Problems and Solutions in SAS

Ken Butler

1 Introduction

To prepare for SAS questions that you will be handing in, it's a good idea to go through the setup below first.

Remember that you need to hand in your code, your output and your answers to the questions each time. The procedure for SAS involves a bit of setup first:

- Go to SAS Studio. Look top right. You'll see "SAS Programmer", a funny symbol with three lines and some dots, a question mark, and "Sign out".
- The one that is three lines and some dots has a tooltip "More Application Options". Click the three lines and dots.
- Select Preferences from the pop-up menu.
- Click Results (on the left).
- Look for RTF on the right. Click the box next to "Produce RTF Output" and make sure it has a check mark in it.
- Click Save.
- Open some SAS code and run it. The Word link should no longer be greyed out. Click on the Word button (the third one, with a tooltip Download Results as an RTF file). It will download a file containing the results and graphs, which you can open in Word, and copy-paste into your assignment document. This should continue to work in the future (that is, you won't need to do all the preceding steps again).

So now, for an assignment, make a Word document, and for each part of each question that requires coding, you need to:

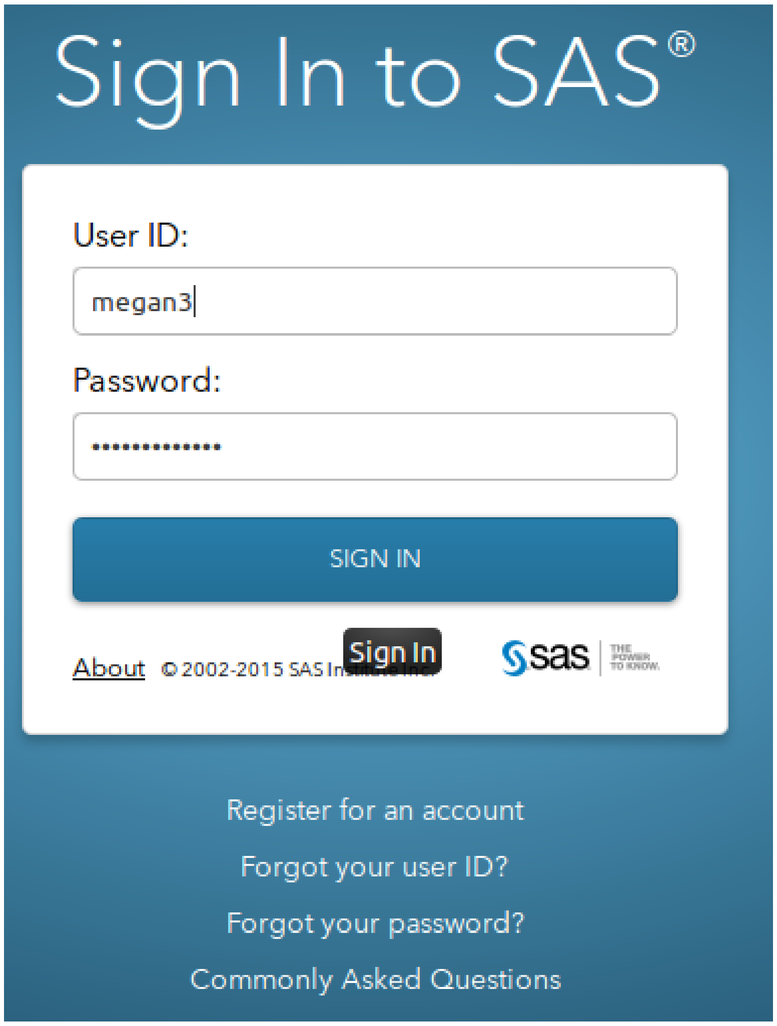
- copy-paste the code from the Code tab
- open the RTF file you downloaded from SAS Studio and copy its contents into your document
- below that, write your answers to the question.

This is not very elegant (at least, not as elegant as the R procedure is), but it's the best we have for now.

2 Introduction

2.1. This question will introduce you to SAS.

Open up a web browser (sometimes Firefox works better than Chrome for this) and go to `https://odamid.oda.sas.com`. Bookmark this page. You'll see this (only without a username and password filled in):


Sign In to SAS®

User ID:

megan3

Password:

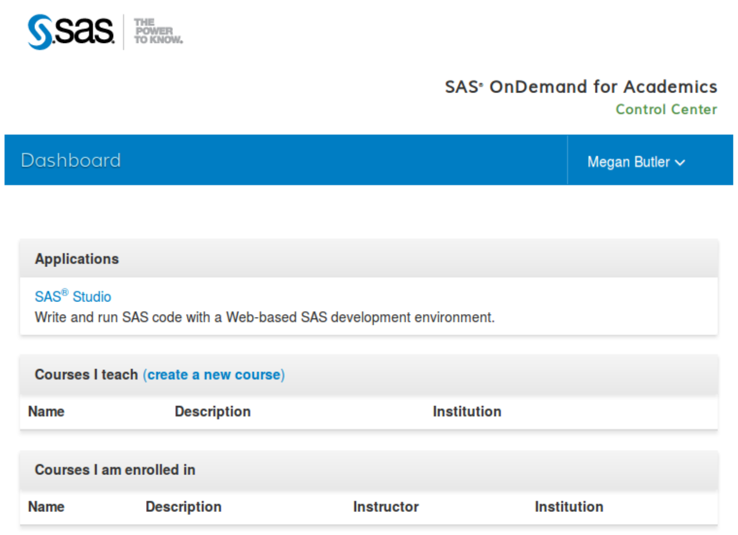
•••••••••••••

(a) Click on Register for an Account (unless you happen to have one already, in which case sign in with username and password). Fill in your first name, last name, e-mail address (twice) and select Country. Click Submit.

(b) Check your e-mail (the address you used above). There should be an e-mail from SAS with a link in it. Click that link. This will take you to a page where you choose a password. Enter your e-mail address again, and enter a password (twice). Note the password rules at the bottom. You need characters from at least *three* of the four categories, so that (for example) if your password contains uppercase and lowercase letters and numbers, you don't need any punctuation characters. Click Create Account.

(c) Next, you get a window with your new user ID in it. Note it down, or save it somehow.[1] There's a link to the Sign In screen.[2] Click it, and sign in with your user ID and the password you chose.

Solution: If you did that properly, you'll be greeted with something like this:



SAS® OnDemand for Academics
Control Center

Dashboard | Megan Butler

Applications

SAS® Studio
Write and run SAS code with a Web-based SAS development environment.

Courses I teach (create a new course)

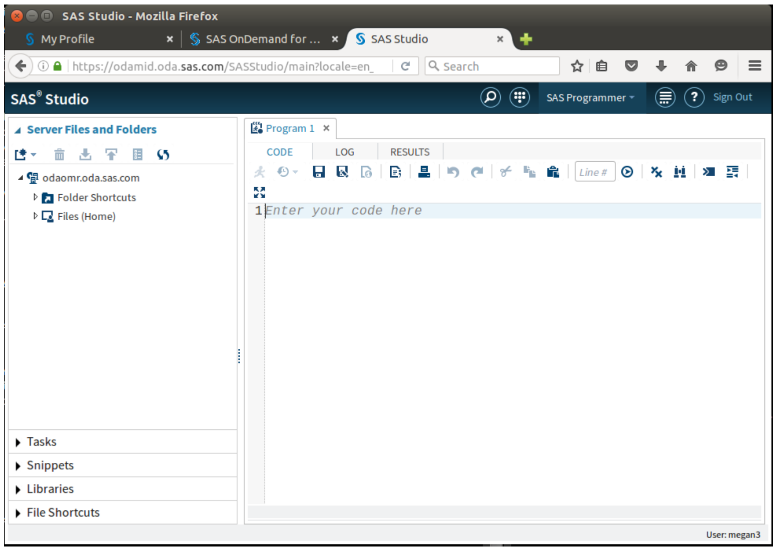
Name	Description	Institution

Courses I am enrolled in

Name	Description	Instructor	Institution

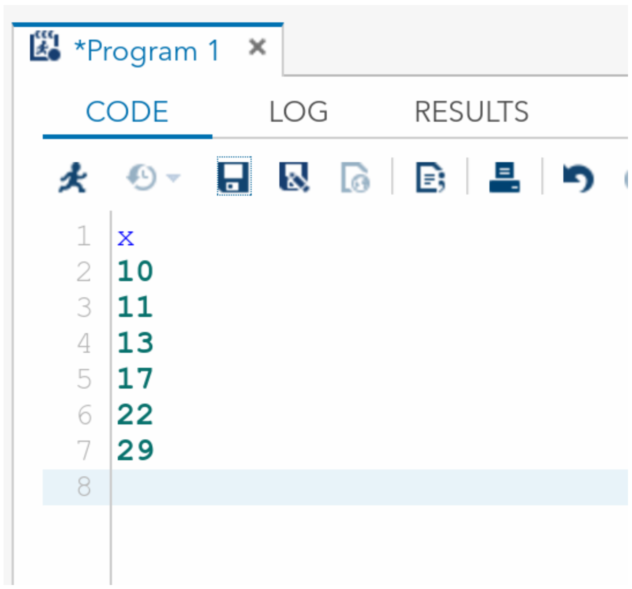
(d) Click on SAS Studio. You will eventually see something like what appears in the Solution below. This is usually the slowest part of the whole operation. If it seems to be taking a long time,[3] leave the rest of this question for now and come back to it later.

Solution: This is the kind of thing you will (eventually) see:



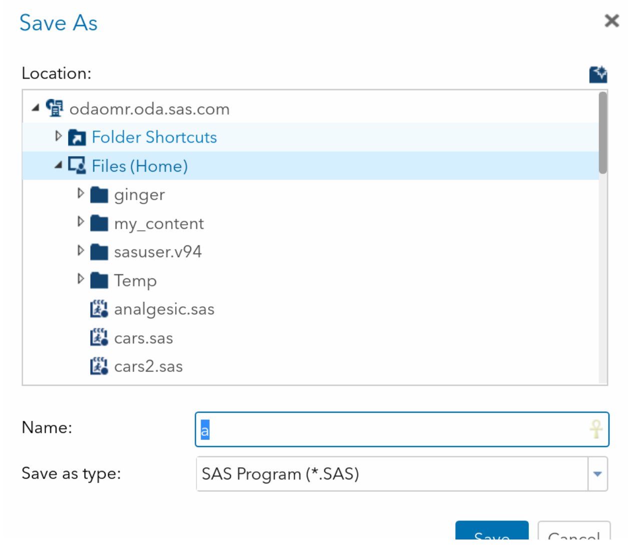
(e) Go over to where it says "Enter your code here". We are going to do two things: first, we'll enter some data and save it, and then (in the next part), we'll write some code to read in the data and run that code.

First, the data. The first row is the name of the variable, `x`, and below that come the values. We only have one variable this time:



```
x
10
11
13
17
22
29
```

Now to save this. Click on the disk icon (its tooltip says "save program"). The Save button at the bottom is pale blue, meaning that you can't save anything yet. Click on the line Files (Home), which should turn the Save button dark blue. Then go down to the Name box, erase what is there, and put just `a` in the box. Leave the Save as Type alone:

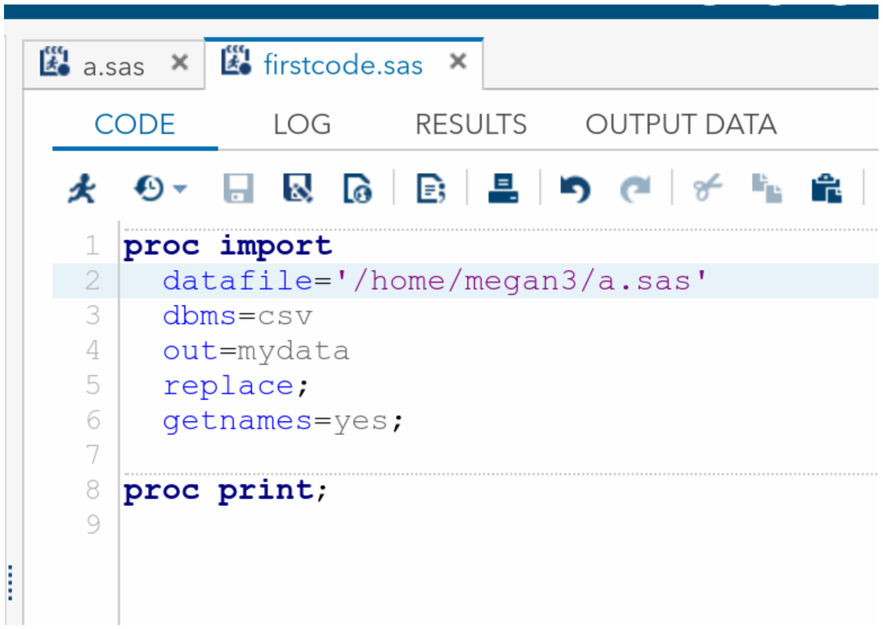


Name: a

Save as type: SAS Program (*.SAS)

I only appear to be able to see half my Save button, but that's OK: I can click on what I can see. When you have what you see above, click Save. You should see the file `a.sas` appear over on the left under Files (Home). Files there are yours and are saved until you delete them.

(f) Next, to read in that data file. Find the New button. This is the leftmost one of the six buttons under Server Files and Folders. Click it. From the dropdown, select SAS Program. (Don't select Import Data, though you might be tempted to do so. That is a "wizard", but we are going to write code to read in the data file, to get practice for later.) You'll now have two tabs: one called `a.sas` with the saved data, and one still called Program 1 since we haven't saved it yet. Type the code shown below into the new tab, and save it as `firstcode.sas`, the same way you saved the data file. When you have done that, you should see what is below:



```
proc import
  datafile='/home/megan3/a.sas'
  dbms=csv
  out=mydata
  replace;
  getnames=yes;

proc print;
```

Check the code very carefully. Make sure that the lines that end with semicolons in my code above also end in semicolons in yours. Finally, for this part, go back to the line that starts `datafile=`, and *change* `megan3` *to your username, the one you used to log into SAS Studio with.* Save your code again. (Just clicking on the Save button will do it, since SAS Studio knows where to save it.)

Solution: What does that code do? Two things: it reads the data in from a file (the `proc import` and the five lines below that), and then it displays it on the screen (the `proc print`). I like to use the indentation shown, though unlike Python it doesn't actually matter, because I want to be able to see that the lines down to `getnames` belong to `proc import` and the `proc print` is a separate thing. SAS Studio helps by using a bold font for the `proc` lines and a regular font for everything else.

The lines under the `proc import` say this:

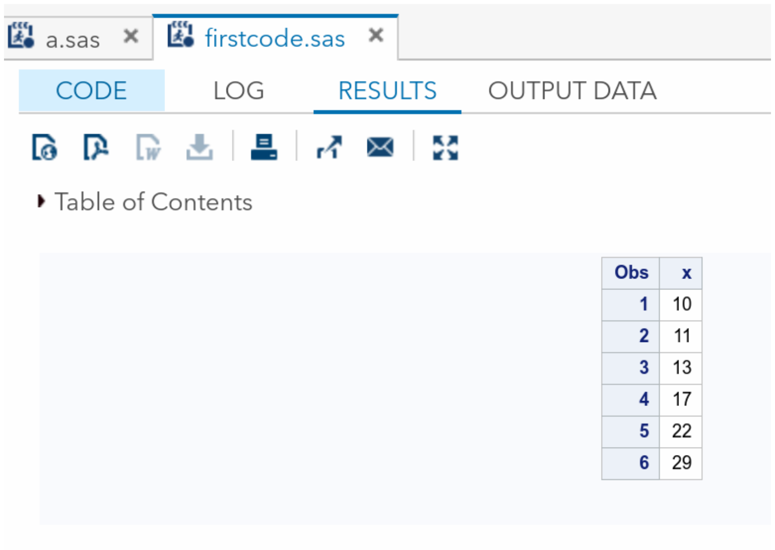
1. where the data file is stored. Mine means "the file called `a.sas` under the account with username `megan3` on SAS's servers".
2. the kind of file it is. We are pretending that this is a `.csv` file, even though it isn't really.
3. the name of the SAS data set to create (which doesn't matter here since we never refer to it by name)
4. Replace any previous SAS data set called `mydata` that we might have created.
5. Get the variable name(s) from the first line of the data file, which is why we put the name `x` there before.

`proc print` displays the most recently-created data set, showing you all the variables in it.

This is the usual structure of SAS code: `proc import` to read some data in from a file, and one or more other `procs` to do something with it.

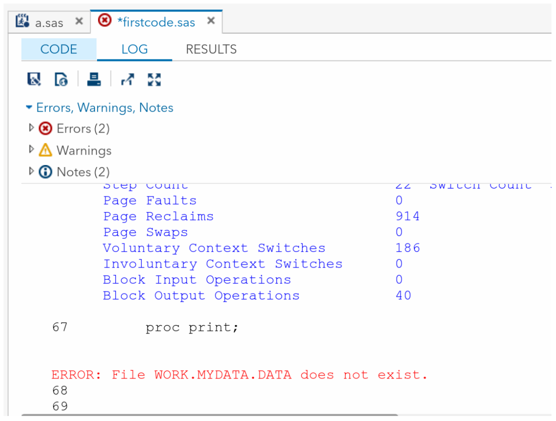
(g) Now try your code and see whether it works. Look for the "running humanoid" under the Code tab, and click it.

Solution: One of two things will happen: either it will work, and you'll see your data set displayed in the Results tab:



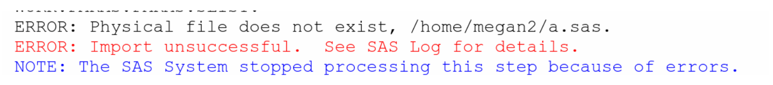
Obs	x
1	10
2	11
3	13
4	17
5	22
6	29

or there will be an Error, and you'll get taken to the Log tab. Here I mistakenly typed the username `megan2` instead of `megan3`:



```
        Step Count                        22  Switch Count
        Page Faults                       0
        Page Reclaims                     914
        Page Swaps                        0
        Voluntary Context Switches        186
        Involuntary Context Switches      0
        Block Input Operations            0
        Block Output Operations           40

 67          proc print;

 ERROR: File WORK.MYDATA.DATA does not exist.
 68
 69
```

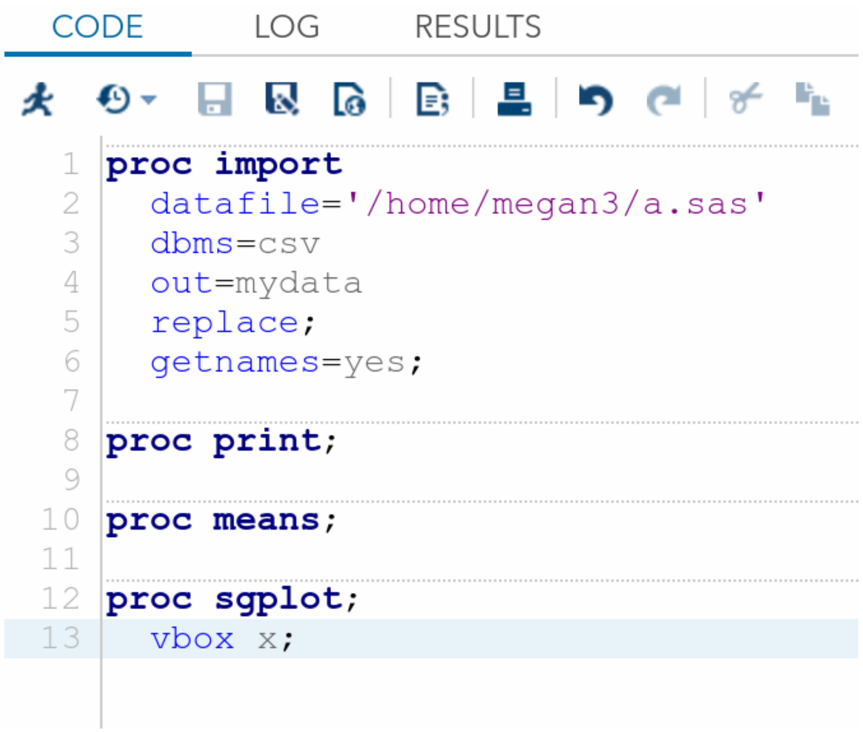
To find out what the error was, scroll up in the Log tab until you find the first red Error, which is here:



```
ERROR: Physical file does not exist, /home/megan2/a.sas.
ERROR: Import unsuccessful.  See SAS Log for details.
NOTE: The SAS System stopped processing this step because of errors.
```

The first line (in black) is telling you what the error is: the file I am asking for doesn't exist, either because I have the filename wrong, or because I have the username wrong. There are many other possible errors, but, whatever error you have, the strategy is to find the *first* place where there was a problem, since that might have caused other errors. In this case, the data set couldn't be created because SAS couldn't find the data file to create it from. Fix that, and the second error will fix itself.

