08-14           Arsenal            Liverpool  3  4 2016-08-14
## 2  2016-08-20           Burnley            Liverpool  2  0 2016-08-20
## 3  2016-08-27 Tottenham Hotspur            Liverpool  1  1 2016-08-27
## 4  2016-09-10         Liverpool       Leicester City  4  1 2016-09-10
## 5  2016-09-16           Chelsea            Liverpool  1  2 2016-09-16
## 6  2016-09-24         Liverpool            Hull City  5  1 2016-09-24
## 7  2016-10-01      Swansea City            Liverpool  1  2 2016-10-01
## 8  2016-10-17         Liverpool    Manchester United  0  0 2016-10-17
## 9  2016-10-22         Liverpool West Bromwich Albion  2  1 2016-10-22
## 10 2016-10-29    Crystal Palace            Liverpool  2  4 2016-10-29
## 11 2016-11-06         Liverpool              Watford  6  1 2016-11-06
## 12 2016-11-19       Southampton            Liverpool  0  0 2016-11-19
## 13 2016-11-26         Liverpool           Sunderland  2  0 2016-11-26
```

(or without the pipe and with the data frame as the first thing in the `filter`).

Strictly speaking, if you use the data frame you read in from the file, you would have to test for `Liverpool` followed by enough blanks to make 27 characters long, since that's what got read in from the file. That's why I wanted to get rid of the blanks on my text. If you have what I had just above, or something equivalent to it, without removing the blanks on the end, though, I'm good.

These do not work:

```
england2 %>% filter(team1=="Liverpool", team2=="Liverpool")
```

```
## [1] date    team1   team2   s1      s2      thedate
## <0 rows> (or 0-length row.names)
```

```
england2 %>% filter(team1=="Liverpool") %>%
  filter(team2=="Liverpool")
```

```
## [1] date    team1   team2   s1      s2      thedate
## <0 rows> (or 0-length row.names)
```

This is because both of these test that team 1 is Liverpool *and* team 2 is Liverpool, which is never true because Liverpool never play against themselves. We want "or" (as above) not "and".

If you forgot how to do "or" in one of these, there are at least two workarounds. One is to first collect the games where Liverpool plays at home, then collect the games where Liverpool plays away, then glue them together:

```
liv1=filter(england2,team1=="Liverpool")
liv2=filter(england2,team2=="Liverpool")
bind_rows(liv1,liv2)
```

```
##          date             team1                team2 s1 s2    thedate
## 1  2016-09-10         Liverpool       Leicester City  4  1 2016-09-10
## 2  2016-09-24         Liverpool            Hull City  5  1 2016-09-24
## 3  2016-10-17         Liverpool    Manchester United  0  0 2016-10-17
## 4  2016-10-22         Liverpool West Bromwich Albion  2  1 2016-10-22
## 5  2016-11-06         Liverpool              Watford  6  1 2016-11-06
## 6  2016-11-26         Liverpool           Sunderland  2  0 2016-11-26
## 7  2016-08-14           Arsenal            Liverpool  3  4 2016-08-14
## 8  2016-08-20           Burnley            Liverpool  2  0 2016-08-20
## 9  2016-08-27 Tottenham Hotspur            Liverpool  1  1 2016-08-27
## 10 2016-09-16           Chelsea            Liverpool  1  2 2016-09-16
## 11 2016-10-01      Swansea City            Liverpool  1  2 2016-10-01
## 12 2016-10-29    Crystal Palace            Liverpool  2  4 2016-10-29
## 13 2016-11-19       Southampton            Liverpool  0  0 2016-11-19
```

You may not have seen `bind_rows` before. I think `rbind` would also work, and I would certainly have accepted it.

That was inspired by a couple of attempts. Somebody else had the idea that it would be much easier if there were only *one* team column, which appears to call for `gather`. Let's see what happens, building a pipe as we go:

```
england2 %>% gather(whichteam,team,team1:team2) %>% head()
```

```
##         date s1 s2    thedate whichteam            team
## 1 2016-08-13  1  1 2016-08-13     team1     Southampton
## 2 2016-08-13  1  1 2016-08-13     team1   Middlesbrough
## 3 2016-08-13  1  1 2016-08-13     team1         Everton
## 4 2016-08-13  2  1 2016-08-13     team1 Manchester City
## 5 2016-08-13  0  1 2016-08-13     team1  Crystal Palace
## 6 2016-08-13  0  1 2016-08-13     team1         Burnley
```

Now we only need to check whether `team=="Liverpool"`:

```
england2 %>% gather(whichteam,team,team1:team2) %>%
  filter(team=="Liverpool")
```

```
##          date s1 s2    thedate whichteam      team
## 1  2016-09-10  4  1 2016-09-10     team1 Liverpool
## 2  2016-09-24  5  1 2016-09-24     team1 Liverpool
## 3  2016-10-17  0  0 2016-10-17     team1 Liverpool
## 4  2016-10-22  2  1 2016-10-22     team1 Liverpool
## 5  2016-11-06  6  1 2016-11-06     team1 Liverpool
## 6  2016-11-26  2  0 2016-11-26     team1 Liverpool
## 7  2016-08-14  3  4 2016-08-14     team2 Liverpool
## 8  2016-08-20  2  0 2016-08-20     team2 Liverpool
## 9  2016-08-27  1  1 2016-08-27     team2 Liverpool
## 10 2016-09-16  1  2 2016-09-16     team2 Liverpool
## 11 2016-10-01  1  2 2016-10-01     team2 Liverpool
## 12 2016-10-29  2  4 2016-10-29     team2 Liverpool
## 13 2016-11-19  0  0 2016-11-19     team2 Liverpool
```

This kind of works, except that it's lost the name of the team that Liverpool was playing *against*, which might be something we want to keep track of. The problem with this approach is that the `gather` produces two rows from each original row, and it's the other row of the pair that contains Liverpool's opposition. So, while I think this is a cute idea (and I wouldn't have thought of it myself), I don't think this is a viable way of answering the question. The way to make it work is probably to create a column that identifies the matches, and then to pull out all the information for the matches identified in code like the above:

```
england2 %>% mutate(thematch=1:n()) %>%
  gather(whichteam,team,team1:team2) %>%
  filter(team=="Liverpool")
```

```
##          date s1 s2    thedate thematch whichteam      team
## 1  2016-09-10  4  1 2016-09-10       33     team1 Liverpool
## 2  2016-09-24  5  1 2016-09-24       53     team1 Liverpool
## 3  2016-10-17  0  0 2016-10-17       80     team1 Liverpool
## 4  2016-10-22  2  1 2016-10-22       82     team1 Liverpool
## 5  2016-11-06  6  1 2016-11-06      107     team1 Liverpool
## 6  2016-11-26  2  0 2016-11-26      122     team1 Liverpool
## 7  2016-08-14  3  4 2016-08-14        8     team2 Liverpool
## 8  2016-08-20  2  0 2016-08-20       17     team2 Liverpool
## 9  2016-08-27  1  1 2016-08-27       24     team2 Liverpool
## 10 2016-09-16  1  2 2016-09-16       41     team2 Liverpool
## 11 2016-10-01  1  2 2016-10-01       66     team2 Liverpool
## 12 2016-10-29  2  4 2016-10-29       95     team2 Liverpool
## 13 2016-11-19  0  0 2016-11-19      112     team2 Liverpool
```

Then we use the `thematch` column to `slice` the original data frame (that is, pick out rows by number):

```
england2 %>% mutate(thematch=1:n()) %>%
  gather(whichteam,team,team1:team2) %>%
  filter(team=="Liverpool") -> livgames
england2 %>% slice(livgames$thematch)
```

```
##          date             team1                team2 s1 s2    thedate
## 1  2016-09-10         Liverpool       Leicester City  4  1 2016-09-10
## 2  2016-09-24         Liverpool            Hull City  5  1 2016-09-24
## 3  2016-10-17         Liverpool    Manchester United  0  0 2016-10-17
## 4  2016-10-22         Liverpool West Bromwich Albion  2  1 2016-10-22
## 5  2016-11-06         Liverpool              Watford  6  1 2016-11-06
## 6  2016-11-26         Liverpool           Sunderland  2  0 2016-11-26
## 7  2016-08-14           Arsenal            Liverpool  3  4 2016-08-14
## 8  2016-08-20           Burnley            Liverpool  2  0 2016-08-20
## 9  2016-08-27 Tottenham Hotspur            Liverpool  1  1 2016-08-27
## 10 2016-09-16           Chelsea            Liverpool  1  2 2016-09-16
## 11 2016-10-01      Swansea City            Liverpool  1  2 2016-10-01
## 12 2016-10-29    Crystal Palace            Liverpool  2  4 2016-10-29
## 13 2016-11-19       Southampton            Liverpool  0  0 2016-11-19
```

It works, but it seems like an awful lot of work for two points. Also, I couldn't fit it all into one pipe. I had to save the first one into data frame `livgames`, and then pass a column of that into `slice`. Not the most aesthetic.

(g) (2 marks) What R `dplyr` code would show, for each different date:

- the total number of goals scored,
- and the number of games played,

on that date?

Solution: This is a `group_by` and `summarize`. For the grouping part, I think you can use either the actual dates that you created, or the original text read in as `date`. For the goals, you need both `s1` and `s2` summed, either together or separately:

```
england2 %>% group_by(thedate) %>%
  summarize(games=n(),goals=sum(s1+s2))
```

```
## # A tibble: 35 x 3
##    thedate    games goals
##    <date>     <int> <int>
##  1 2016-08-13     7    14
##  2 2016-08-14     2    11
##  3 2016-08-15     1     3
##  4 2016-08-19     1     2
##  5 2016-08-20     7    16
##  6 2016-08-21     2     4
##  7 2016-08-27     8    18
##  8 2016-08-28     2     4
##  9 2016-09-10     8    27
## 10 2016-09-11     1     4
## # ... with 25 more rows
```

I think this would be equivalent, with the goals summed separately:

```
england2 %>% group_by(date) %>%
  summarize(games=n(),goals=sum(s1)+sum(s2))
```

```
## # A tibble: 35 x 3
##    date       games goals
##    <chr>      <int> <int>
##  1 2016-08-13     7    14
##  2 2016-08-14     2    11
##  3 2016-08-15     1     3
##  4 2016-08-19     1     2
##  5 2016-08-20     7    16
##  6 2016-08-21     2     4
##  7 2016-08-27     8    18
##  8 2016-08-28     2     4
##  9 2016-09-10     8    27
## 10 2016-09-11     1     4
## # ... with 25 more rows
```

Even though `date` is text and `thedate` is a proper `Date`, either of them are good to be grouped by, and because of the way the text dates are formatted, they are listed in the same order.

You can quickly check these values for reasonableness; for example, there are two games on August 14, with a total of $3 + 4 + 1 + 3 = 11$ goals in the two games.

For marking here, I gave 2 points for anything that looked as if it would work, or was a small blemish away from working. Anything that got at least as far as `group_by(date)` was worth 1, with the unfortunate side-effect that 1 covered a rather wide range, from no `summarize` at all to one part of the `summarize` being correct. In one sense, it would have been nice to have 3 points available for this, but on the other hand I didn't want to over-emphasize its importance in the exam. So we have to live with what there is. This is one of those questions intended to identify the A and A+ students. I recognize that a lot of people won't get this one.

What would perhaps be more interesting is the number of games in a week (which ought to be 10, since each team plays once a week on average and there are 20 teams in the league). `lubridate` has a function called `week`, which returns the week number of a date. We may have to come back and look at its definition, but we'll try it first and see:

```
england2 %>% mutate(wk=week(thedate)) %>% dplyr::select(-date) -> england5
head(england5,20)
```

```
##                   team1                team2 s1 s2    thedate wk
## 1           Southampton              Watford  1  1 2016-08-13 33
## 2         Middlesbrough           Stoke City  1  1 2016-08-13 33
## 3               Everton    Tottenham Hotspur  1  1 2016-08-13 33
## 4       Manchester City           Sunderland  2  1 2016-08-13 33
## 5        Crystal Palace West Bromwich Albion  0  1 2016-08-13 33
## 6               Burnley         Swansea City  0  1 2016-08-13 33
## 7             Hull City       Leicester City  2  1 2016-08-13 33
## 8               Arsenal            Liverpool  3  4 2016-08-14 33
## 9       AFC Bournemouth    Manchester United  1  3 2016-08-14 33
## 10              Chelsea      West Ham United  2  1 2016-08-15 33
## 11    Manchester United          Southampton  2  0 2016-08-19 34
## 12    Tottenham Hotspur       Crystal Palace  1  0 2016-08-20 34
## 13 West Bromwich Albion              Everton  1  2 2016-08-20 34
## 14       Leicester City              Arsenal  0  0 2016-08-20 34
## 15           Stoke City      Manchester City  1  4 2016-08-20 34
## 16              Watford              Chelsea  1  2 2016-08-20 34
## 17              Burnley            Liverpool  2  0 2016-08-20 34
## 18         Swansea City            Hull City  0  2 2016-08-20 34
## 19           Sunderland        Middlesbrough  1  2 2016-08-21 34
## 20      West Ham United      AFC Bournemouth  1  0 2016-08-21 34
```