(h) Add some lines to your code to make it look like this:



```
proc import
  datafile='/home/megan3/a.sas'
  dbms=csv
  out=mydata
  replace;
  getnames=yes;

proc print;

proc means;

proc sgplot;
  vbox x;
```

Run it. What do you get? (You'll probably need to scroll down in the Results tab to see it all.)

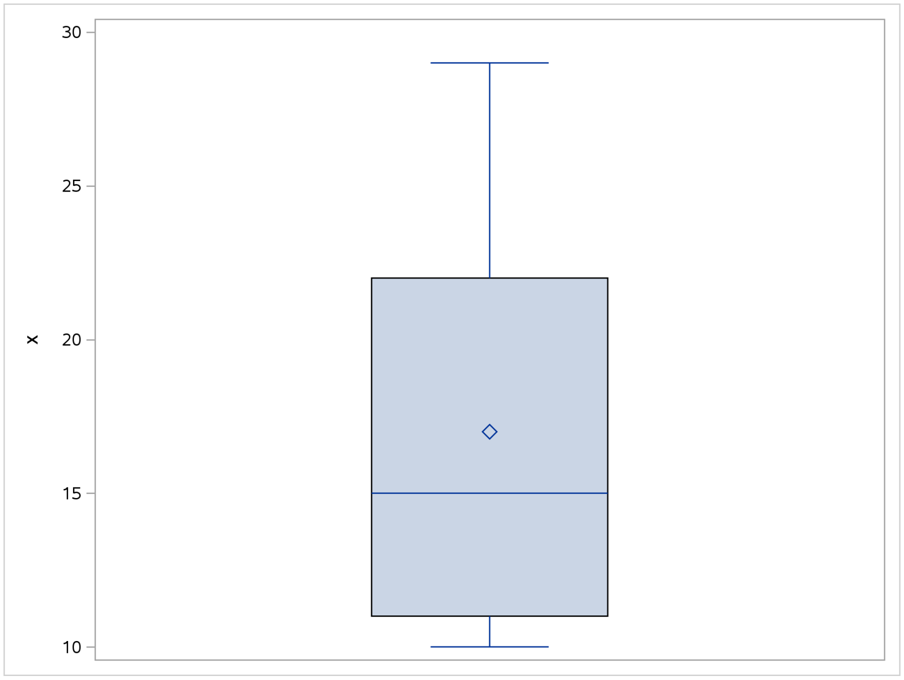
Solution: Here is my code again:

```
proc import
  datafile='/home/ken/a.sas'
  dbms=csv
  out=mydata
  replace;
  getnames=yes;

proc print;

proc means;

proc sgplot;
  vbox x;
```

and here is the output it produces in the Results tab:

```
Obs          x

  1         10
  2         11
  3         13
  4         17
  5         22
  6         29
```

```
                       The MEANS Procedure

                      Analysis Variable : x

        N            Mean         Std Dev         Minimum         Maximum
        -----------------------------------------------------------------
        6      17.0000000       7.3484692      10.0000000      29.0000000
        -----------------------------------------------------------------
```



This is:

- a listing of the data (as before)
- a summary of the variable `x`: the mean, SD, min and max, produced by `proc means` (which makes a summary of all the variables, but here we only have one).
- a boxplot of `x`. `proc sgplot` produces all kinds of different plots; `vbox` is a regular (vertical) boxplot; `hbox` produces a sideways one.

The diamond in the middle of the boxplot is the mean; it is a bit bigger than the median. Also, the long upper whisker adds to the impression of the data being skewed to the right. Which was how I expected it to be, looking at the data values.

In the Results tab, there are buttons that allow you to download the results (for handing in). The best way is Word format. If that is greyed out for you, make sure you have followed the instructions above this question.[4] The "Word" format actually used is called RTF,[5] but it will open in Word and copy-paste to another Word document.

This Word document, click on it shows the kind of thing that you would hand in, as an answer to this question. (You will probably find that the document downloads rather than displaying. Find it in your Downloads folder if it doesn't otherwise display.)

2.2. The quality of orange juice produced by a manufacturer (identity unknown) is constantly being monitored. The manufacturer has developed a "sweetness index" for its orange juice, for which a higher value means sweeter juice. Is the sweetness index related to a chemical measure such as the amount of water-soluble pectin (parts per million) in the orange juice? Data were obtained from 24 production runs, and the sweetness and pectin content were measured for each run. The data are in `http://www.utsc.utoronto.ca/~butler/c32/ojuice.txt`.

(We saw this data set before in R.)

(a) Now we're going to read the data and do the same "analysis" that we did in R before, but now using SAS. Go to SAS Studio, and read in and display your data file.

Solution: Since the file is on the web, get it from the URL using a `filename` line:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/ojuice.txt";

proc import
  datafile=myurl
  dbms=dlm
  out=mydata
  replace;
  delimiter=' ';
  getnames=yes;
```

I had to make a couple of changes from the first one (that I copied): this is a space-delimited file rather than a (supposed) `.csv` file, so `dbms` has to be `dlm`, and then, in the same way as for `read_delim`, I had to say what the delimiter separating the data values was.

Now, to display it, `proc print` is much the easiest way. Add this after your `proc import`:

```
proc print;
```

with these results:

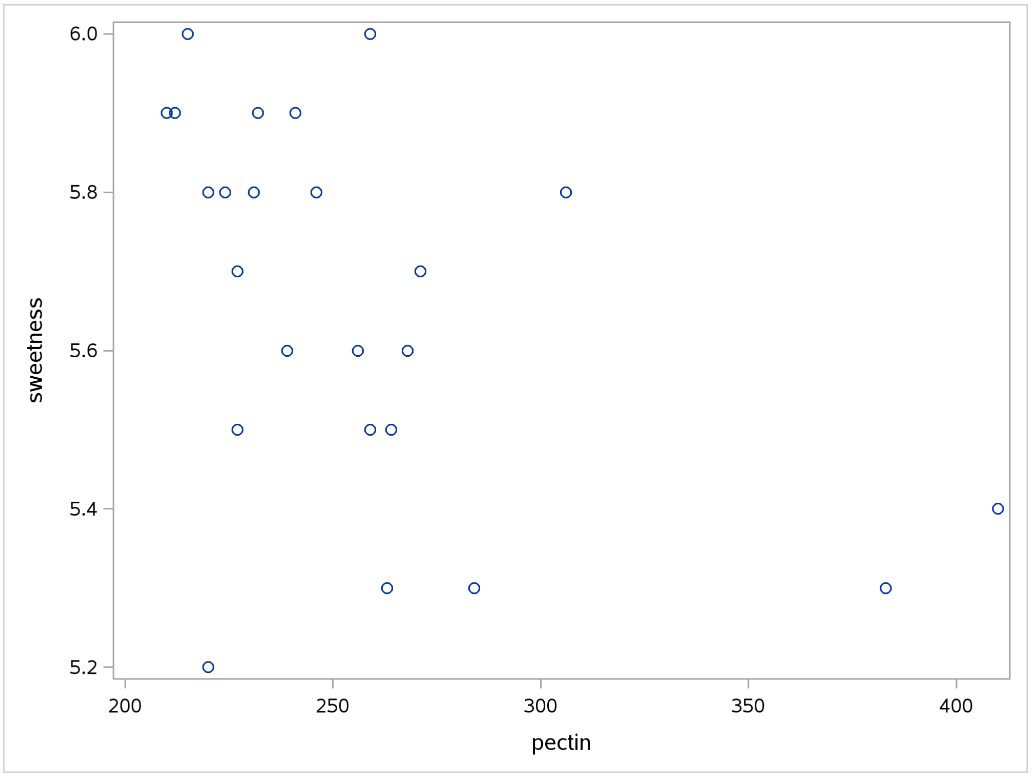
```
      Obs          run      sweetness        pectin

        1            1            5.2           220
        2            2            5.5           227
        3            3              6           259
        4            4            5.9           210
        5            5            5.8           224
        6            6              6           215
        7            7            5.8           231
        8            8            5.6           268
        9            9            5.6           239
       10           10            5.9           212
       11           11            5.4           410
       12           12            5.6           256
       13           13            5.8           306
       14           14            5.5           259
       15           15            5.3           284
       16           16            5.3           383
       17           17            5.7           271
       18           18            5.5           264
       19           19            5.7           227
       20           20            5.3           263
       21           21            5.9           232
       22           22            5.8           220
       23           23            5.8           246
       24           24            5.9           241
```

This displays all 24 lines.

(b) Now create a SAS scatterplot of sweetness against pectin. Go down to the bottom of the code where you read in the data, and add the appropriate code. You can delete the `proc print` if you like, since that has served its purpose.

Solution: Here's the code I added:

```
proc sgplot;
  scatter x=pectin y=sweetness;
```

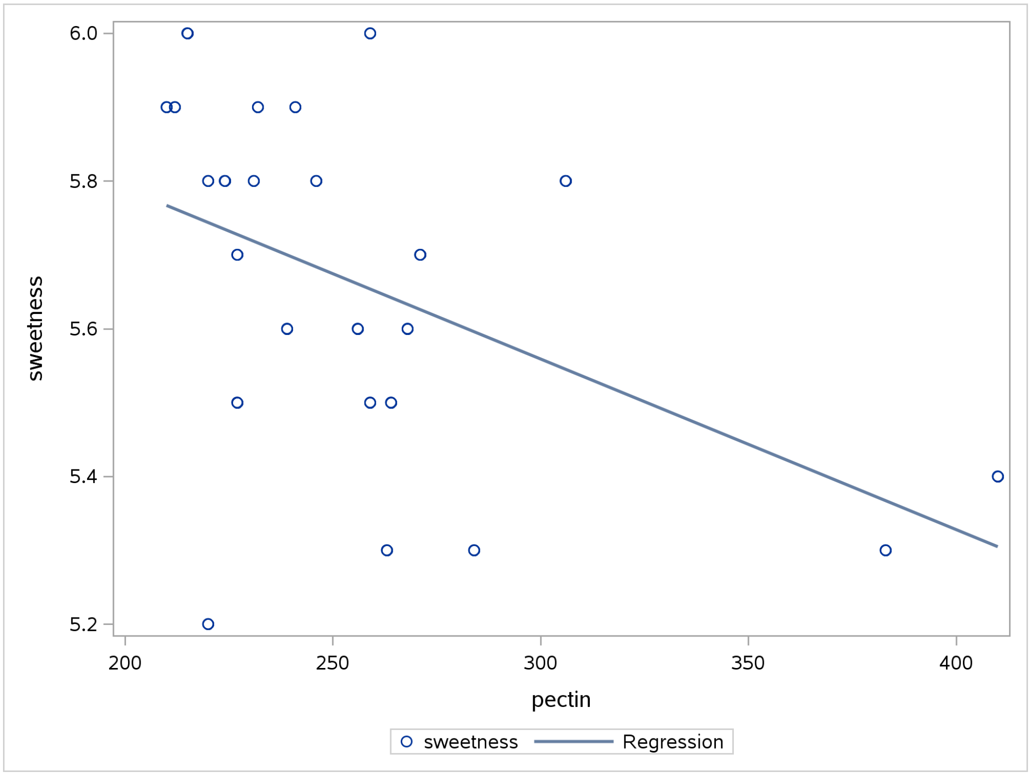


This came out the same as my R plot.

(c) Add a regression line to your plot. Does it go uphill or downhill?

Solution: What you do is to add a `reg` line to your `proc sgplot` with the same `x` and `y` as you used before:

```
proc sgplot;
  scatter x=pectin y=sweetness;
  reg x=pectin y=sweetness;
```



The trend goes downhill, largely I suspect because of those two high-pectin production runs that had low sweetness.

Extra: we had to repeat ourselves on the `reg` line because SAS will let you add *any* regression line to a plot, even one from a completely unrelated set of points!

2.3. In 2008, there were 30 teams that played professional baseball in North America. Fourteen of these teams played in the American League, and the other 16 in the National League. Each team played 162 games in total during the season, and I recorded the total number of runs scored by each team. The data are in `http://www.utsc.utoronto.ca/~butler/c32/runs.csv`.

(a) Using SAS, read in and display the data.

Solution: Use the URL and the `filename` idea:

```
filename myurl url 'http://www.utsc.utoronto.ca/~butler/c32/runs.csv';

proc import
  datafile=myurl
  dbms=csv
  out=mydata
  replace;
  getnames=yes;

proc print;
```

If you prefer, download and upload the data file to your SAS Studio file storage and read it in from there. I don't mind how you do it, but this way is the least work.

Here's what the data look like:

```
Obs    team                  league              runs

  1    Baltimore             American             782
  2    Boston                American             845
  3    Chicago White Sox     American             811
  4    Cleveland             American             805
  5    Detroit               American             821
  6    Kansas City           American             691
  7    Los Angeles Angels    American             765
  8    Minnesota             American             829
  9    New York Yankees      American             789
 10    Oakland               American             646
 11    Seattle               American             671
 12    Tampa Bay             American             774
 13    Texas                 American             901
 14    Toronto               American             714
 15    Arizona               National             720
 16    Atlanta               National             753
 17    Chicago Cubs          National             855
 18    Cincinnati            National             704
 19    Colorado              National             747
 20    Florida               National             770
 21    Houston               National             712
 22    Los Angeles Dodger    National             700
 23    Milwaukee             National             750
 24    New York Mets         National             799
 25    Philadelphia          National             799
 26    Pittsburgh            National             735
 27    San Diego             National             637
 28    San Francisco         National             640
 29    St Louis              National             779
 30    Washington            National             641
```

Oh, the Los Angeles Dodgers came out singular. Not sure how that happened.

Actually, I *do* know how that happened. If you want to know too, read on; otherwise skip to the next part.

SAS has been around for approaching 50 years. I think it was originally written in Fortran but is now written in C.[6] Anyway, in those languages, pieces of text are of a fixed length that you have to specify up front: “these names are of length 20”, or similar. How does `proc import` figure out what length to use? It actually reads the data file twice (or, at least, reads the first few lines the first time). It uses this first read to guess how long any pieces of text, like the team names here, are. The default number of lines it reads the first time is 20. So, according to SAS, the length of the text with the team names in it is the maximum team name length it found in the first 20 lines. This is the Los Angeles Angels. But the Los Angeles Dodgers have a name that is one character longer; it just happens not to be in the first 20 lines. So it got cut off to the same length as the Los Angeles Angels.

`proc import` has an option `guessingrows` that controls how many lines this first read is. If we set `guessingrows` to 25, the longest name in those 25 rows will be the Dodgers, and so it should get read in its entirety:

```
filename myurl url 'http://www.utsc.utoronto.ca/~butler/c32/runs.csv';

proc import
  datafile=myurl
  dbms=csv
  out=mydata
  replace;
  getnames=yes;
  guessingrows=25;

proc print;
```

```
Obs    team                  league            runs

  1    Baltimore             American           782
  2    Boston                American           845
  3    Chicago White Sox     American           811
  4    Cleveland             American           805
  5    Detroit               American           821
  6    Kansas City           American           691
  7    Los Angeles Angels    American           765
  8    Minnesota             American           829
  9    New York Yankees      American           789
 10    Oakland               American           646
 11    Seattle               American           671
 12    Tampa Bay             American           774
 13    Texas                 American           901
 14    Toronto               American           714
 15    Arizona               National           720
 16    Atlanta               National           753
 17    Chicago Cubs          National           855
 18    Cincinnati            National           704
 19    Colorado              National           747
 20    Florida               National           770
 21    Houston               National           712
 22    Los Angeles Dodgers   National           700
 23    Milwaukee             National           750
 24    New York Mets         National           799
 25    Philadelphia          National           799
 26    Pittsburgh            National           735
 27    San Diego             National           637
 28    San Francisco         National           640
 29    St Louis              National           779
 30    Washington            National           641
```

It works. We just got unlucky before.

The reason why `guessingrows` exists is that SAS is designed to work with large files, with millions of lines. Reading in such a file even once could take a long time, let alone if you read it twice to guess the length of text. Reading in only a small part of a file the first time is a reasonable compromise: it won't take very long, and it will usually produce a reasonable guess at how long text variables are (and if it doesn't, this is how you fix it up).

(b) Why do you think I stored the data in a `.csv` file rather than one where the data values are separated by spaces? Explain briefly.

Solution: Take a look at the data. Some of the team names are more than one word and have spaces *in* them, so if we used spaces to separate one value from the next, we wouldn't know whether, for example, "Chicago White Sox" is three values or one. In fact, we'd run into problems because each line of the data file won't have the same number of values: the line containing "Chicago White Sox" has five space-separated things (the three words of the team name, the number of runs and the league name) while the line containing "Toronto" only has three.

If you're wondering why "Chicago White Sox" but not "Toronto Blue Jays": Chicago has *two* major-league baseball teams, the other one being the Cubs, so the team name has to be used to distinguish them. Toronto has only one major-league team so there's no need to distinguish the Blue Jays from anyone else. (New York and Los Angeles also have two teams, as you see.)

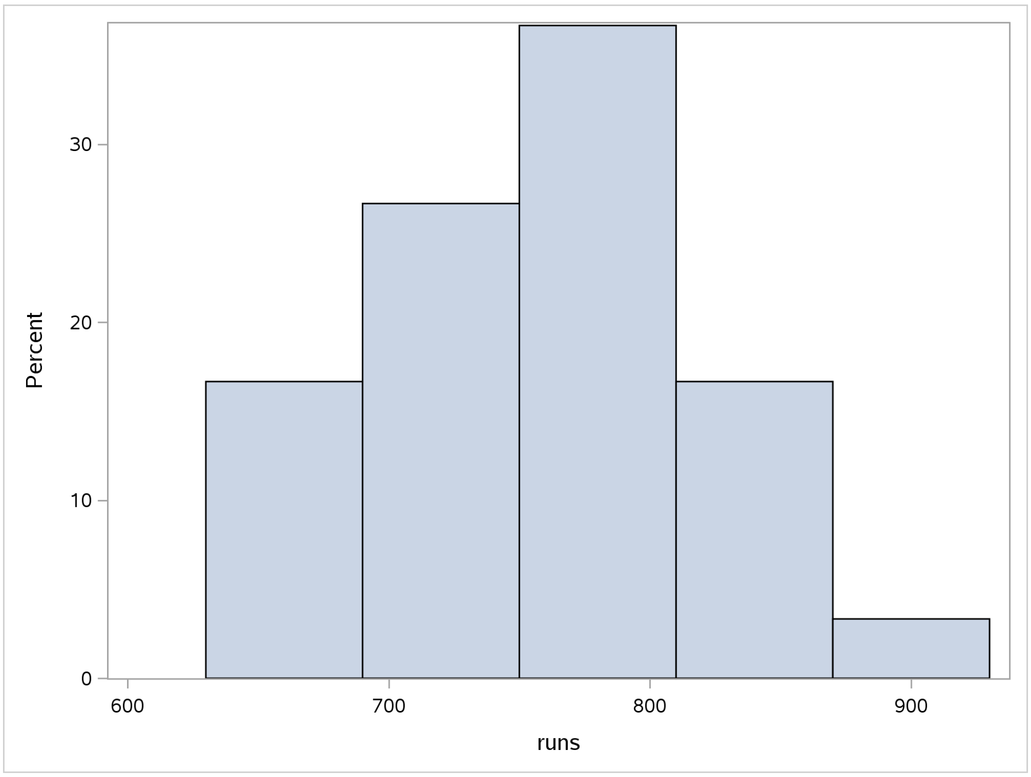
This was (when it was to be handed in) only one point, so if you've recognized that some of the team names have spaces in them, I'm good.

(c) Create a suitable graph to show the distribution of the number of runs scored by each team.

Solution: The obvious thing is a histogram:

```
proc sgplot;
  histogram runs;
```

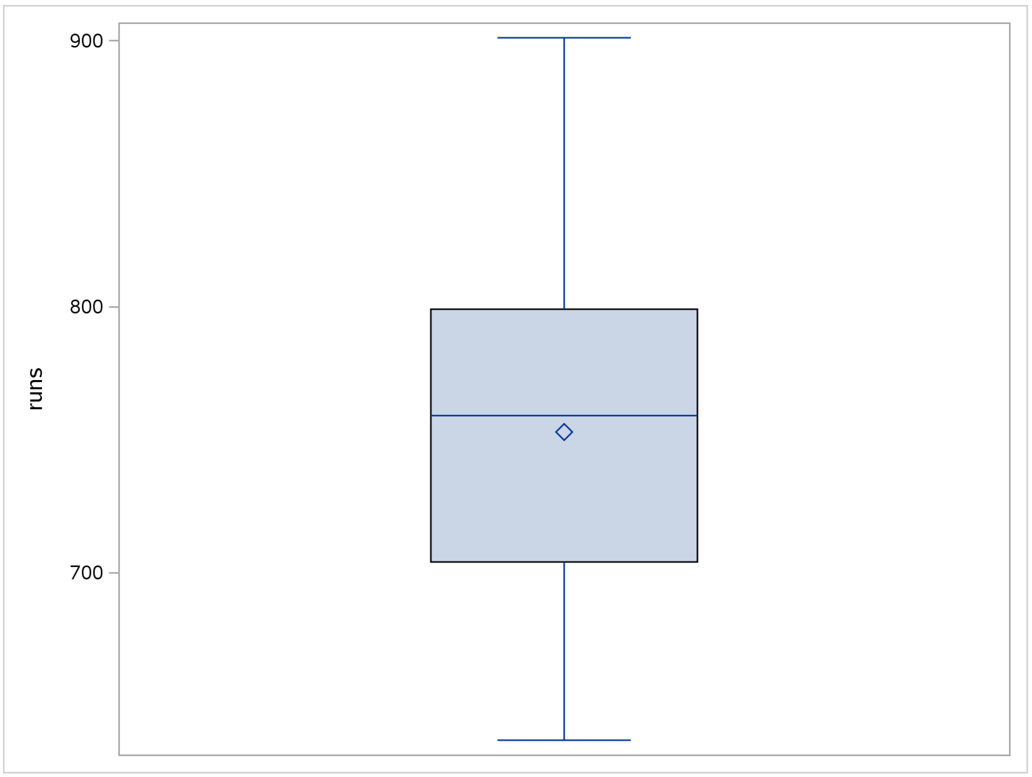
giving



Another possibility is a boxplot:

```
      proc sgplot;
        vbox runs;
```

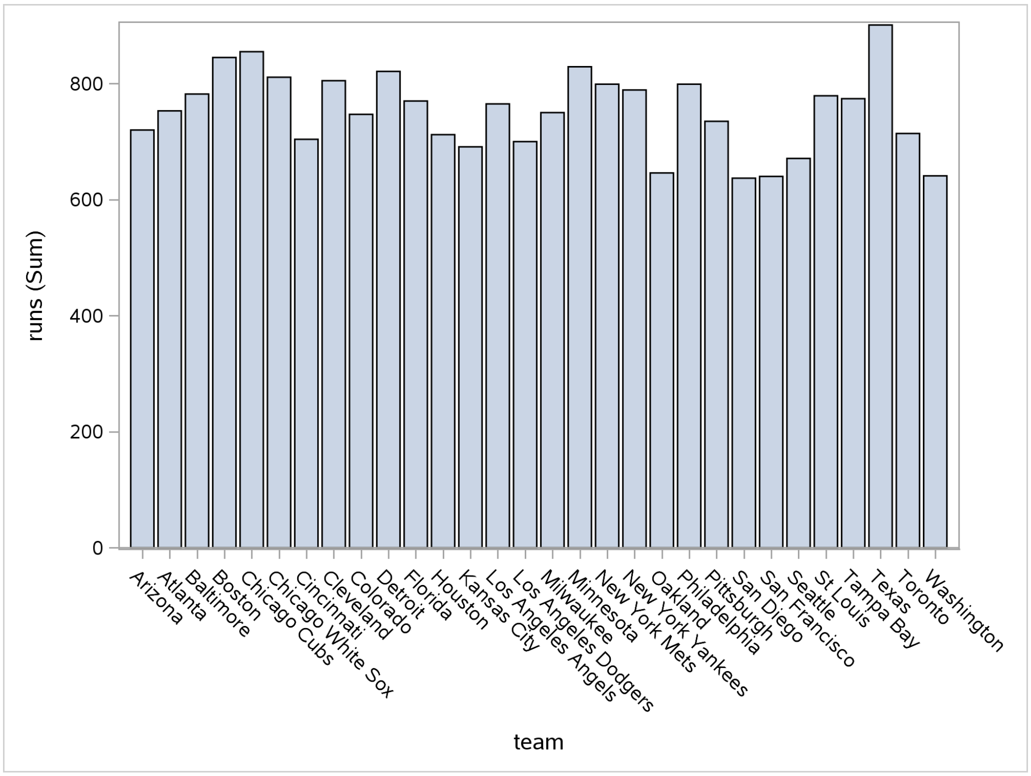
which gives



There's not really much to say about either of those. They're both pretty symmetric.

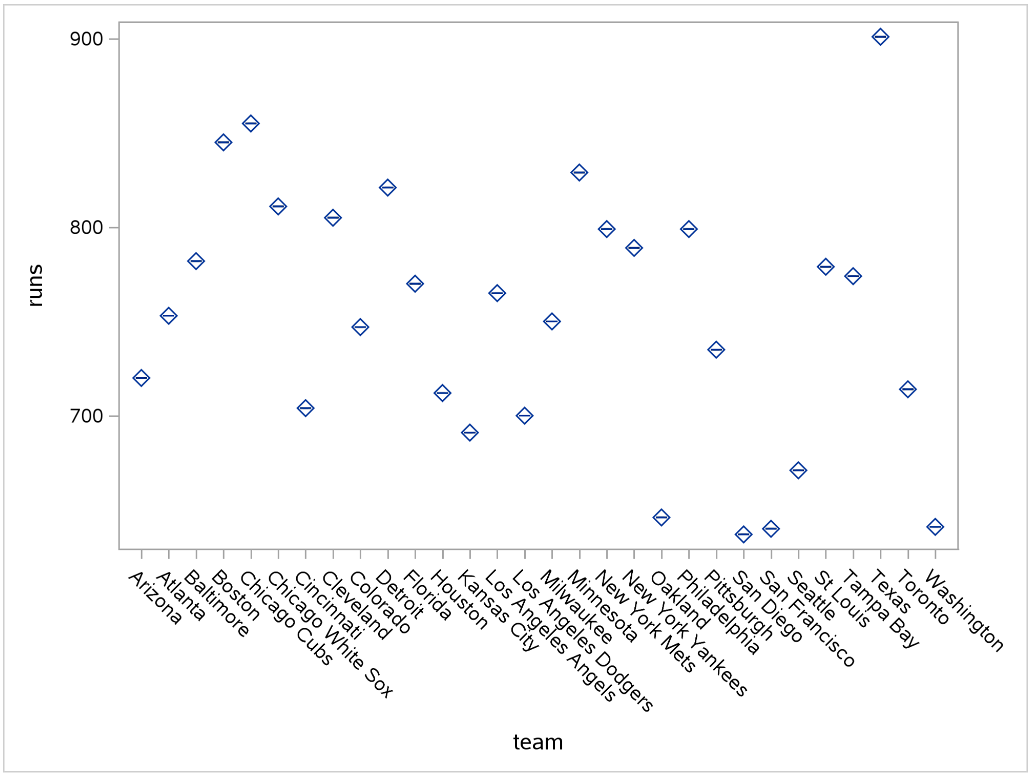
This was meant to be easy, but you could also read the question as asking you to show the team names on your graph as well, which makes it more difficult (and beyond what I showed you in class). I think the best graph along these lines would be a bar chart but instead of having frequency on the y axis, have the value of **`runs`**. That goes like this:

```
proc sgplot;
  vbar team / response=runs;
```



This shows that the numbers of runs vary from about 600 to about 900, and shows how many runs each team got. So I would accept this, or a rather odd-looking boxplot, thus:

```
proc sgplot;
  vbox runs / category=team;
```



The reason this looks strange is that each team produces only *one* number of runs. Thus the horizontal bar is the median of the one observation for each team, and the diamond is the mean of that one observation (and of course these are the same). Where this kind of plot would score is if you had the numbers of runs scored by each team *for each of several different years*, and then you would have a distribution over years that would have a genuine mean and median.

Each of these is a reasonable attempt to produce a graph according to your reading of the question. That's what this course is about.

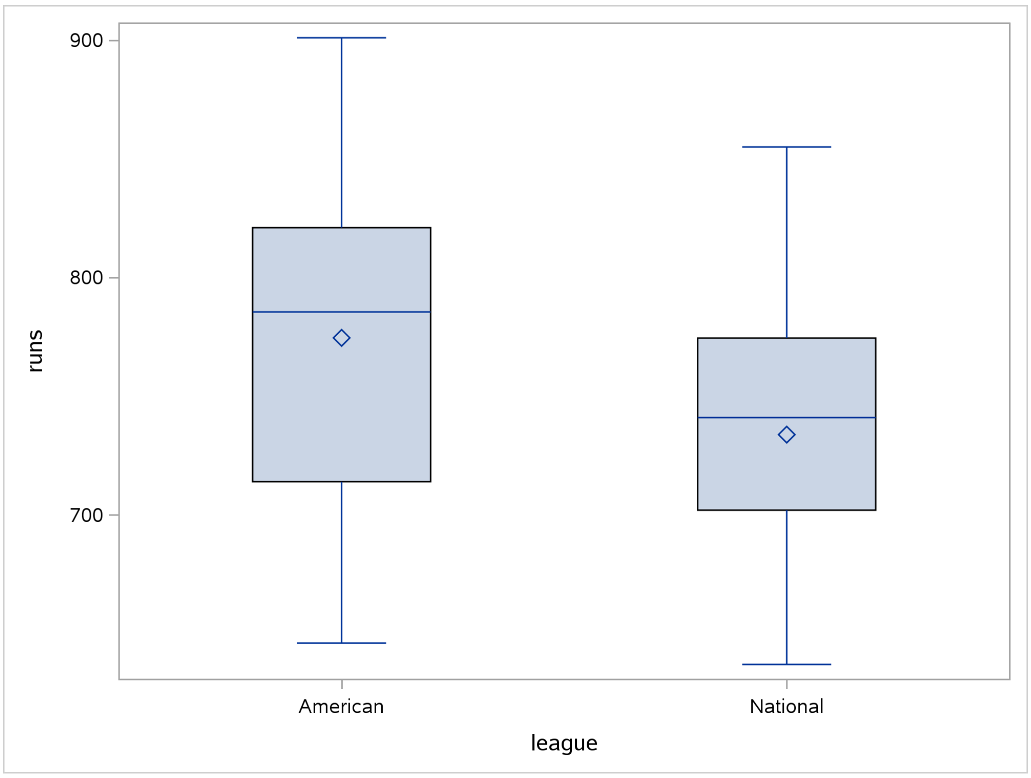
(d) The American League has a rule called the "designated hitter rule". This means that in games where an American League team is playing on their home field, both teams have a player who only bats (does not field), who bats in place of the pitcher. When a National League team is playing at home, the pitcher has to bat. Players who are good at pitching are not usually good at batting, so American League teams would be expected to score more runs on average than National League teams.

Make a suitable graph to compare the runs scored by the American and National League teams. Do you think that the Designated Hitter rule increases the number of runs, on average? Explain briefly.

Solution: The obvious graph is a boxplot (one quantitative variable `runs` and one categorical variable `league`):

```
proc sgplot;
  vbox runs / category=league;
```

which produces



I think that's a substantial difference in average, with the mean and median number of runs being noticeably higher for the American League (as predicted by the designated hitter rule).

If you want to, you can say that there is a lot of variability, and so the means/medians are not that different relative to how much variability there is. I don't think I like that conclusion so much, but it's a valid inference from the picture, so I can go with it. Once again, *make a call*, and then *support it*. If you do both of those properly, I'm happy.

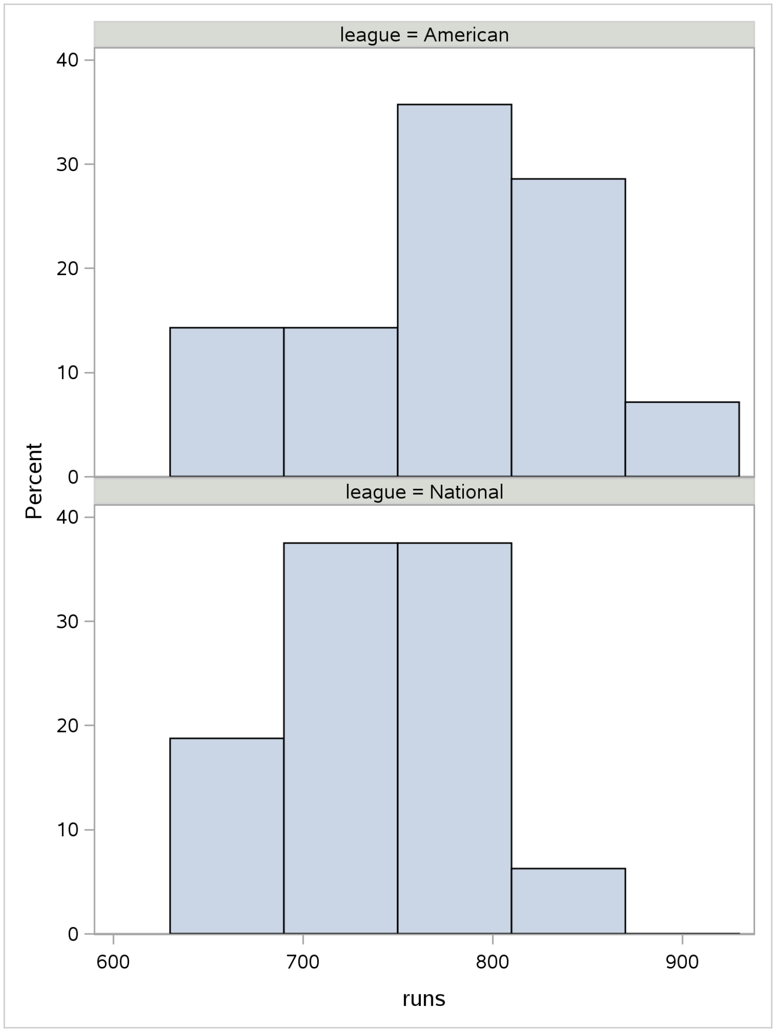
Or, you can say that there might be other factors that would also explain the difference between the two leagues (and then name one or two). The one that first comes to my mind is stadium size: it might be (I haven't investigated) that American league teams typically have smaller stadiums, in which it would be easier to hit home runs.

Another possibility, for the graph, is histograms above and below, which goes like this. The `columns=1` makes all the histograms (here 2 of them) come out one above another:

```
proc sgpanel;
  panelby league / columns=1;
  histogram runs;
```

The first two lines are the mechanism to get separate plots by, in this case, `league`; the rest of it is the same as you would feed into `proc sgplot`.

That produces:

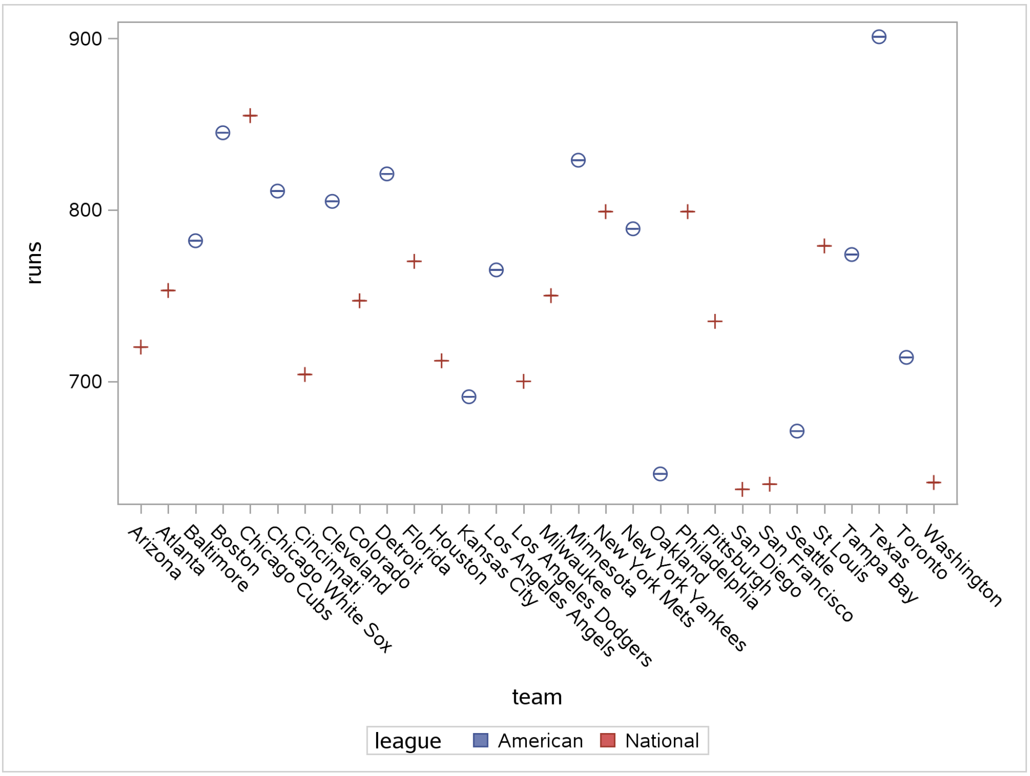


league = American
league = National
Percent
0
10
20
30
40
600
700
800
900
runs

Think about where the “centres” of those histograms are. For the American League on the left, the centre is somewhere near 800, I think, whereas for the National League on the right, the centre appears to be less, maybe around 750. It’s easier to compare boxplots than histograms, though, I’d say.

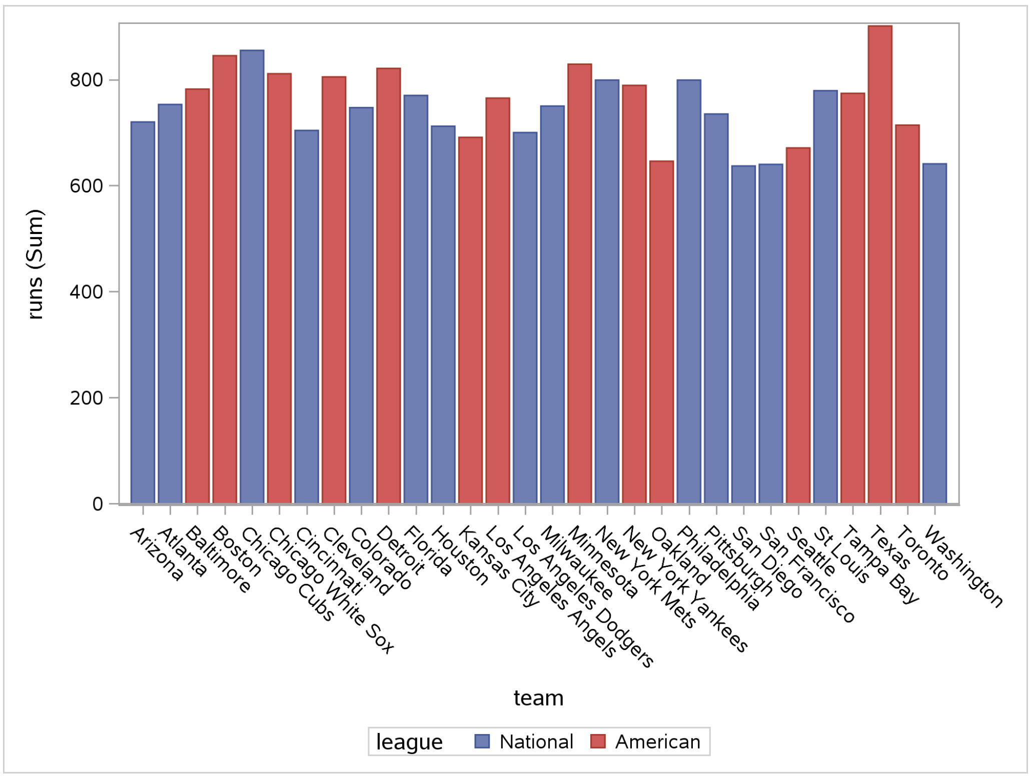
If you thought that you needed to show team names on your graph again, then you need to distinguish the leagues somehow. I think the easiest way to do that is to start from the not-really-boxplot:

```
proc sgplot;
  vbox runs / category=team group=league;
```



Then make a call about the average of the red values vs. the blue ones. This is a lot harder to do than on the boxplot. Or do this:

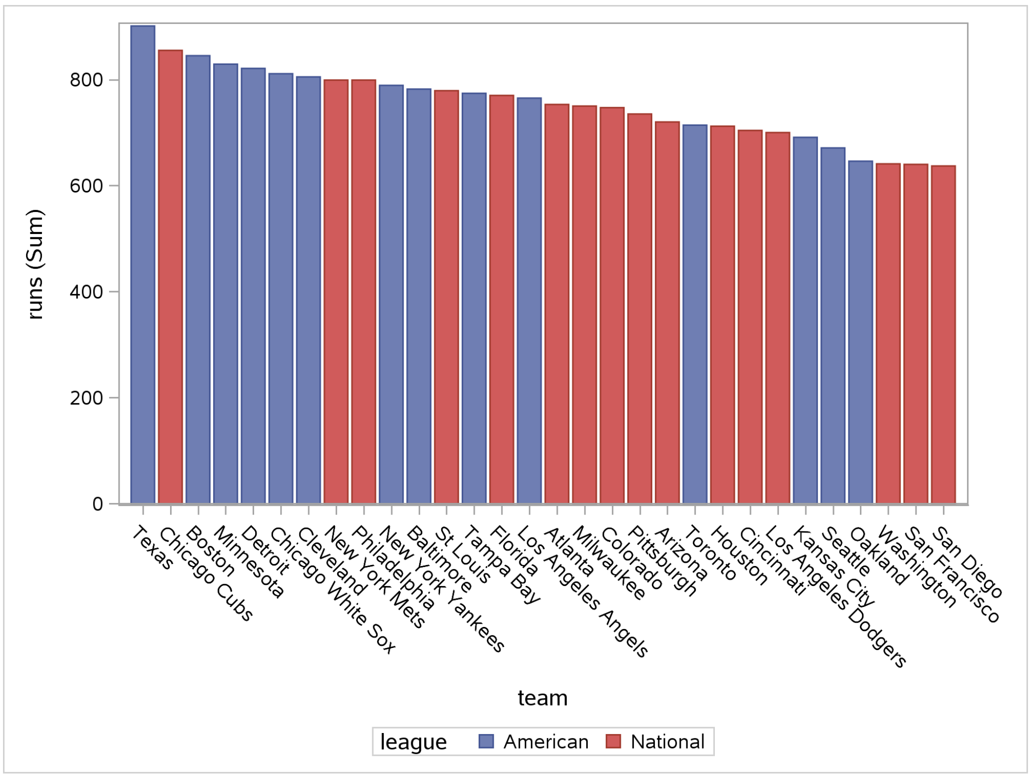
```
proc sgplot;
  vbar team / response=runs group=league;
```



Actually, I think this is a rather aesthetically pleasing graph. But I also think that a graph with the team names on it makes it more difficult to compare the overall run-scoring in the two *leagues*, which is what we really wanted to do. So I don't think a graph with team names on it, in this part, should be worth full marks. In my opinion, the boxplot is *much* the clearest way of comparing the two leagues.[7]

Let me try something else:

```
proc sgplot;
  vbar team / response=runs group=league categoryorder=respdesc;
  xaxis discreteorder=data;
```



This piece of gadgetry sorts the bars into order by number of runs. I seem to need both the `categoryorder` and the `discreteorder`, and I'm not sure why. Anyway, the teams with the most runs are on the left. The value of this plot is that you can eyeball it to see whether the blue bars (American League) are mostly on the left and the red bars (National League) are mostly on the right, which I think they kind of are (especially if you think of the Chicago Cubs as being an outlier among the National League teams).

You might be wondering whether that's a *significant* difference in means between the two leagues. That's inference, which in SAS we haven't done yet, but to get the flavour, it's a two-sample t-test, which we ought to be happy with since the distributions are pretty symmetric:

```
proc ttest sides=U;
  var runs;
  class league;
```

with output

```
league          N        Mean     Std Dev     Std Err     Minimum     Maximum

American       14       774.6     71.5657     19.1267       646.0       901.0
National       16       733.8     61.5513     15.3878       637.0       855.0
Diff (1-2)            40.7589     66.3890     24.2959
   league        Method               Mean       95% CL Mean        Std Dev

   American                          774.6       733.3    815.9     71.5657
   National                          733.8       701.0    766.6     61.5513
   Diff (1-2)    Pooled            40.7589     -0.5715  Infty       66.3890
   Diff (1-2)    Satterthwaite     40.7589     -1.1183  Infty

               league        Method            95% CL Std Dev

               American                       51.8818    115.3
               National                       45.4682  95.2624
               Diff (1-2)    Pooled           52.6849  89.7879
               Diff (1-2)    Satterthwaite
          Method           Variances        DF    t Value    Pr > t

          Pooled           Equal            28       1.68    0.0523
          Satterthwaite    Unequal      25.879       1.66    0.0545
                            Equality of Variances

              Method      Num DF    Den DF    F Value    Pr > F

              Folded F        13        15       1.35    0.5711
```

I did a one-sided test (that's the `sides` thing) because I suspected without looking at the data that the mean would be higher for the American League. Remember when we did this with R, there were two flavours of two-sample t-test to choose from: the Welch-Satterthwaite one, which made no assumption about the spreads of the two groups, and the pooled one (done with `var.equal`) which assumed that the two groups had the *same* spread (strictly, variance).

SAS being SAS, it gives you both t-test results and lets you pick out the one you want. (This is the SAS way: you get a ton of output, and you choose what you want and discard the rest.) In our boxplot, it looked as if the distribution of runs for the American league had a bigger spread, so the appropriate P-value is the one labelled Satterthwaite down near the bottom, which is 0.0545.

This is not quite less than the standard α of 0.05, so we don't *quite* have enough evidence to say that the designated hitter rule increases the mean number of runs.

I did a bit more research and found that things are a bit muddier than this because there are (and were in 2008) "interleague games", that is, games played between one team in the American League and one in the National League. In an interleague game, it depends on which team is playing at home whether there is a Designated Hitter or not. So there are games played by National League teams that *do* have a designated hitter, and games played by American League teams that *do not*. So if we are going to look at the effect of the Designated Hitter rule properly, we should investigate *game by game*, rather than aggregating by team as we did here. The data are out there, but require more work to organize. See, for example, the graph at the top of page 12 of `http://tigerprints.clemson.edu/cgi/viewcontent.cgi?article=1878&context=all_theses`. According to that, the American League has consistently had more runs per game since 1973 when the rule was introduced.[8]

`http://www.baseball-reference.com/leagues/MLB/2008-schedule.shtml` has all the game scores for the entire season, but they need some processing to be ready for analysis. I wanted to see how this worked out, so I wrote a blog post about it at `https://nxskok.github.io/docs/2017/06/08/the-designated-hitter/`. This is in R rather than SAS, so I think you'll be able to figure out most of what I did.

3 The next bunch: exploring data

3.1. Let's re-use the North Carolina births data set to answer some similar questions in SAS to the ones we answered in R before. Recall that the data in file `http://www.utsc.utoronto.ca/~butler/c32/ncbirths.csv` were about 500 randomly chosen births of babies in North Carolina. There is a lot of information: not just the weight at birth of the baby, but whether the baby was born prematurely, the ages of the parents, whether the parents are married, how long (in weeks) the pregnancy lasted (this is called the "gestation") and so on.

(a) Read the data into SAS. Your reading in will have to respect what kind of data you have, and where you are getting it from. You should use `proc print` until you are confident that the data have been read in correctly, but the output from that is *very* long, so take out the `proc print` when you are happy.

Solution: This is a `.csv` file, so the `dbms` in `proc import` will have to say that. Also, we're reading from a website, so we should do the `filename` thing first:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/ncbirths.csv";
```

```
proc import
  datafile=myurl
  dbms=csv
  out=ncbirths
  replace;
  getnames=yes;
```

It is likely that this won't work the first time, so you should probably glue a `proc print` on the end of that until you are satisfied. Here's the first 20 lines of mine:

```
proc print data=ncbirths(obs=20);
```

```
                                             Weeks_     Pre_natal_       Marital_
Obs    Father_Age     Mother_Age         Gestation         Visits         Status

  1            27             26                38             14              1
  2            35             33                40             11              1
  3            34             22                37             10              2
  4             .             16                38              9              2
  5            35             33                39             12              1
  6            32             24                36             12              1
  7            33             33                38             15              2
  8            38             35                38             16              1
  9            28             29                40              5              1
 10             .             19                34             10              2
 11            28             26                39             15              1
 12            34             31                39             15              1
 13             .             19                34              0              2
 14             .             14                42             15              2
 15             .             18                42             15              2
 16            28             18                38             15              2
 17            33             20                39             15              2
 18            22             20                39             14              1
 19             .             22                37              9              2
 20            28             26                40             14              2

          Mother_
          Weight_                            Weight__
Obs        Gained   Low_Birthweight_          pounds_        Premie_   Few_Visits_

  1            32                  0            6.875              0             0
  2            23                  0           6.8125              0             0
  3            50                  0             7.25              0             0
  4             .                  0           8.8125              0             0
  5            15                  0           8.8125              0             0
  6            12                  0           5.8125              1             0
  7            60                  0           6.5625              0             0
  8             2                  0           10.125              0             0
  9            20                  0            7.375              0             1
 10             .                  1            2.875              1             0
 11            45                  0           7.1875              0             0
 12            22                  0           8.6875              0             0
 13            20                  0            5.875              1             1
 14            20                  0            7.875              0             0
 15            27                  0           7.4375              0             0
 16            30                  0                6              0             0
 17            41                  0             8.25              0             0
 18            25                  0           9.9375              0             0
 19            41                  0             6.25              0             0
 20            21                  0           5.9375              0             0
```

Where you see a dot instead of a number in the output, the value is missing (not recorded). The father's age was often missing, and the mother's weight gained was sometimes missing. This plays out below.

This works for me, but it may not work for you. In the online SAS Studio, the variables get read in with variable names including spaces and question marks. The question then becomes how you refer to them later. There appear to be two ways around this:

1. when you need to use a variable name with a space or question mark in it, surround it by *single* quotes *and* put the letter n on the end. This is known in the SAS world as a "name literal". You need to do this every time you use every such variable (and thus it is a big pain in the neck). When I do it, a variable thus referred to in the code editor is shown in teal green. For example, `'weight (pounds)'n`. It has to be single quotes and it has to have an `n` after.
2. perhaps better, put this line at the top of your code:

   ```
   options validvarname=v7;
   ```

   That will turn all the column names into "valid variable names", by replacing the spaces and brackets and question marks with underscores, the same as happened for me automatically. This seems to be a system option: my system has it already set, SAS Studio doesn't.

In the virtual-machine SAS Studio (the one that runs under VirtualBox on your own computer), it seems to work the same as it did for me above.

(b) Run `proc means` on all your quantitative variables. Why are there fewer than 500 observations for some of your variables? (You might like to look back at your `proc print` output to figure this out.)

Solution:

```
proc means;
```

```
                                The MEANS Procedure

   Variable                   N            Mean         Std Dev         Minimum
   ----------------------------------------------------------------------------
   Father_Age               420      30.0214286       6.6506423      16.0000000
   Mother_Age               500      26.8820000       6.3542765      13.0000000
   Weeks_Gestation          499      38.3326653       3.0122833      20.0000000
   Pre_natal_Visits         498      12.2068273       3.9216924               0
   Marital_Status           500       1.3760000       0.4848651       1.0000000
   Mother_Weight_Gained     487      30.4004107      14.1769331               0
   Low_Birthweight_         500       0.1080000       0.3106913               0
   Weight__pounds_          500       7.0687500       1.5062001       1.1875000
   Premie_                  499       0.1503006       0.3577244               0
   Few_Visits_              498       0.0642570       0.2454568               0
   ----------------------------------------------------------------------------

                       Variable                     Maximum
                       ------------------------------------
                       Father_Age                50.0000000
                       Mother_Age                50.0000000
                       Weeks_Gestation           45.0000000
                       Pre_natal_Visits          30.0000000
                       Marital_Status             2.0000000
                       Mother_Weight_Gained      75.0000000
                       Low_Birthweight_           1.0000000
                       Weight__pounds_           11.6250000
                       Premie_                    1.0000000
                       Few_Visits_                1.0000000
                       ------------------------------------
```

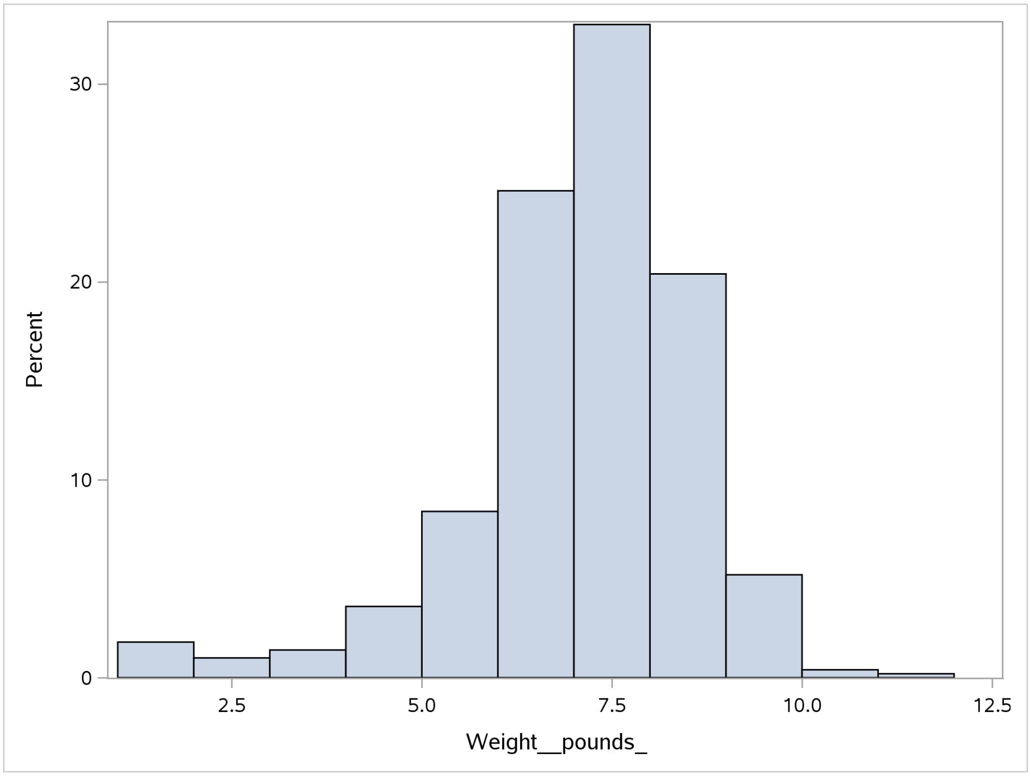
The value in the N column is the number of observations for each variable. Or, more precisely, since we know there were 500 rows, these are the number of *non-missing* observations for each variable. Looking at all 500 rows, the father's age was the most often missing, something we would have guessed from our scan above of the first 20 rows of the dataset.

Note that the variable names have gained *underscores* in them, where the spaces and brackets were (since SAS variable names cannot have either of those, and the equivalent to R's "backtick" thing is "name literals" that we were trying to avoid). So we have to remember to use the underscores below.

(c) Make a histogram of the birth weights. Do you have to worry about the number of bins for the histogram? How many bins did SAS choose? Does the distribution look approximately normal, and if not, how not?

Solution: Lots of questions to answer. The first answer is that SAS chooses the number of bins on a histogram for itself, so we don't have to worry about that. Don't forget to include the right number of underscores: two between `Weight` and `pounds`, and one after:

```
proc sgplot;
  histogram Weight__pounds_;
```

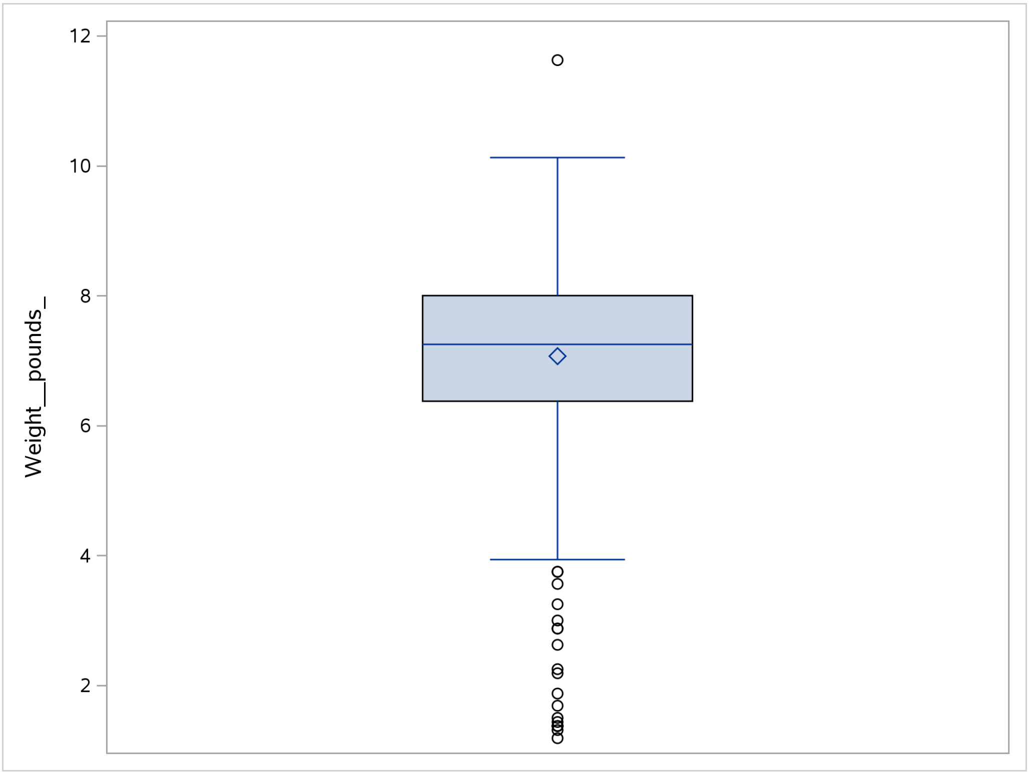


SAS chose 11 bins (with bin width 1 and bin boundaries on the whole numbers). This is similar to the 10 bins that Sturges' rule gives.

This (for me at least) has the same overall look as the `ggplot` histogram: it has a more or less normally-distributed look, but with too many extra data values at the bottom. (There should be basically *no* values down below 2.5 pounds, but there are several.)

Extra: a boxplot offers some additional insight:

```
proc sgplot;
  vbox Weight__pounds_;
```



There is one outlier at the top, and *a lot* of outliers at the bottom. The question to ask yourself when you have as many outliers as this is "are they errors, or are they legit values that would be expected to be different?"

The reason for the extra values at the bottom is that these are (usually) different kinds of births, as we will see. The issue when you have outliers is *not* an automatic reaction of "these are outliers: they must be removed", but to stop and think about *reasons* why these values are outliers. You might have two (or more) sets of values collected under different conditions, all mixed up. We explore this below.