```
england5 %>% group_by(wk) %>% summarize(games=n())
```

```
## # A tibble: 13 x 2
##       wk games
##    <dbl> <int>
##  1    33    10
##  2    34    10
##  3    35    10
##  4    37    10
##  5    38    10
##  6    39    10
##  7    40    10
##  8    42    10
##  9    43    10
## 10    44    10
## 11    45    10
## 12    47    10
## 13    48    10
```

That worked better than I expected. I thought it would get hung up on games on Mondays, since Monday is often first day of the week, but it's the *last* day of the soccer week (games on Monday are usually the last ones of the previous weekend).

What about that `dplyr::select` thing? I think I loaded a package that *also* has a `select`, so that calling for simply `select` is now getting the wrong one. To make sure I get the `dplyr select`, I can call for it by name, as shown: "the `select` that lives in `dplyr`, accept no substitutes".

Are there any games on Mondays? We can find that out too:

```
england5 %>% mutate(dow=wday(thedate,label=T)) %>%
  filter(dow=="Mon")
```

```
##                   team1              team2 s1 s2    thedate wk dow
## 1               Chelsea   West Ham United  2  1 2016-08-15 33 Mon
## 2            Sunderland           Everton  0  3 2016-09-12 37 Mon
## 3               Burnley           Watford  2  0 2016-09-26 39 Mon
## 4             Liverpool Manchester United  0  0 2016-10-17 42 Mon
## 5            Stoke City      Swansea City  3  1 2016-10-31 44 Mon
## 6 West Bromwich Albion           Burnley  4  0 2016-11-21 47 Mon
```

There are. Do they count as the end of the week or the beginning? To find out, display the Sunday games (of which there are a lot) as well as the Monday games, and compare weeks:

```
england5 %>% mutate(dow=wday(thedate,label=T)) %>%
  filter(dow=="Mon" | dow=="Sun")
```

```
##                   team1                team2 s1 s2    thedate wk dow
## 1               Arsenal            Liverpool  3  4 2016-08-14 33 Sun
## 2       AFC Bournemouth    Manchester United  1  3 2016-08-14 33 Sun
## 3               Chelsea      West Ham United  2  1 2016-08-15 33 Mon
## 4            Sunderland        Middlesbrough  1  2 2016-08-21 34 Sun
## 5       West Ham United      AFC Bournemouth  1  0 2016-08-21 34 Sun
## 6       Manchester City      West Ham United  3  1 2016-08-28 35 Sun
## 7  West Bromwich Albion        Middlesbrough  0  0 2016-08-28 35 Sun
## 8          Swansea City              Chelsea  2  2 2016-09-11 37 Sun
## 9            Sunderland              Everton  0  3 2016-09-12 37 Mon
## 10          Southampton         Swansea City  1  0 2016-09-18 38 Sun
## 11    Tottenham Hotspur           Sunderland  1  0 2016-09-18 38 Sun
## 12       Crystal Palace           Stoke City  4  1 2016-09-18 38 Sun
## 13              Watford    Manchester United  3  1 2016-09-18 38 Sun
## 14      West Ham United          Southampton  0  3 2016-09-25 39 Sun
## 15              Burnley              Watford  2  0 2016-09-26 39 Mon
## 16    Manchester United           Stoke City  1  1 2016-10-02 40 Sun
## 17    Tottenham Hotspur      Manchester City  2  0 2016-10-02 40 Sun
## 18       Leicester City          Southampton  0  0 2016-10-02 40 Sun
## 19              Burnley              Arsenal  0  1 2016-10-02 40 Sun
## 20          Southampton              Burnley  3  1 2016-10-16 42 Sun
## 21        Middlesbrough              Watford  0  1 2016-10-16 42 Sun
## 22            Liverpool    Manchester United  0  0 2016-10-17 42 Mon
## 23              Chelsea    Manchester United  4  0 2016-10-23 43 Sun
## 24      Manchester City          Southampton  1  1 2016-10-23 43 Sun
## 25          Southampton              Chelsea  0  2 2016-10-30 44 Sun
## 26              Everton      West Ham United  2  0 2016-10-30 44 Sun
## 27           Stoke City         Swansea City  3  1 2016-10-31 44 Mon
## 28              Arsenal    Tottenham Hotspur  1  1 2016-11-06 45 Sun
## 29            Liverpool              Watford  6  1 2016-11-06 45 Sun
## 30       Leicester City West Bromwich Albion  1  2 2016-11-06 45 Sun
## 31            Hull City          Southampton  2  1 2016-11-06 45 Sun
## 32         Swansea City    Manchester United  1  3 2016-11-06 45 Sun
## 33        Middlesbrough              Chelsea  0  1 2016-11-20 47 Sun
## 34 West Bromwich Albion              Burnley  4  0 2016-11-21 47 Mon
## 35              Arsenal      AFC Bournemouth  3  1 2016-11-27 48 Sun
## 36    Manchester United      West Ham United  1  1 2016-11-27 48 Sun
## 37          Southampton              Everton  1  0 2016-11-27 48 Sun
## 38              Watford           Stoke City  0  1 2016-11-27 48 Sun
```

Soccer-wise, the Monday ought to be considered as the *end* of the week, since Friday through Monday is one soccer weekend. Is that what happened? It looks as if it is (look for the Mondays, and see that the Sunday before each is the *same* week number). So that worked.

Which makes me wonder, when does the `lubridate` week start? Let's find out:

```
dates=as.Date("2016-11-28")+0:7
data.frame(thedate=dates) %>%
  mutate(dow=wday(thedate,label=T),wno=week(thedate))
```

```
##      thedate dow wno
## 1 2016-11-28 Mon  48
## 2 2016-11-29 Tue  48
## 3 2016-11-30 Wed  48
## 4 2016-12-01 Thu  48
## 5 2016-12-02 Fri  49
## 6 2016-12-03 Sat  49
## 7 2016-12-04 Sun  49
## 8 2016-12-05 Mon  49
```

The weeks this year seem to be starting on Friday (the start of the soccer weekend). Maybe that's because January 1st this year was a Friday?

```
dates=as.Date("2015-12-30")+0:7
data.frame(thedate=dates) %>%
  mutate(dow=wday(thedate,label=T),wno=week(thedate))
```

```
##      thedate dow wno
## 1 2015-12-30 Wed  52
## 2 2015-12-31 Thu  53
## 3 2016-01-01 Fri   1
## 4 2016-01-02 Sat   1
## 5 2016-01-03 Sun   1
## 6 2016-01-04 Mon   1
## 7 2016-01-05 Tue   1
## 8 2016-01-06 Wed   1
```

It seems that this is the story. That's also what the help for `week` says (although not in the absolutely clearest way): week 1 of any year starts on January 1 of that year.

`lubridate` also has a function `isoweek` that obtains week numbers according to the ISO standard: week 1 of a year is the one containing the first Thursday, and weeks always start on a Monday. (Days of a year before week 1 belong to week 53 of the previous year.) Businesses have used week numbers according to this standard for a long time, for payroll and other things that recur weekly.

```
dates=as.Date("2015-12-30")+0:7
data.frame(thedate=dates) %>%
  mutate(dow=wday(thedate,label=T),wno=isoweek(thedate))
```

```
##      thedate dow wno
## 1 2015-12-30 Wed  53
## 2 2015-12-31 Thu  53
## 3 2016-01-01 Fri  53
## 4 2016-01-02 Sat  53
## 5 2016-01-03 Sun  53
## 6 2016-01-04 Mon   1
## 7 2016-01-05 Tue   1
## 8 2016-01-06 Wed   1
```

That seems to check out. If I had used ISO week numbers, I would have had to move the dates

back by a day (by subtracting 1) to get Monday to be the end of the previous week rather than the beginning of the next one.

10. A study was conducted to evaluate the performance of a diesel engine run on three different types of fuel. The response variable was called the Mass Burning Rate, and it was thought to depend on both the Fuel type and on the Brake Power. Three observations were (intended to be) taken at each brake power; each observation is labelled with a unique one-letter ID. You may assume that the data have been read into a SAS data set called `synfuels`, as shown in Figure 34.

    (a) (4 marks) A plot is shown in Figure 35. Your task is to give the SAS code that produced this plot, bearing in mind all the features of the plot.

Solution: The actual code I used was as below