3.2. This is an exploration of some extra issues around the North Carolina births data set.

(a) Use SAS to find the mean birth weight according to whether the birth was premature or not. Are full-term (not-premature) babies typically heavier? How do you know?

Solution: Add this code to the end of the code you ran before, with the `proc import` in it.

This one is just `proc means`, with the right variable names:

```
proc means;
  var Weight__pounds_;
  class Premie_;
```

```
                                The MEANS Procedure

                         Analysis Variable : Weight__pounds_

                  N
      Premie_   Obs     N          Mean       Std Dev       Minimum       Maximum
    -----------------------------------------------------------------------------
            0   424   424     7.4046285     1.0884856     3.7500000    11.6250000

            1    75    75     5.1683333     2.0538818     1.1875000     9.2500000
    -----------------------------------------------------------------------------
```

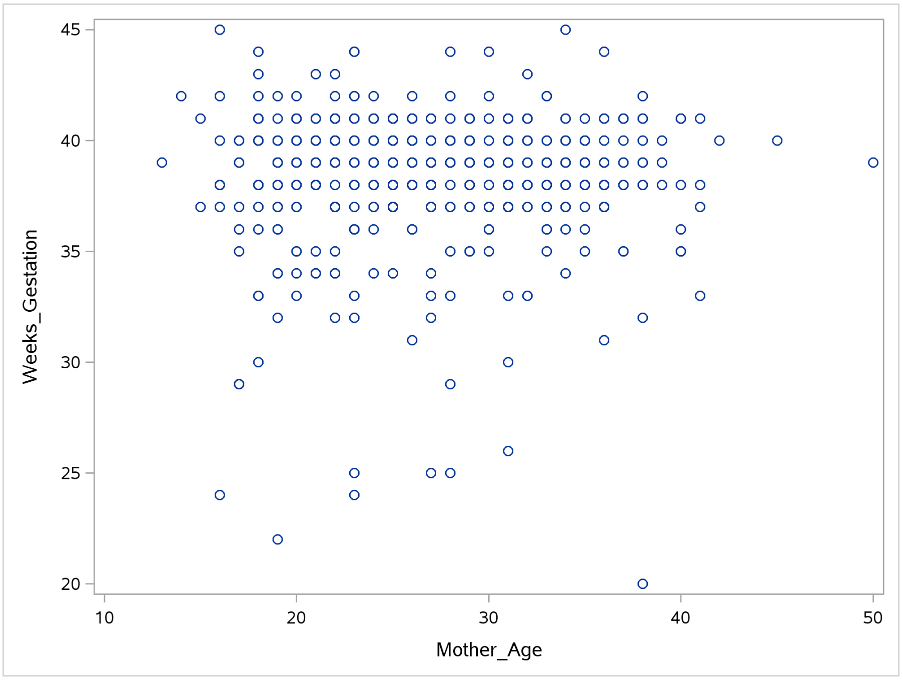
Yes, full-term babies weigh an average of 7.40 pounds, while premature babies have a mean weight of only 5.17 pounds. (Your answer ought to be a bit more than "yes": how do you know the answer is "yes"?)

You can see also that the premature babies have a greater *spread* of birth weights as well: the standard deviation is almost twice as big. This might be because premature babies could be premature for a mixture of different reasons, such as health reasons that mean it is safest for the mother to give birth early, or because the baby is finished growing early and needs to come out!

(b) Is a premature birth more likely when the mother is older? Assess this by looking for a relationship between gestation and mother's age, obtaining a suitable plot. Use SAS.

Solution: What I had in mind was a scatterplot, since these two variables are both numerical. (Looking at the proportion of premature births by age is trickier, since it needs contingency tables or similar.)

```
proc sgplot;
  scatter y=Weeks_Gestation x=Mother_Age;
```

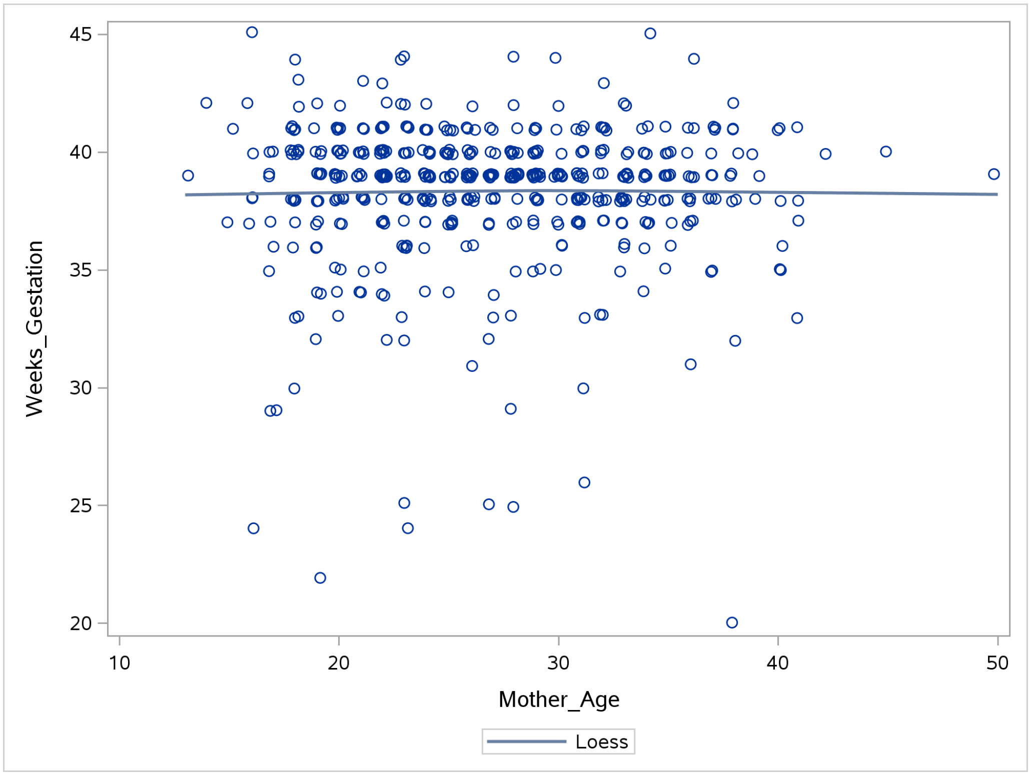

Weeks_Gestation
Mother_Age
45
40
35
30
25
20
10
20
30
40
50

A *really* premature birth (one where the gestation is less than about 30 weeks) seems to be *less* likely when the mother is older. But taking the definition of “premature” as “less than 37 weeks”, the picture is less clear. Most of the mothers are younger, and in terms of actual numbers of premature births, these are higher when the mother is younger too. So it’s not clear.

What you conclude is up to you. I am most interested in some sensible discussion that supports your conclusion, whatever it is. If you think there is no clear relationship, you need to say so. Your data-analytic career will be full of pictures like this for you to try to make sense of.

There is another issue here: the mother’s age and the weeks of gestation are both whole numbers, so it is possible that two different mothers could have been the same age and gestation period, and the points would have plotted over each other on the scatterplot. This is hard to diagnose, but we can eyeball it: most of the mothers’ ages cover about a 30-year span, and most of the gestations cover about a 15-week period. So there are somewhere around $30 \times 15 = 450$ age-gestation combinations. But there are 500 births, so there are bound to be repeats somewhere. One way to work around that is to put a “loess curve” on the plot (we’ll learn more about this later) which tells you something about the overall trend without assuming that it is linear. This permits an option “jitter” that moves the points slightly so that we can see any overlaid ones. That would look like this:

```
proc sgplot;
  loess y=Weeks_Gestation x=Mother_Age / jitter;
```



Weeks_Gestation
45
40
35
30
25
20
10
20
30
40
50
Mother_Age
Loess

The jitter has had the biggest effect. You can see that there were a *lot* of overlaid points, especially for mother's age around 30 and gestation just under 40. The loess actually goes almost exactly straight across, which says that there is actually *no* relationship between mother's age and length of gestation. Those very short gestations were, as we now see, very small in number compared to the bulk of the data; the vast majority of the pregnancies were of more or less normal length, and for those there is no relationship at all between gestation and mother's age.

There is a kind of "cheating" way to get the proportion of premature births by age. It uses the idea that the mean of a 0-1 variable is the proportion of 1's in it. The `var` and the `class` below look the wrong way around, since `age` is quantitative and `premie` is really categorical, but it does the right thing: "give me the mean value of `premie` for each (group defined by) mother's age":

```
proc means;
  var Premie_;
  class Mother_Age;
```

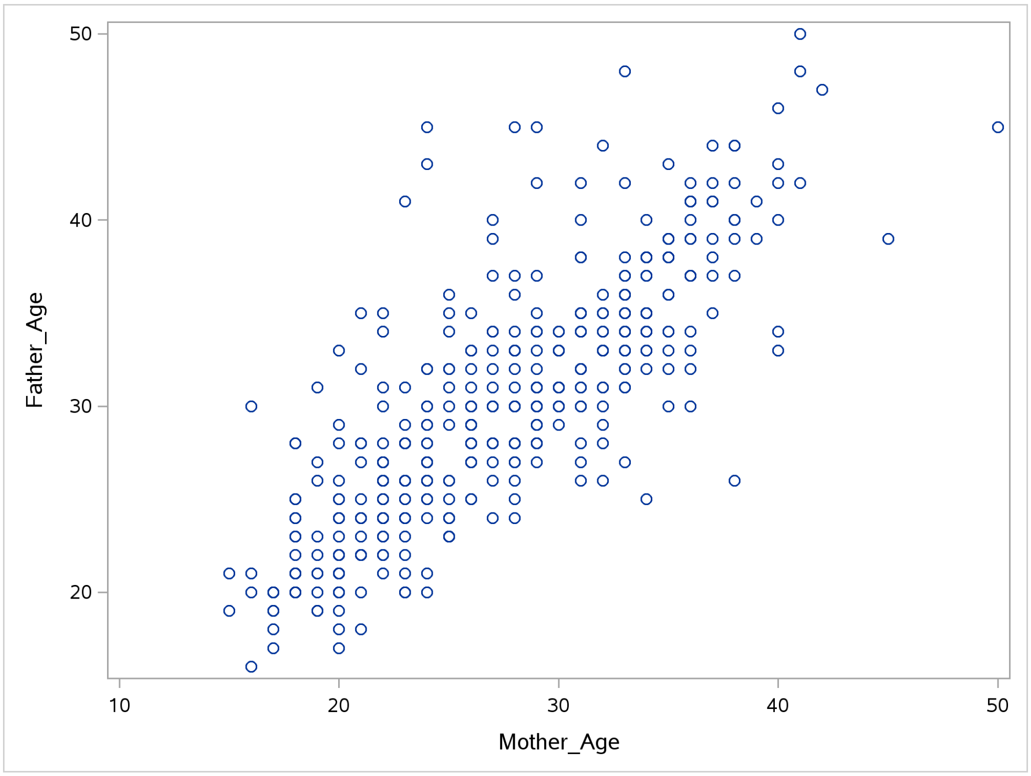
```
                          The MEANS Procedure

                     Analysis Variable : Premie_

                 N
 Mother_Age    Obs     N          Mean       Std Dev       Minimum       Maximum
 ------------------------------------------------------------------------------
         13      1     1             0             .             0             0

         14      1     1             0             .             0             0

         15      2     2             0             0             0             0

         16      7     7     0.1428571     0.3779645             0     1.0000000

         17      9     9     0.4444444     0.5270463             0     1.0000000

         18     23    23     0.1739130     0.3875534             0     1.0000000

         19     21    21     0.2857143     0.4629100             0     1.0000000

         20     31    31     0.1290323     0.3407771             0     1.0000000

         21     19    19     0.1578947     0.3746343             0     1.0000000

         22     34    34     0.1176471     0.3270350             0     1.0000000

         23     28    28     0.3214286     0.4755949             0     1.0000000

         24     26    26     0.0769231     0.2717465             0     1.0000000

         25     25    25     0.0400000     0.2000000             0     1.0000000

         26     26    26     0.1153846     0.3258126             0     1.0000000

         27     21    21     0.1904762     0.4023739             0     1.0000000

         28     28    28     0.1428571     0.3563483             0     1.0000000

         29     27    27     0.0740741     0.2668803             0     1.0000000

         30     17    17     0.1764706     0.3929526             0     1.0000000

         31     26    26     0.1153846     0.3258126             0     1.0000000

         32     19    18     0.1111111     0.3233808             0     1.0000000

         33     24    24     0.1250000     0.3378320             0     1.0000000

         34     19    19     0.1052632     0.3153018             0     1.0000000

         35     15    15     0.1333333     0.3518658             0     1.0000000

         36     14    14     0.0714286     0.2672612             0     1.0000000

         37     11    11     0.1818182     0.4045199             0     1.0000000

         38     10    10     0.2000000     0.4216370             0     1.0000000

         39      3     3             0             0             0             0

         40      6     6     0.5000000     0.5477226             0     1.0000000

         41      4     4     0.2500000     0.5000000             0     1.0000000

         42      1     1             0             .             0             0

         45      1     1             0             .             0             0

         50      1     1             0             .             0             0
 ------------------------------------------------------------------------------
```

This isn't clear either, the interpretation not being helped by there being very few mothers that are very young or very old. Confining our attention to the ages where there are at least 20 mothers, two young ages (19 and 23) have a lot of premature births (29% and 32% respectively) and there are some higher ages (such as 29 and 33) where the proportion of premature births is less than 10%. But is that just a quirk of these data? I think it is.

(c) The father's age is often not known in this data set, but when it is, does a large mother's age tend to go with an large father's age? Use SAS to draw a picture or obtain a number that helps you decide. (You might find Chapter 10 of the SAS text helpful, if you have it.) What do you conclude?

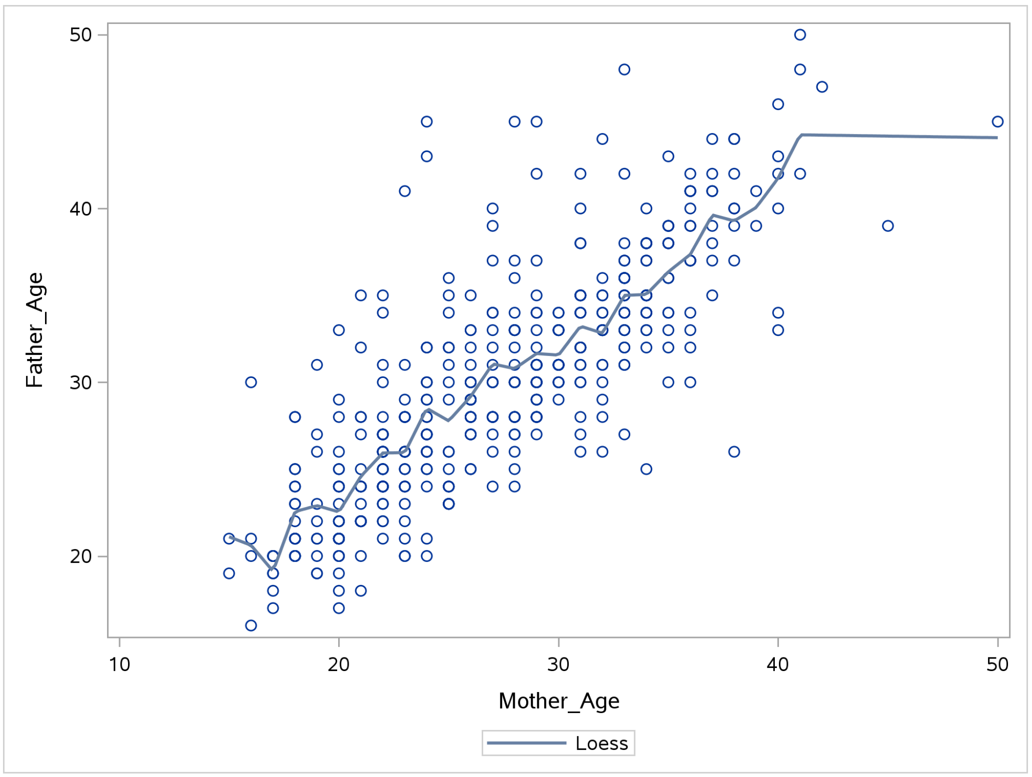
Solution: This one is also most obviously a scatter plot:

```
proc sgplot;
  scatter y=Father_Age x=Mother_Age;
```



Or, in the light of our previous investigations, you might think of putting a loess on it (replace `scatter` by `loess`):

```
proc sgplot;
  loess y=Father_Age x=Mother_Age;
```



The loess wiggles a fair bit, and reacts rather a lot to that mother of age 50 (!), but the pattern is generally straight (not obviously curved).

You could have done mother's and father's age the other way around on your scatterplot. There's no reason why one is explanatory and the other is response.

This is about as clear a trend as you could wish to see for this kind of dataset. When the mother is older, the father tends to be older as well, and younger with younger. It is also a pretty nearly linear trend.

With that in mind, you might also have thought about calculating a correlation. You'll have to do some investigating to find that `proc corr` does this. See, for example, `https://support.sas.com/documentation/cdl/en/procstat/63104/HTML/default/viewer.htm#procstat_corr_sect003.htm`, or Chapter 10 of our SAS text:

```
proc corr;
  var Mother_Age Father_Age;
```

```
                                The CORR Procedure

                       2  Variables:    Mother_Age Father_Age
                                Simple Statistics

Variable           N         Mean     Std Dev         Sum     Minimum     Maximum

Mother_Age       500     26.88200     6.35428       13441    13.00000    50.00000
Father_Age       420     30.02143     6.65064       12609    16.00000    50.00000
                        Pearson Correlation Coefficients
                           Prob > |r| under H0: Rho=0
                              Number of Observations

                                      Mother_       Father_
                                          Age           Age

                     Mother_Age       1.00000       0.80538
                                                     <.0001
                                          500           420

                     Father_Age       0.80538       1.00000
                                       <.0001
                                          420           420
```

The correlation is about 0.80. With this much data, there is no doubt at all that there is a (positive) relationship: older mother goes with older father. (That <0.0001 under the correlation is a P-value for testing the null hypothesis that the (population) correlation is zero, against the alternative that it is not zero. This one is strongly significantly nonzero.)

If you don't like this, you can also do a regression (with either variable as response and the other as explanatory). Again, this is getting ahead of ourselves, but it doesn't take much searching to unearth `proc reg` (this is also in Chapter 10 of the SAS text):

```
proc reg;
  model Father_Age=Mother_Age;
```

```
                              The REG Procedure
                                Model: MODEL1
                        Dependent Variable: Father_Age

              Number of Observations Read                         500
              Number of Observations Used                         420
              Number of Observations with Missing Values           80
                             Analysis of Variance

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

Model                     1            12021          12021     771.68    <.0001
Error                   418       6511.61963       15.57804
Corrected Total         419            18533
             Root MSE              3.94690    R-Square     0.6486
             Dependent Mean       30.02143    Adj R-Sq     0.6478
             Coeff Var            13.14695
                             Parameter Estimates

                         Parameter       Standard
    Variable      DF      Estimate          Error    t Value    Pr > |t|

    Intercept      1       6.70430        0.86119       7.78      <.0001
    Mother_Age     1       0.85329        0.03072      27.78      <.0001
```

A significantly positive slope, meaning that older mother goes with older father. Again, you could have response and explanatory variable either way around.

I like the scatterplot best, because it allows us to see the kind of relationship we have, rather than just assuming it is linear.

3.3. Nenana, Alaska, is about 50 miles west of Fairbanks. Every spring, there is a contest in Nenana, called the Ice Classic. A wooden tripod is placed on the frozen river, and people try to guess the exact minute when the ice melts enough for the tripod to fall through the ice. The contest started in 1917 as an amusement for railway workers, and has taken place every year since. Now, hundreds of thousands of people enter their guesses on the Internet and the prize for the winner can be as much as $300,000.

Because so much money is at stake, and because the exact same tripod is placed at the exact same spot on the ice every year, the data are consistent and accurate. The data are in `http://www.utsc.utoronto.ca/~butler/c32/nenana.txt`.

The Ice Classic has its own website at `http://www.nenanaakiceclassic.com/`.

We did this before in R.

(a) Read in the data set and assess what you have for reasonableness. There are a lot of observations, so calculate the mean, SD, min and max for each variable.

Solution: Reading in the data involves that thing to get "separated by tab". The last sentence of the question suggests to run `proc means` by way of checking the data:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/nenana.txt";

proc import
  datafile=myurl
  dbms=dlm
  out=mydata
  replace;
  delimiter='09'x;
  getnames=yes;

proc means;
```

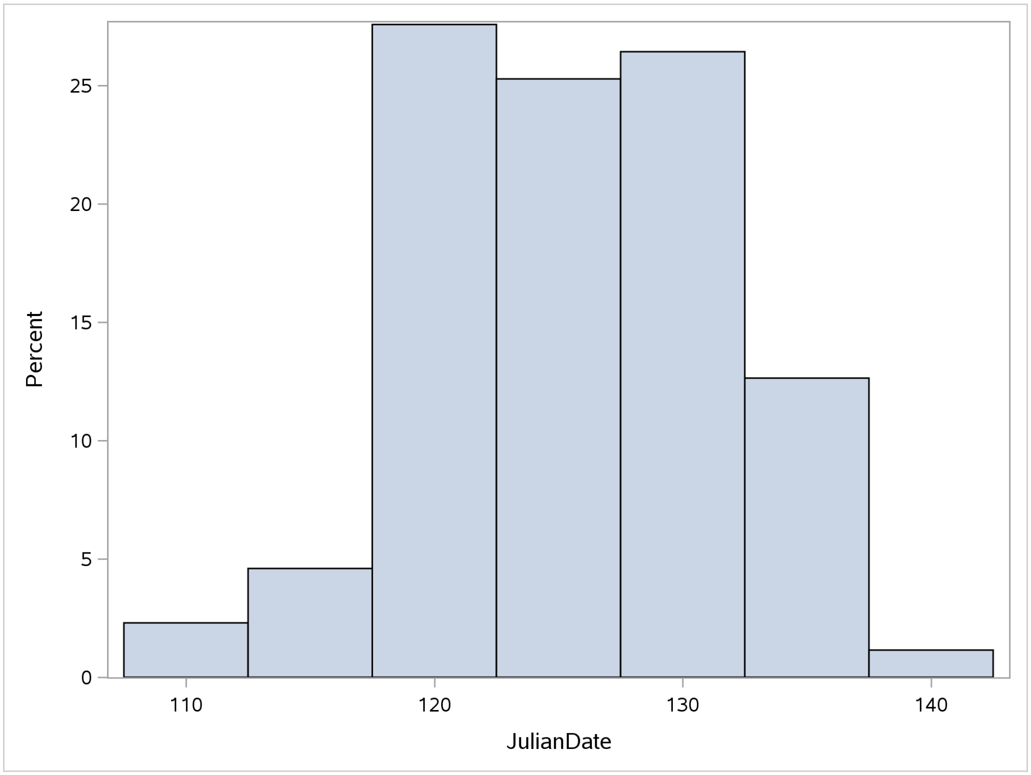
```
                              The MEANS Procedure

Variable       N            Mean         Std Dev         Minimum         Maximum
--------------------------------------------------------------------------------
Year          87         1960.00      25.2586619         1917.00         2003.00
JulianDate    87     125.5443126       5.9317755     110.7045000     141.4872000
--------------------------------------------------------------------------------
```

That looks good: the right years, and reasonable-looking Julian dates, a hundred and something days into the year.

(b) Make a histogram of the Julian dates.

Solution: Histogram of Julian dates:

```
proc sgplot;
  histogram JulianDate;
```

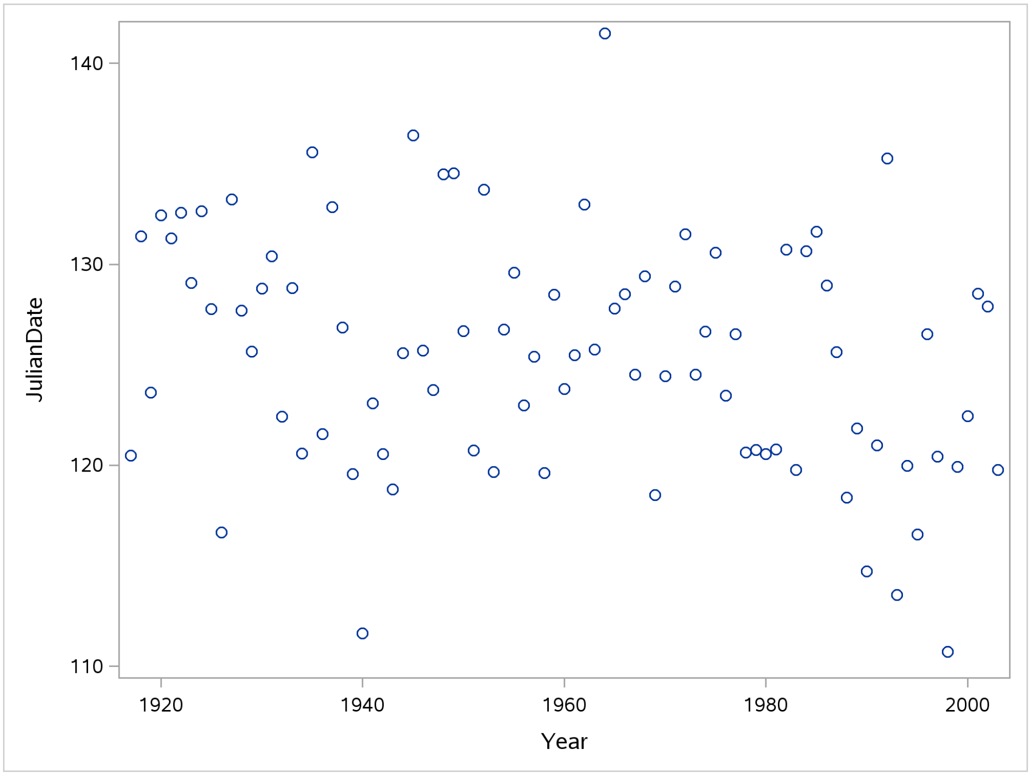


That looks more normal than my R histogram, because SAS used a different number of bins, seven rather than eight, and the choice of bins makes a difference to the look.

(c) To assess whether there is a trend of the Julian dates over time, plot the Julian dates against the year. I'll show you how to add a smooth trend.

Solution:

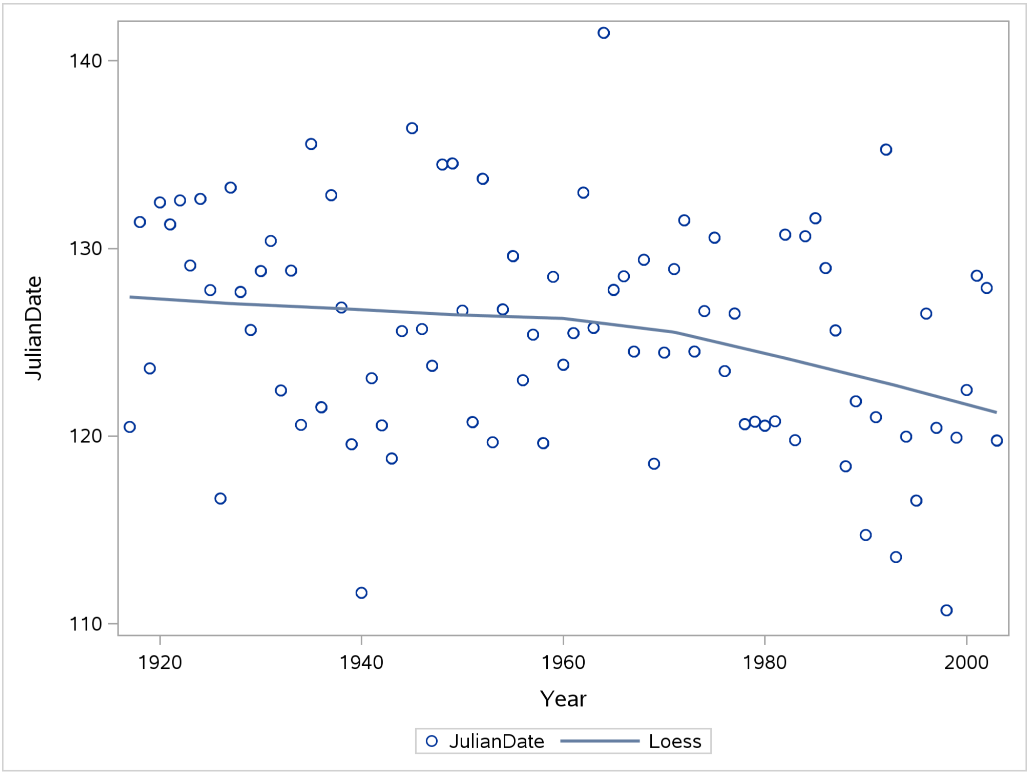
```
proc sgplot;
  scatter x=year y=JulianDate;
```



JulianDate
140
130
120
110
1920
1940
1960
1980
2000
Year

or (as an extra) put a smooth trend on it:

```
proc sgplot;
  scatter x=year y=JulianDate;
  loess x=year y=JulianDate;
```



The trend is smoother here (there is a parameter that controls the smoothness; evidently R and SAS have different defaults for it). Here, the smooth trend is slightly downhill and then, after 1960 or so, definitely downhill. (The R smooth trend was a bit down-and-up before 1960 but decidedly downhill after that.)

You'll recall that this pattern had a couple of implications:

1. the mean Julian date is not constant over time. This means (here) that the histogram is not actually *one* distribution but instead a mixture of a number of different distributions (and so it was kind of a lucky break that it looked normal in shape).
2. we have evidence that the ice is breaking up *earlier* every year, which is an indication of climate change (and, when we did this in R, I observed that this kind of thing is happening all over the Arctic, with implications for animal life, human habitation and all kinds of other things).

3.4. A used-car website lists several used Toyota Corollas for sale within a 250-mile radius of Redlands, California. For each car, its age (in years) and advertised price (in thousands of dollars) are recorded. The data are in `http://www.utsc.utoronto.ca/~butler/c32/corollas.txt`.

(a) Read the data into SAS and display the whole data set. (It is not too big, so displaying the whole thing is OK.)

Solution: The usual:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/corollas.txt";

proc import
  datafile=myurl
  out=corollas
  dbms=dlm
  replace;
  getnames=yes;
  delimiter=' ';

proc print;
```

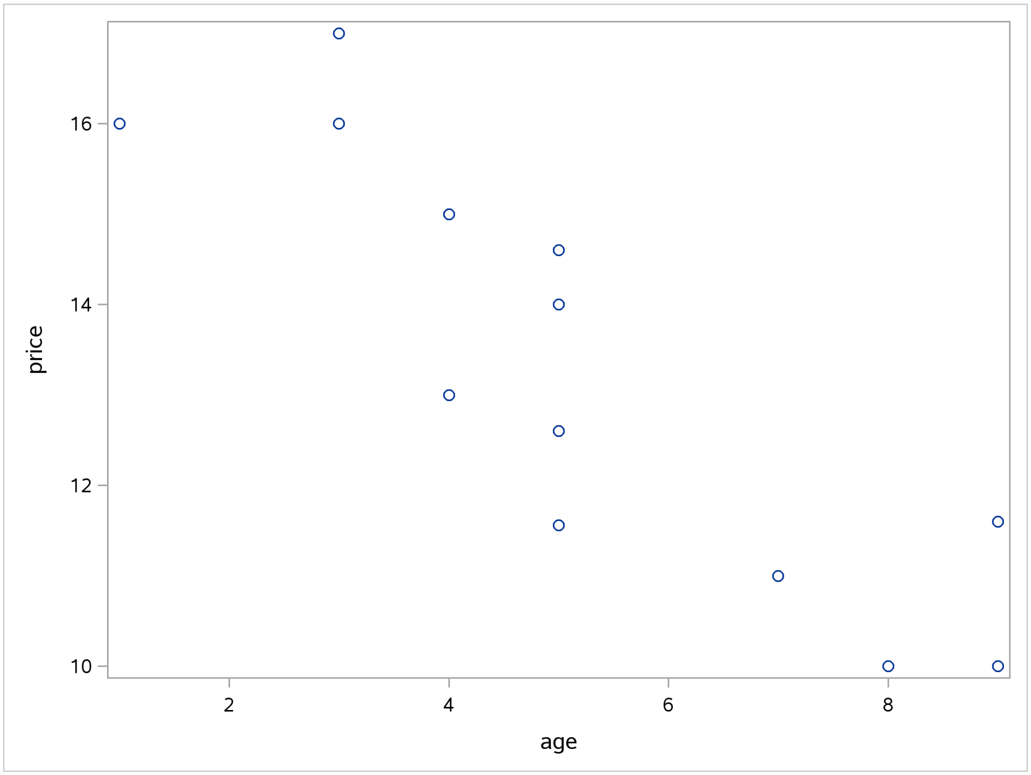
```
                    Obs              age           price

                      1                9            11.6
                      2                4              15
                      3                4              13
                      4                7              11
                      5                3              16
                      6                5            14.6
                      7                5           11.56
                      8                8              10
                      9                9              10
                     10                1              16
                     11                5            12.6
                     12                3              17
                     13                5              14
```

(b) Make a suitable graph of your two variables. Justify your choice of graph briefly.

Solution: The two variables `age` and `price` are both quantitative, so the right graph is a scatterplot. I think that the price is an outcome variable and age is explanatory, so `price` should be on the y-axis and `age` on the x. You should justify which variable is on which axis, or be able to say that it doesn't matter (if you can come up with a convincing argument for the latter, I'm good with it):

```
proc sgplot;
  scatter y=price x=age;
```

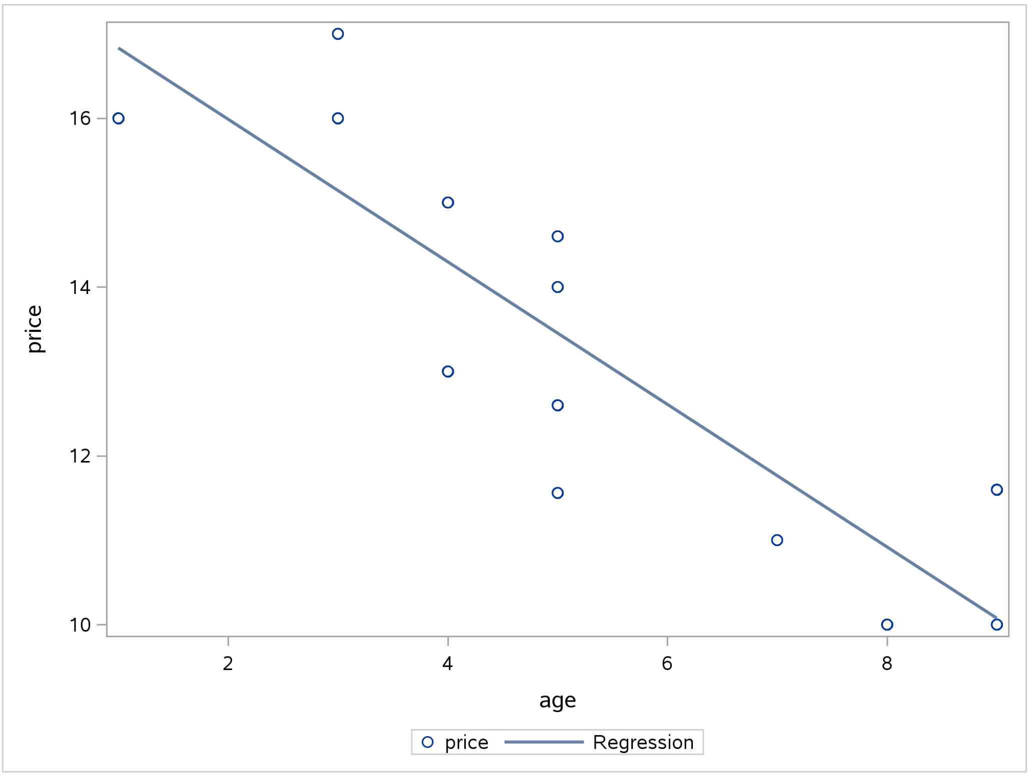
The `x=` and `y=` can be in either order. All that matters is that they are both there.



price
16
14
12
10
2
4
6
8
age

If you like (not obligatory, but it makes the next part easier), you can add a regression line to the plot, thus:

```
proc sgplot;
  scatter y=price x=age;
  reg y=price x=age;
```



(c) What does your plot tell you about any association between `age` and `price`? Does that correspond to what you know or can guess about the association between age and price of used cars? Explain briefly.

Solution: The scatterplots (especially my one with the regression line on it) point to a *downward* trend: that is to say, older cars tend to have a *lower* price. You would probably guess that an older car would have fewer years of use left, or would have been driven more kilometres, or would need a lot of repair, and so you would expect to pay less money for an older car. (Any one of those reasons is good.)

Note also that these cars are all the same model (Toyota Corollas), so there should be no effect of the data being a mixture of different models of car, which would weaken the trend. This is a decently strong trend.

(d) Find the mean and standard deviation of age and price. (It is enough to obtain output with these values on it.)

Solution: This is a simple application of `proc means`. You don't need to specify anything at all by way of variables, because these are all the quantitative variables in the data set:

```
proc means;
```

```
                                The MEANS Procedure

 Variable     N            Mean         Std Dev         Minimum         Maximum
 ------------------------------------------------------------------------------
 age         13       5.2307692       2.4205318       1.0000000       9.0000000
 price       13      13.2584615       2.3608821      10.0000000      17.0000000
 ------------------------------------------------------------------------------
```

There is no problem about specifying the names of the variables whose mean and SD you want, since the answer will be the same:

```
proc means;
  var age price;
```

```
                                The MEANS Procedure

 Variable     N            Mean         Std Dev         Minimum         Maximum
 ------------------------------------------------------------------------------
 age         13       5.2307692       2.4205318       1.0000000       9.0000000
 price       13      13.2584615       2.3608821      10.0000000      17.0000000
 ------------------------------------------------------------------------------
```

or even asking for the mean and SD by name:

```
proc means mean stddev;
  var age price;
```

```
                                The MEANS Procedure

                    Variable            Mean         Std Dev
                    ----------------------------------------
                    age            5.2307692       2.4205318
                    price         13.2584615       2.3608821
                    ----------------------------------------
```

Anything that gets the answers is good. I don't mind how you do it, but you may as well figure out how to do it with the smallest amount of work. In this case, that would mean figuring out that the defaults are what you need: that you don't need a `var` or a `class` for this one.

(e) Find the median and inter-quartile range of `price`. Again, obtaining output with the answers on it is good.

Solution: This is `proc means` again, but specifying the things to calculate on the first line, and this time you definitely do need to specify the variable to calculate them for:

```
proc means median Qrange;
  var price;
```

```
                              The MEANS Procedure

                          Analysis Variable : price

                                             Quartile
                              Median            Range
                         ----------------------------
                          13.0000000        3.4400000
                         ----------------------------
```

The median price is 13 (thousand dollars) and the inter-quartile range is 3.44 (thousand dollars).

This might seem like a largish spread. If you knew the `age` of a car, you could use regression to predict its selling price more accurately than this based on its age, because we saw earlier that older cars typically sell for less money (and therefore, knowing the age is valuable information if you want to say something about selling price). This is the kind of issue that R-squared in a regression gets into: the standard deviation (or the IQR) of price tells you that there is a largish amount of variation in the prices overall, but R-squared, which will also be fairly large, tells you that quite a lot of that variation is because we have a mixture of cars of different ages. Thus knowing the age of a car would allow you to predict its selling price with reasonable accuracy.

3.5. The "ecological footprint" of a person, city or country is defined by the World Wildlife Fund[9] as

> ...the impact of human activities measured in terms of the area of biologically productive land and water required to produce the goods consumed and to assimilate the wastes generated. More simply, it is the amount of the environment necessary to produce the goods and services necessary to support a particular lifestyle.