```
proc sgplot;
  scatter y=massburnrate x=brakepower / datalabel=id group=fuel;
  yaxis label="Mass Burn Rate";
  xaxis label="Brake Power";
```

The key features to make sure your code includes are:

- scatterplot with the right variables on the right axes
- the points labelled by their ID
- the different colours and plotting symbols for each `fuel`
- the labels on the x and y axes (the clue being that these are not the names of the variables, so they must have been specified separately).

On a scatterplot, `markers` doesn't do anything, so I've ignored it in the grading (you needed to get both the `datalabel` and the `group`, as well as the axis labels, for full marks).

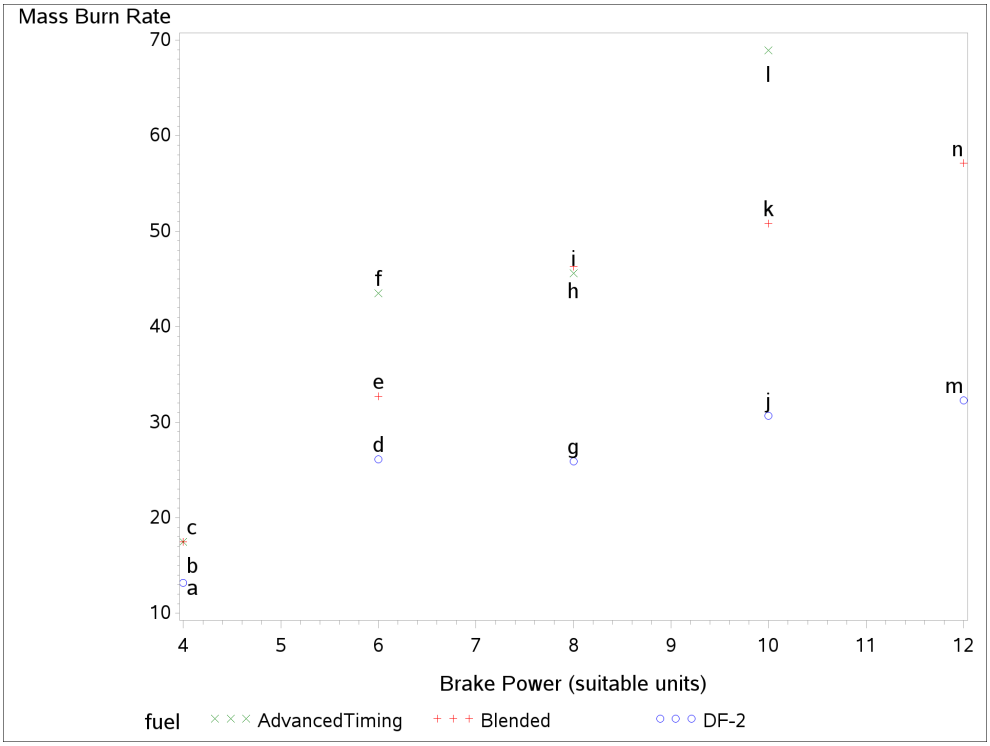
It may also be possible to use `proc gplot` to get the same graph. You might find that in the textbook and in old notes/exams, but using it also rather betrays that you didn't come to class. Nonetheless, if you can make it work, it's good. (It will be a lot more fiddly than `proc sgplot`.)

To demonstrate, I have to read in the data again:

```
data synfuels;
  infile '/home/ken/synfuels.txt' firstobs=2;
  input id $ brakepower fuel: $15. massburnrate;
```

To work:

```
symbol1 value=x cv=green pointlabel=(height=10pt '#id');
symbol2 value=plus cv=red pointlabel=(height=10pt '#id');
symbol3 value=circle cv=blue pointlabel=(height=10pt '#id');
axis1 label=('Brake Power (suitable units)');
axis2 label=('Mass Burn Rate');
proc gplot;
  plot massburnrate*brakepower=fuel / haxis=axis1 vaxis=axis2;
```



That's as close as I can get, but is close enough for full marks if you can do it. (There are other ways also, marked according to what they successfully did.)

Marking: 1 point for making the scatterplot with no annotation or axis labels; 2 for adding one of `datalabel` or `group` (or the axis labels) correctly; 3 for adding both of the above annotations correctly; 4 for adding the axis labels as well. (Most people forgot the axis labels, but getting those right and missing one of `datalabel` or `group` would be 3 as well.) This is the `proc sgplot` scale; I was as consistent as possible with this if you used `proc gplot`. In one or two `gplot` cases, I tested whether the given code would work by actually running it, and marked according to what was successfully produced.

(b) (3 marks) How would you label the points with the `id` *only* for the `AdvancedTiming` fuel, and not for the other fuels? Give code to accomplish this. In the part of your code that draws the revised plot, you only need to describe any *changes* to your code from (a) (that is, you don't need to write out the whole thing again).

Solution:

I need to create a new piece of text that is the id for the observations that are from the `AdvancedTiming` fuel and blank otherwise, which is done by making a new SAS data set, thus:

```
data synfuels2;
  set synfuels;
  if fuel="AdvancedTiming" then do
    newid=id;
  end;
```

Did that work?

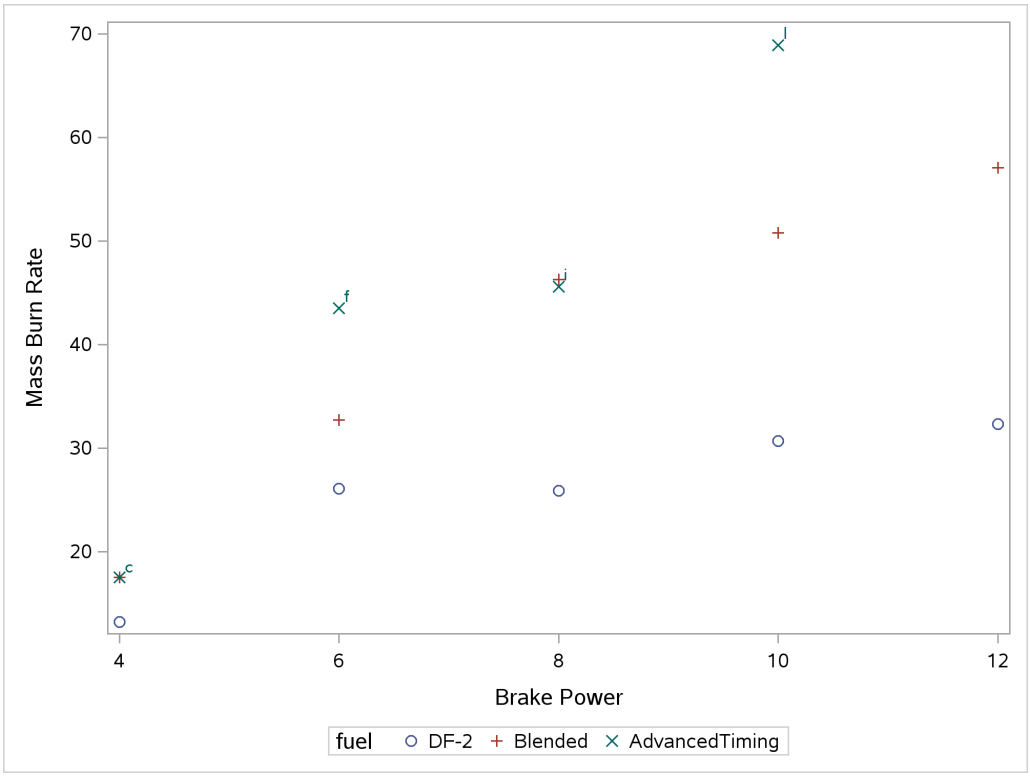
```
proc print;
```

Obs	id	brakepower	fuel	massburnrate	newid
1	a	4	DF-2	13.2	
2	b	4	Blended	17.5	
3	c	4	AdvancedTiming	17.5	c
4	d	6	DF-2	26.1	
5	e	6	Blended	32.7	
6	f	6	AdvancedTiming	43.5	f
7	g	8	DF-2	25.9	
8	h	8	Blended	46.3	
9	i	8	AdvancedTiming	45.6	i
10	j	10	DF-2	30.7	
11	k	10	Blended	50.8	
12	l	10	AdvancedTiming	68.9	l
13	m	12	DF-2	32.3	
14	n	12	Blended	57.1	

You see that the only `id` values that appear in the `newid` column are the ones that went with the `AdvancedTiming` fuel. So now we redo the plot, but changing `datalabel=id` to `datalabel=newid` (or whatever you called your new ID variable).