See `http://wwf.panda.org/knowledge_hub/teacher_resources/webfieldtrips/ecological_balance/eco_footprint/`. The units of an ecological footprint for a country is hectares per capita.

Data on 66 countries from the Americas, Europe and Asia (mainly the western part of Asia) are given in `http://www.utsc.utoronto.ca/~butler/c32/footprint.txt`. There are three columns: the name of the country, with spaces removed, the continent (called `Region`), with S standing for Asia, and the ecological footprint.

(a) (3 marks) Read the data into SAS and display the first 20 rows.

Solution: The first step is to take a look at the data: the values are separated by a single space, so "delimited" is the way to go. Remember to specify that the delimiting character is a space.

To display a certain number of rows of the data set, we need to specify the name of the data set and put `obs=20` in brackets afterwards:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/footprint.txt";
proc import
  datafile=myurl
  out=footprint
  dbms=dlm
  replace;
  getnames=yes;
```

```
    delimiter=" ";

  proc print data=footprint(obs=20);
```

```
                                                    Eco_
                   Obs    Country         Region      footprint

                     1    Argentina         A              2.5
                     2    Bolivia           A              2.1
                     3    Brazil            A              2.4
                     4    Chile             A                3
                     5    Colombia          A              1.8
                     6    CostaRica         A              2.3
                     7    Cuba              A              1.8
                     8    DominicanRep      A              1.5
                     9    Ecuador           A              2.2
                    10    ElSalvador        A              1.6
                    11    Guatemala         A              1.5
                    12    Haiti             A              0.5
                    13    Honduras          A              1.8
                    14    Jamaica           A              1.1
                    15    Mexico            A              3.4
                    16    Nicaragua         A                2
                    17    Panama            A              3.2
                    18    Paraguay          A              3.2
                    19    Peru              A              1.6
                    20    TrinidadToba      A              2.1
```

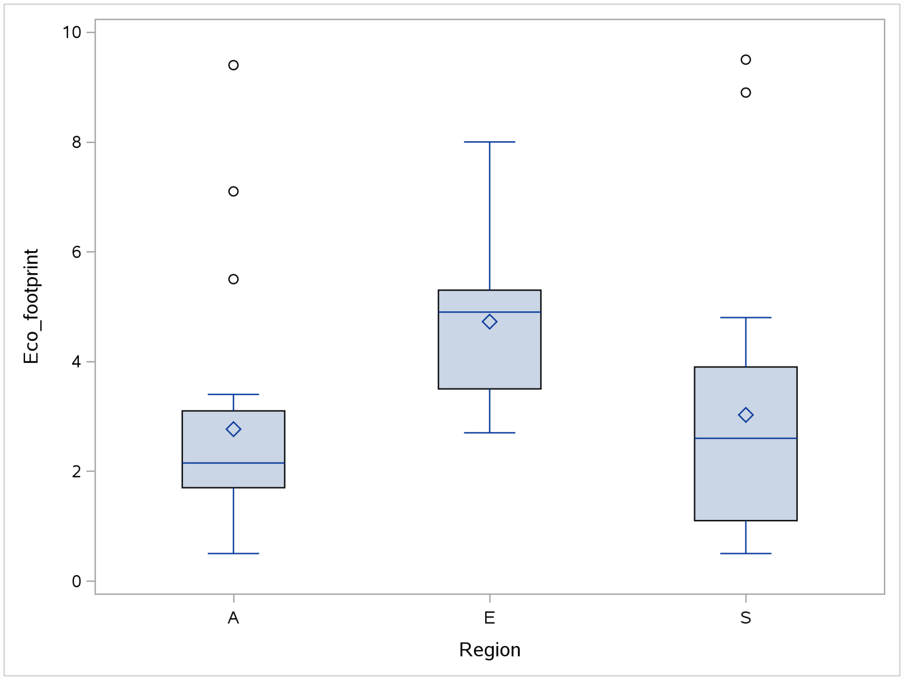
To display only some rows, you need to use the data set name (the one on `out`; whatever you choose is fine, but it has to be the same in both places) and `obs=20` in brackets afterwards.

Normally you only need to say `proc print` and SAS displays (all of) the most recently-created data set, but the `obs=20` is an "option" that is attached to a data set, so you have to give the data set name to attach it to.

(b) (2 marks) Make a suitable graph of the ecological footprint values for each Region. Note that in SAS, variable names are *not* case-sensitive.

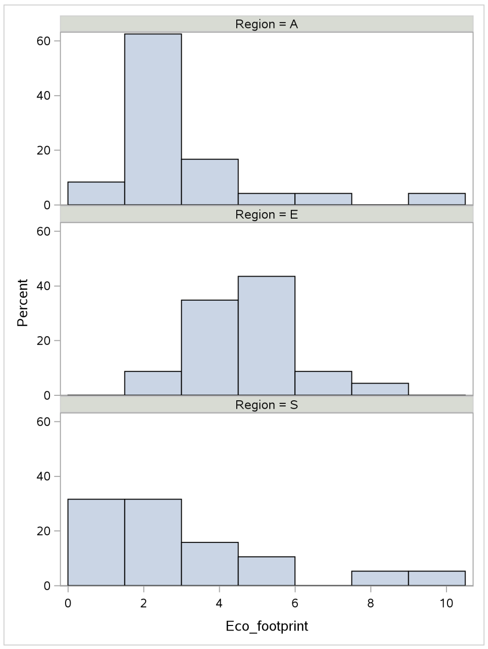
Solution: With a quantitative variable (`Eco_footprint`) and a categorical one `Region`, this suggests a boxplot, which I think is best. I am using lowercase for my variable names, for ease of typing.[10]

```
  proc sgplot;
    vbox eco_footprint / category = region;
```


10
8
6
4
2
0
Eco_footprint
A
E
S
Region

If you *must*, you can do a histogram for each region, but since we'll be wanting to compare the histograms later, a panelled plot of histograms is the way to go:

```
proc sgpanel;
  panelby region / columns=1;
  histogram eco_footprint;
```



I made one column of histograms (using the `columns=1`) so that the graphs would be vertically above each other on the same scale, which makes it easier to compare them with each other (coming later).

There really *isn't* a good way of making separate histograms for the three different regions, since that involves making separate data sets, one for each region, and we don't know how to do that yet.

(c) (2 marks) Find the mean and median ecological footprint by region.

Solution: This is a custom `proc means`:

```
proc means mean median;
  var eco_footprint;
  class region;
```

The `var` and the `class` can be either way around. In fact, you can omit the `var` because `proc means` will only do the calculations for the quantitative variables in the data set, and this is the only one.

The output:

```
                                The MEANS Procedure

                         Analysis Variable : Eco_footprint

                    N
          Region  Obs          Mean        Median
          ------------------------------------------
          A        24     2.7666667     2.1500000

          E        23     4.7260870     4.9000000

          S        19     3.0263158     2.6000000
          ------------------------------------------
```

To verify my assertion from above:

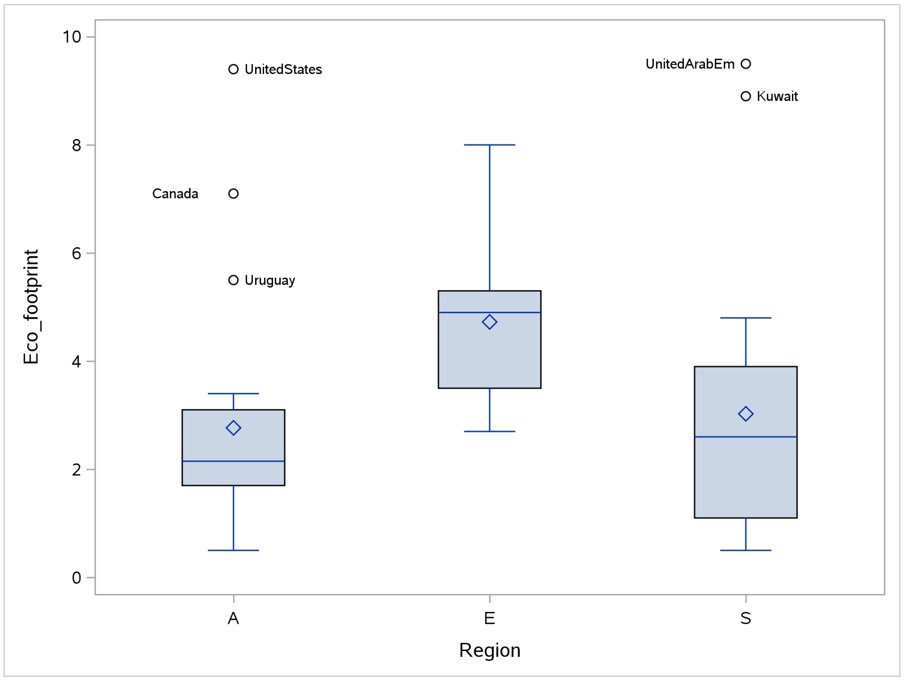
```
proc means mean median;
  class region;
```

```
                           The MEANS Procedure

                N
Region        Obs    Variable                   Mean          Median
--------------------------------------------------------------------
A              24    Eco_footprint         2.7666667       2.1500000
                     __ClsFmtIdx1__        1.0000000       1.0000000

E              23    Eco_footprint         4.7260870       4.9000000
                     __ClsFmtIdx1__        2.0000000       2.0000000

S              19    Eco_footprint         3.0263158       2.6000000
                     __ClsFmtIdx1__        3.0000000       3.0000000
--------------------------------------------------------------------
```

Well, kinda. I think that second variable with all the underscores in its name is a "hidden" quantitative variable that identifies the regions, but I'm not sure. If you get this, it will probably scare you into putting the `var` line in so that you get exactly what you wanted.

Extra: you might be curious about *which* countries those outliers are. Later, we'll be learning how to label observations on a plot, most commonly a scatterplot. The technique is called `datalabel`. I discovered that `datalabel` also works on boxplots, and it only labels the outliers, which (as here) is usually exactly what you want:

```
proc sgplot;
  vbox eco_footprint / category = region datalabel=country;
```



The outliers are not terribly surprising: Canada and the US in the Americas, and two oil-rich nations in Asia. I'm not sure, however, why Uruguay is so high. Many of the other high countries are small in area, so maybe that's what happened here.

(d) (3 marks) Explain briefly how your graph and your calculations are consistent. (There are a number of different approaches you can take; anything that offers insight into how the graph and the calculations are telling the same story is good.)

Solution: I think there are these general approaches you can take:

1. estimate the mean and median from the boxplots and show that you get the same thing as the calculation
2. compare the mean and median for each group and show that the boxplot and calculations agree in terms of which is bigger
3. say something about *why* you would expect the mean to be bigger than the median (or not) in the cases where it is (or is not).

4. compare the regions with each other and note that the biggest medians (or means) agree.

There are probably more ways that I didn't think of. If you came up with something that in the grader's opinion shows "sufficient insight" into how the graphs and calculations are consistent, I'm good with that. Likewise, if you drew histograms rather than boxplots, what you'll be able to do by way of comparison is different.

All right, down to business. Numbering the approaches as I did above, we have:

1. Looking at the boxplots: for the Americas, the median is just over 2 and the mean is maybe a bit less than 3. (2.15 and 2.77 are the exact answers.) For Europe, median is about 5 and mean is a little less (exact answers 4.9 and 4.72). Finally, for Asia, the mean is about 3 and the median is something like 2.5 (3.03 and 2.6). As you see, you don't need to be very accurate reading off the plot; anything that gives the general idea is good.

2. The boxplot says that the mean is noticeably higher than the median for the Americas and for Asia, but for Europe, the mean and median are much closer (with the median being slightly bigger). From `proc means`, the Americas have mean 2.77 and median 2.15; Asia has mean 3.03 and median 2.6; Europe has median 4.9 and mean 4.72, which all agree with what we just said about how the mean and median compare.

3. It seems a pretty good guess that the mean being bigger than the median is driven by the outliers at the high end. The Americas have 3 upper outliers pulling the mean up (but not the median), and Asia has two very high outliers likewise pulling the mean up. Europe has no outliers, so from this point of view the mean is not being pulled anywhere. On the other hand, Europe has a long upper whisker, indicating a right skew, so you would expect the mean to be bigger than the median, though it is actually a bit less. The likely explanation of *that* is something to do with the shape of the distribution, which boxplots don't give you much detail on.

   If you drew histograms rather than boxplots, this is probably the best way to tackle this question. The Americas and Asia histograms are very clearly skewed to the right: the tail is basically *all* to one side. The Europe histogram is a lot more symmetric (or only slightly right-skewed), so there isn't much chance for the mean to be a lot different from the median. Another way to look at the Europe histogram is to say that it is *short*-tailed, so that the median will be somewhere in the third bar and the mean might be a bit less because of there being a lot of values in the second bar.

4. Comparing the regions with each other: Europe has the highest mean and median. Comparing the Americas with Asia, I'd say the means are about the same but the median for Asia is a little bigger. These check out with the `proc means` output; in fact, the mean for Asia is slightly bigger than that for the Americas.

   This also works with histograms: Europe is the farthest to the right (highest mean/median), and the Americas and Asia are similar with the mean higher than the median. (It's hard to judge more than that from the histograms.)

In all of these, *make a judgment and defend it*; you don't have to get the same comparisons between things that I did, and that's fine.[11] For example, if you took the fourth approach and said that both the mean and median for Asia are a little bigger than for the Americas, that's fine.

Last thing: *do not* refer to the regions by their letters; go back to the description in the question and find the *real names* of the regions (continents). The single-letter names for the regions are a convenience for the person who entered the data,[12] but when you're interpreting the results for the benefit of someone else, you are *not* entitled to make them work to figure out what you mean. That's your job.

4 Basic inference

4.1. Recall that we previously investigated the North Carolina births in SAS. Here we revisit that data set, which was at `http://www.utsc.utoronto.ca/~butler/c32/ncbirths.csv`.

(a) Read the data set into SAS again.

Solution: Since you've done it before, you can do it again. If you're on SAS Studio online, you'll need the first line (or you'll have to grapple with variable names containing spaces again):

```
options validvarname=v7;

filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/ncbirths.csv";

proc import
  datafile=myurl
  dbms=csv
  out=ncbirths
  replace;
  getnames=yes;
```

(b) Taking all the babies together, obtain a 95% confidence interval for the mean birth weight of all babies. Compare SAS's answer with R's from when you did this before. You will need to be careful to get the right name for this variable.

Solution: In this variable, both the brackets and the space have been replaced by underscores, so there are *two* underscores between the words `Weight` and `pounds`, and one after:

`proc ttest`:

```
proc ttest;
  var Weight__pounds_;
```

```
                 N        Mean     Std Dev     Std Err     Minimum     Maximum

               500      7.0688      1.5062      0.0674      1.1875     11.6250
                   Mean       95% CL Mean        Std Dev      95% CL Std Dev

                 7.0688      6.9364   7.2011      1.5062      1.4183   1.6058
                                DF    t Value    Pr > |t|

                               499     104.94      <.0001
```

6.94 to 7.20 pounds. (Don't take the CI for the standard deviation by mistake!) This is the same as R's.

(c) Likewise taking all the babies together, test the null hypothesis that the mean birthweight is 7.3

pounds against the alternative that it is less. Obtain a P-value, and see whether it is the same as R's from before. (You did the actual test before, so I don't need the conclusion in context this time.)

Solution: Specifying a null hypothesis mean and a directional alternative in SAS, L for "less":

```
proc ttest h0=7.3 sides=L;
  var Weight__pounds_;
```

```
           N        Mean     Std Dev      Std Err     Minimum     Maximum

         500      7.0688      1.5062       0.0674      1.1875     11.6250
             Mean        95% CL Mean        Std Dev      95% CL Std Dev

           7.0688    -Infty      7.1798      1.5062      1.4183   1.6058
                              DF    t Value    Pr < t

                             499      -3.43    0.0003
```

The alternative is that the mean is *lower* than 7.3. The P-value is 0.0003, which is the same (to the accuracy shown) as R's.

If you couldn't remember the `sides` thing, you can do three steps: get a two-sided test (the P-value comes out as 0.0006), check to see that you are on the correct side of the null (the sample mean is 7.07, which *is* less than 7.3), then halve the two-sided P-value (getting 0.0003). But you need all of this. If you have it, you're good.

4.2. Nenana, Alaska, is about 50 miles west of Fairbanks. Every spring, there is a contest in Nenana, called the Ice Classic. A wooden tripod is placed on the frozen river, and people try to guess the exact minute when the ice melts enough for the tripod to fall through the ice. The contest started in 1917 as an amusement for railway workers, and has taken place every year since. Now, hundreds of thousands of people enter their guesses on the Internet and the prize for the winner can be as much as $300,000. The contest has a website: `http://www.nenanaakiceclassic.com/`. On the website is a webcam that shows the latest picture of the river, so that during the contest you can see how close you are to winning (or losing).

Because so much money is at stake, and because the exact same tripod is placed at the exact same spot on the ice every year, the data are consistent and accurate. The data are in `http://www.utsc.utoronto.ca/~butler/c32/nenana.txt`, separated by tabs (since the dates have spaces in them).

Yes, we saw these data before.

(a) Read in the data, and find the means for all the variables. You probably did this before; if you did, you can re-use your code.

Solution: This is the same as last time on this data set. Use the same code as you did before, once you got it working.

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/nenana.txt";

proc import
  datafile=myurl
  dbms=dlm
  out=mydata
  replace;
  delimiter='09'x;
  getnames=yes;

proc means;
```

```
                              The MEANS Procedure

Variable       N            Mean         Std Dev         Minimum         Maximum
--------------------------------------------------------------------------------
Year          87         1960.00      25.2586619         1917.00         2003.00
JulianDate    87     125.5443126       5.9317755     110.7045000     141.4872000
--------------------------------------------------------------------------------
```

(b) Obtain a 90% confidence interval for the mean Julian date.

Solution: The Julian dates are numbers, so it makes perfect sense to calculate the mean (and to find a confidence interval for the population mean that it is an estimate for).

A confidence interval at a non-standard confidence level requires you to note that SAS uses `alpha` as would be appropriate for a test, so a 90% CI needs $\alpha = 0.10$:

```
proc ttest alpha=0.10;
  var JulianDate;
```

```
           N        Mean     Std Dev     Std Err     Minimum     Maximum

          87       125.5      5.9318      0.6360       110.7       141.5
              Mean       90% CL Mean        Std Dev      90% CL Std Dev

             125.5       124.5    126.6      5.9318      5.2774   6.7906
                               DF    t Value    Pr > |t|

                               86     197.41      <.0001
```

124.5 to 126.6, same as R before. Ignore all the rest of the output.

(c) Test whether the mean Julian date is 130 or less than 130. (Remember the bit about the old-timer and his grey beard from before?)

Solution: Test that the mean Julian date is 130, against the alternative that it is less. This needs to be actually done, since a confidence interval is two-sided and this is one-sided (and therefore you can't just use the result of the previous part, at least not without thinking).

```
proc ttest h0=130 sides=L;
  var JulianDate;
```

```
           N        Mean     Std Dev     Std Err     Minimum     Maximum

          87       125.5      5.9318      0.6360       110.7       141.5
              Mean       95% CL Mean        Std Dev      95% CL Std Dev

             125.5    -Infty      126.6      5.9318      5.1624   6.9727
```

```
                 DF    t Value    Pr < t

                 86      -7.01    <.0001
```

Either `side` or `sides` will work. Use whichever you like.

Consistent P-value with R.

You cannot do this just from the confidence interval output from the previous question, because we never specified a null mean there, and so in fact the P-value comes from a test of the population mean being *zero* (the default). Precisely, all the confidence interval tells us about the P-value for a test of $\mu = 130$ vs. $\mu < 130$ is that it is less than half of 0.10.

Not also that 130 is outside the confidence interval, so we'd expect a very small P-value (even though the test is one-sided, so strictly we'd expect a *two-sided* test to have a very small P-value, and therefore a one-sided test to have an *even smaller* P-value, given that the data is on the "right side" of 130).