```
proc sgplot;
  scatter y=massburnrate x=brakepower / datalabel=newid group=fuel;
  yaxis label="Mass Burn Rate";
  xaxis label="Brake Power";
```

with result


70
60
50
40
30
20
Mass Burn Rate
4
6
8
10
12
Brake Power
fuel
DF-2
Blended
AdvancedTiming

Success! Only the green crosses got labelled with the IDs (in green).

What I was asking for was (i) the code to create the new data set, and (ii) the *changes* from the code of (a) required to plot it (that is, the last sentence above the revised plot, not the whole code for the revised plot).

Bear in mind the consistency thing again: act as if your (a) was correct, even if you couldn't do it. If you're stuck, *describe* what you would do, and that's worth something.

If you want to do this with `proc gplot`, you have two options: create the new data set with labels `newid` as above, and redefine `symbol1` through `symbol3` above to put `newid` in the `pointlabel`, or, second, take out the `pointlabel`s for the sets of points you do not want labelled: that is, only keeping the `pointlabel` on `symbol1`.

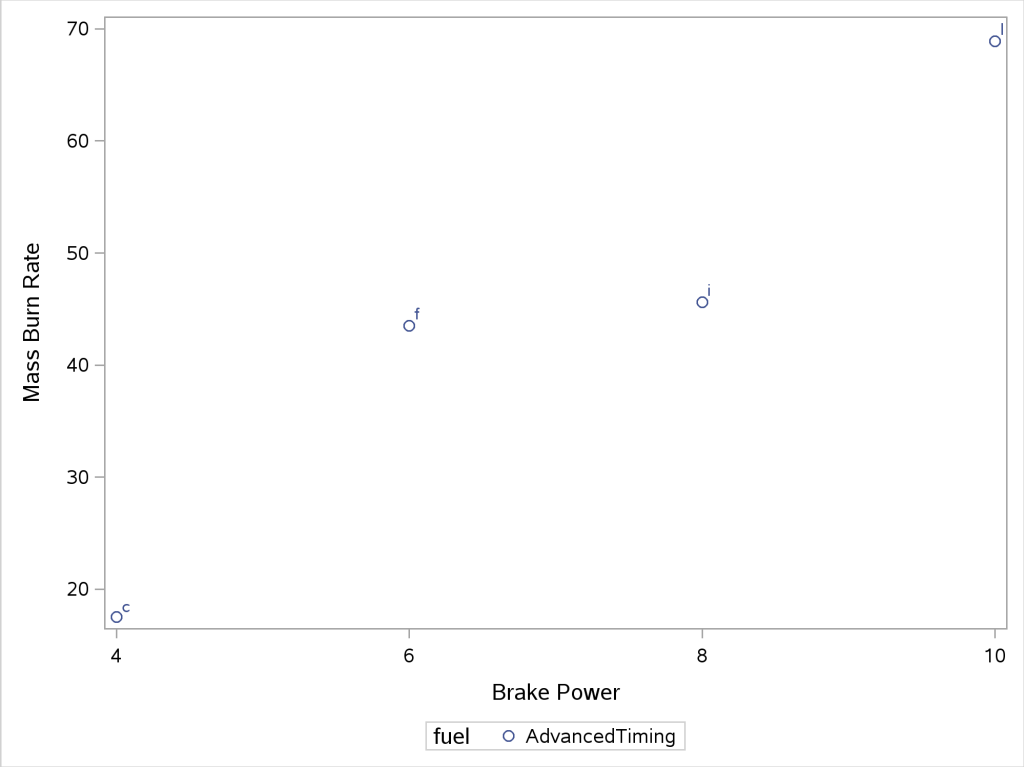
Marking: 2 points for making the new dataset, plus another 1 for saying how the `scatter` line would change. If you did something like this:

```
data synfuels3;
  set synfuels;
  if fuel="AdvancedTiming";
```

and then make your plot again:

```
proc sgplot;
  scatter y=massburnrate x=brakepower / datalabel=id group=fuel;
  yaxis label="Mass Burn Rate";
  xaxis label="Brake Power";
```

you'll find that you only got this:



That is, *only* the Advanced Timing points got plotted at all, whereas what you wanted was *all* the points plotted, but only the Advanced Timing ones *labelled*. That's worth 1 point altogether out of three.