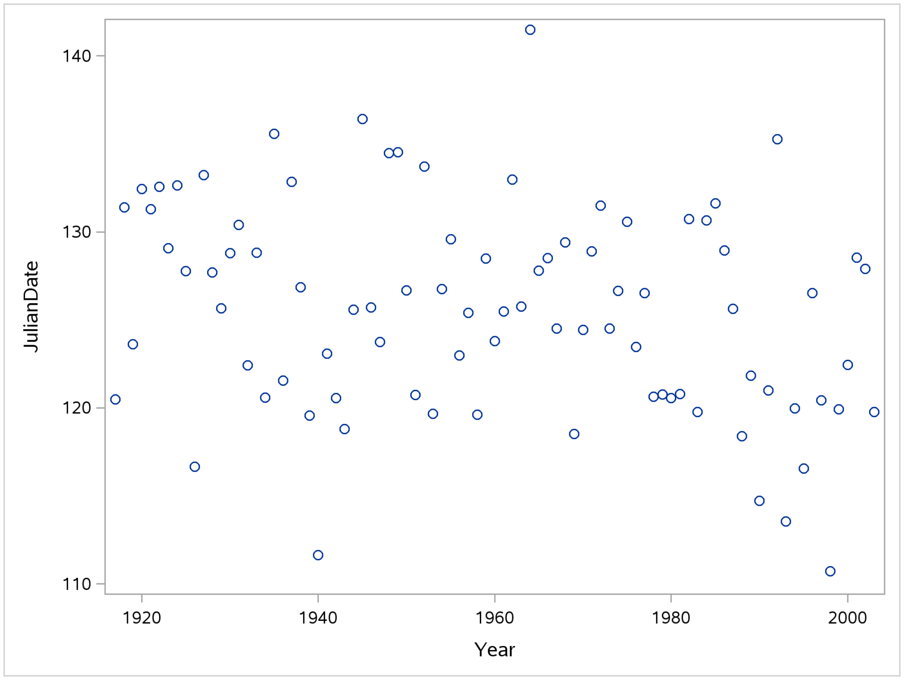
We have the same issues as before about the mean Julian date not being constant, so it doesn't make all that much sense to do inference for it. But that's by the way. Our aim here was to figure out how to do a test and confidence interval.

(d) If you feel up for some exploration, follow through this part, about making a plot of Julian date against year.

Solution: I want to see whether the typical date on which the tripod falls through the ice is changing over time. The obvious thing with two quantitative variables is a scatterplot:

```
proc sgplot;
  scatter x=year y=juliandate;
```

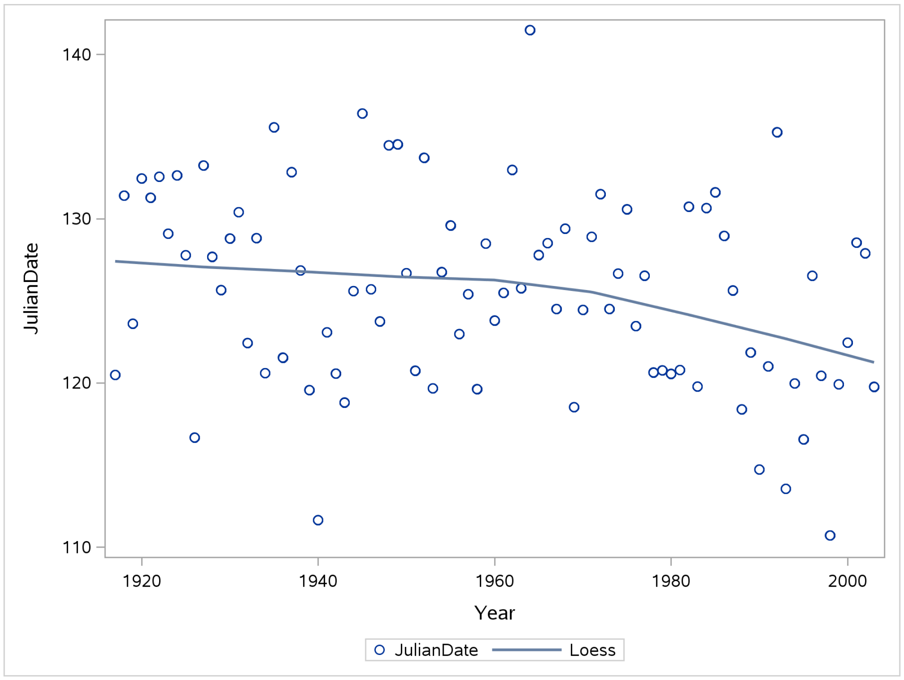
SAS graphs (for me) seem to insist on a large amount of empty space around them:



JulianDate
Year
140
130
120
110
1920
1940
1960
1980
2000

That looks very random, but it is natural data with (as usual for natural data) a lot of variability. A smooth trend would be nice. Down near the end of the course I talk about `loess`, which is the SAS version of `geom_smooth`, so we'll borrow that to use now:

```
proc sgplot;
  scatter x=year y=juliandate;
  loess x=year y=juliandate;
```



There is a small but noticeable downward trend, more pronounced since about 1970. Environmental science people see a lot of graphs with this kind of shape. What this one means is that the ice is melting *earlier* on average every year, and that the trend has been faster since about 1970.

A way to test whether this is real or just chance is to fit a regression line, test the slope for significance and look at its size. We haven't done this in SAS yet, but to give you a preview:

```
proc reg;
  model juliandate=year;
```

```
                            The REG Procedure
                              Model: MODEL1
                       Dependent Variable: JulianDate

                   Number of Observations Read          87
                   Number of Observations Used          87
                            Analysis of Variance

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

Model                     1      238.06059      238.06059       7.26    0.0085
Error                    85     2787.93204       32.79920
Corrected Total          86     3025.99262
             Root MSE                5.72706    R-Square     0.0787
             Dependent Mean        125.54431    Adj R-Sq     0.0678
             Coeff Var               4.56178
                            Parameter Estimates

                         Parameter       Standard
    Variable      DF      Estimate          Error    t Value    Pr > |t|

    Intercept      1     254.64848       47.92518       5.31      <.0001
    Year           1      -0.06587        0.02445      -2.69      0.0085
```

The slope is significantly nonzero (P-value 0.0085) and negative, so that downward trend is real. As for its size, it's about 0.065 days per year, which doesn't seem like much, but if we scale that up to the 80 or so years the Ice Classic has been running:

```
-0.06587*80

## [1] -5.2696
```

Five and a bit days out of the hundred-and-something that the Julian dates typically are.

4.3. A professor collected some information about a random sample of 22 of his students. Specifically, the information collected was:

- height (in inches)
- weight (in pounds)
- birthday (the number of the month it's in)

- the result of a coin toss by that student (1 is heads, 0 is tails)
- the gender of the student. These are recorded as 0 and 1, but unfortunately we don't know which one is male and which one is female.
- Two measurements of the student's pulse rate, taken a few minutes apart.

The data are in `http://www.utsc.utoronto.ca/~butler/c32/students.txt`. Use SAS for this question.

(a) Read in the data from the file. (The data values are all numbers.) Display the 22 rows of data.

Solution:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/students.txt";
proc import
  datafile=myurl
  dbms=dlm
  out=students
  replace;
  delimiter=' ';
  getnames=yes;


proc print;
```

```
        Obs          HEIGHT          WEIGHT        BIRTHDAY            COIN

          1              65             135               4               1
          2              63             119               9               1
          3              72             175              11               0
          4              60             106               9               1
          5              65             135               8               0
          6              72             170              10               1
          7              64             180               8               1
          8              71             205              10               1
          9              75             195               6               0
         10              71             185               8               1
         11              71             182               6               0
         12              65             108               8               0
         13              73             150               4               1
         14              67             128               6               0
         15              74             175               6               1
         16              66             160               9               1
         17              65             143               9               0
         18              72             190              11               0
         19              64             180               2               1
         20              61             195               5               0
         21              72             220               7               1
         22              69             235               7               1

        Obs             SEX          PULSE1          PULSE2

          1               0              73              73
          2               0              62              70
          3               1              72              73
          4               0              73              73
          5               0              75              74
          6               1              69              69
          7               0              68              73
          8               1              72              73
          9               1              68              67
         10               1              68              73
         11               1              70              74
         12               0              73              75
         13               1              70              70
         14               0              74              80
         15               1              68              69
         16               0              74              77
         17               0              70              72
         18               1              68              69
         19               1              68              68
         20               1              90              95
         21               1              75              78
         22               1              74              72
```

22 rows of sensible-looking numbers. I think I am good, even though they are running off the bottom of the page.

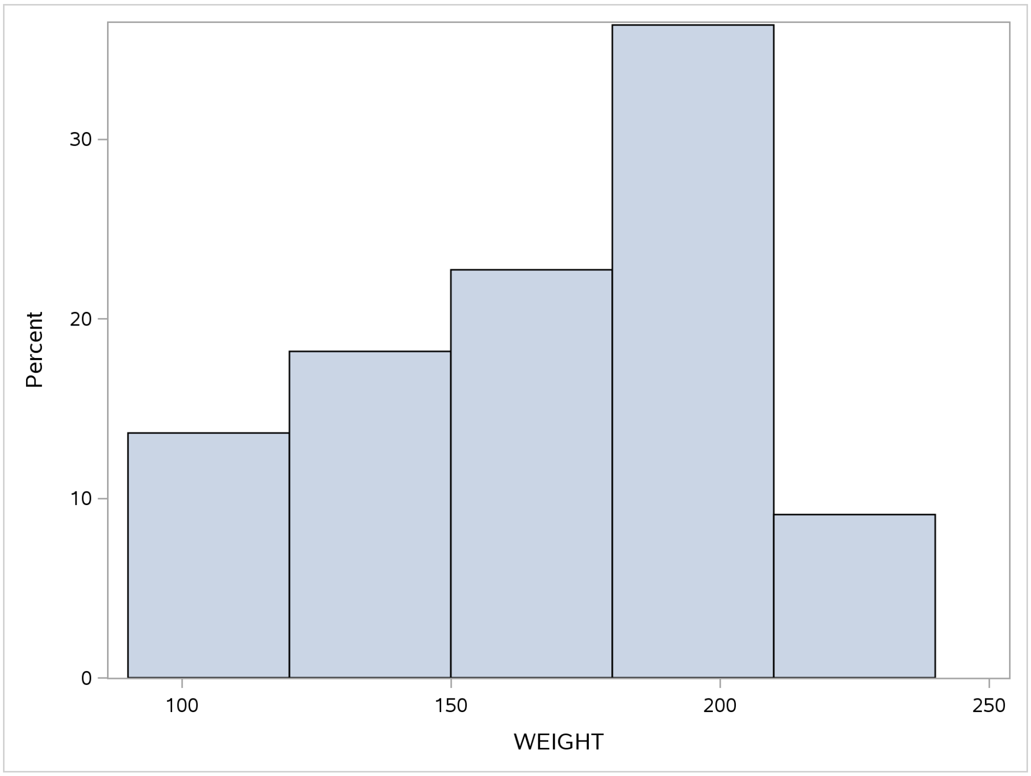
With only 22 rows, it's OK to display them all.

The variable names are in ALL CAPITALS, but SAS is not case-sensitive, so you can use things like `weight` the rest of the way. I'm going to do that, because I'm lazy.

(b) Obtain a histogram of the weights. How would you describe its shape? Comment briefly.

Solution: `proc sgplot`:

```
proc sgplot;
  histogram weight;
```



This is a little bit skewed to the left.

Sturges' rule would say six bins rather then five ($n = 22$ and $2^5 = 32$). SAS doesn't say what its default number of bins is.[13]

(c) Obtain a 99% confidence interval for the mean weight (of all students of whom these are a sample).

Solution: This requires a bit of thinking to get the confidence level right. In SAS, you specify `alpha` (that you would use for a test), which is one minus the confidence level, so:

```
proc ttest alpha=0.01;
```

```
  var weight;
```

```
          N        Mean     Std Dev     Std Err     Minimum     Maximum

         22       166.9     35.1775      7.4999       106.0       235.0
             Mean       99% CL Mean        Std Dev      99% CL Std Dev

            166.9       145.6    188.1     35.1775     25.0535  56.8746
                                 DF    t Value    Pr > |t|

                                 21      22.25      <.0001
```

The output confirms that we have indeed got a 99% interval for the mean, which goes from 145.6 to 188.1 pounds.

(d) Is it reasonable to believe that these students came from a population with mean 140 pounds, or is the mean bigger than that? Carry out a suitable test, and explain briefly what you conclude.

Solution: It's tempting to look at the CI and say "140 is not in there, so the mean is bigger than 140". That isn't quite right, though, because a confidence interval is a two-sided thing, and this test is one-sided (the alternative is "bigger than 140"). So we had better do the test.

The major concern is how to specify the null and alternative hypotheses. There are some choices about how to specify the null value 140: `mu0` or `location` or `h0`. I think they all work, though the documentation says the last one. The alternative is expressed by saying `side=` along with "2" (two-sided, the default), 'L" for lower or "U" for "upper". It is my habit to use uppercase for these, so that "lower" doesn't get confused with "1", whatever that means.[14] Here, the alternative is "bigger", so we want "U". `sides` instead of `side` also works:

```
proc ttest h0=140 sides=U;
  var weight;
```

```
          N        Mean     Std Dev     Std Err     Minimum     Maximum

         22       166.9     35.1775      7.4999       106.0       235.0
             Mean       95% CL Mean        Std Dev      95% CL Std Dev

            166.9       154.0  Infty       35.1775     27.0639  50.2709
                                 DF    t Value    Pr > t

                                 21       3.58    0.0009
```

The P-value is 0.0009, a lot smaller than 0.05, so we reject the null mean of 140 and conclude that the mean weight of students in the population from which these students were drawn is indeed greater than 140 pounds.

(e) Explain briefly why this t-test should be reasonably trustworthy.

Solution: The (relevant) assumption behind the t-test is that the data come from a normal distribution. Our histogram suggested that the weight values are somewhat skewed (not badly skewed, though). This is OK for two reasons (if you get one of them, that's OK): (i) the t-tests are robust to non-normality, so that even if the data are somewhat non-normal, as here, the P-value is still reasonable, or (ii) the sample is on the small side (22), so that the data can *look* a bit non-normal even if they were actually drawn from a normal distribution.

You might have noticed that these students are males and females mixed together, and so you would expect to see two different distributions of weights mixed together. This might be the case in fact, but our sample is too small to be sure about that.

(f) Can you determine from the data whether `SEX` 0 is males and 1 females, or whether it's the other way around? Obtain some numbers or graphs to help you decide, and explain briefly what you conclude.

Solution: The key here is to find something in the data set that is associated with gender (based on what you know or can guess). I don't mind so much what, as long as it is reasonable to believe that it would be associated with gender.

My best guess is that males tend to be "bigger" than females: that is, taller *and* heavier. So height and weight should both be bigger for males. So let's summarize height and weight by gender, and see what we get. This could be a numerical summary, such as comes out of `proc means`, or it could be a graphical one like a boxplot (boxplots are the best for comparing distributions).

Thus, one of these:

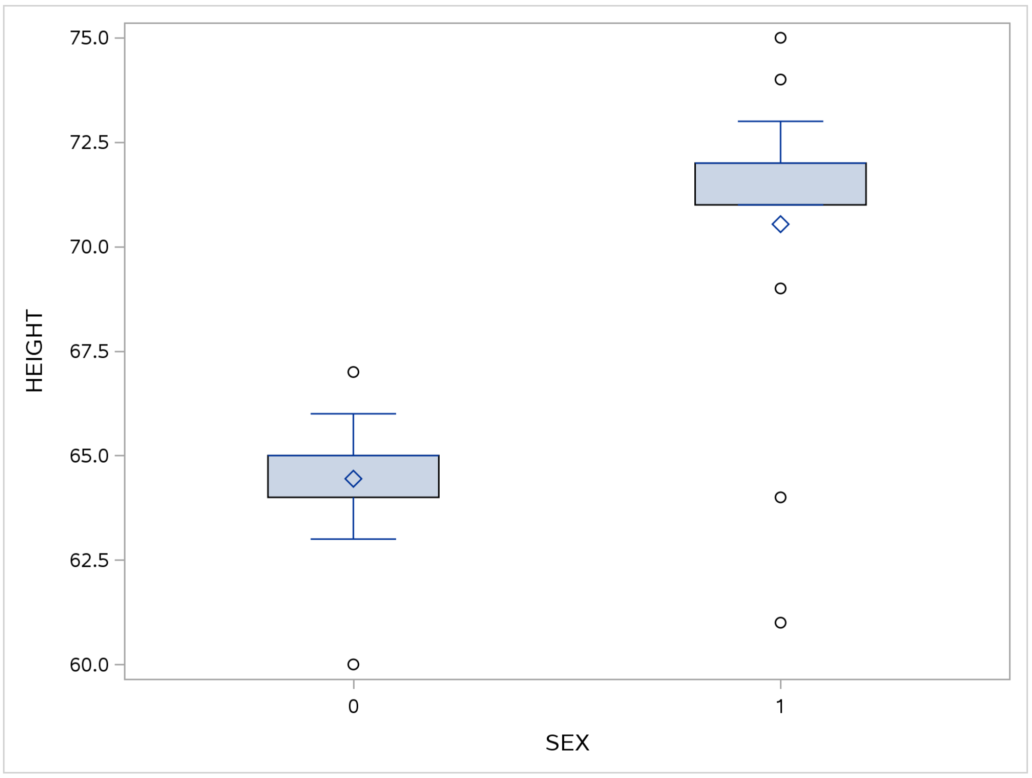
```
proc means;
  var height weight;
  class sex;
```

(you can also do two separate `proc means`)

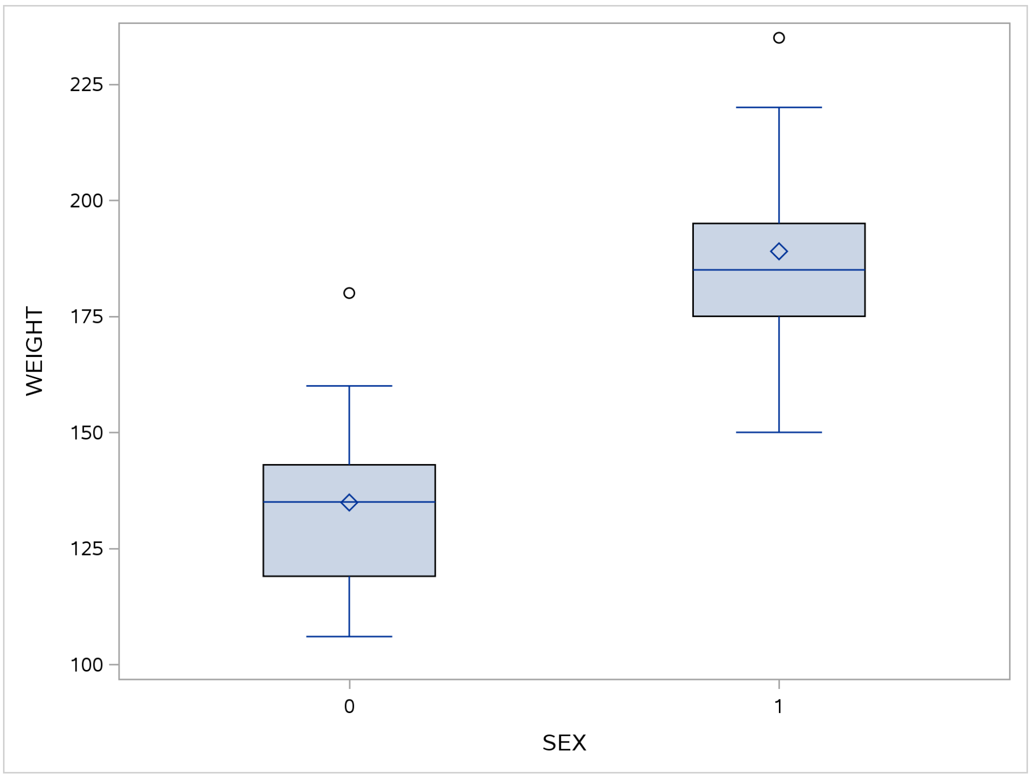
```
                              The MEANS Procedure

             N
     SEX   Obs   Variable    N           Mean        Std Dev        Minimum
------------------------------------------------------------------------------
       0     9   HEIGHT      9     64.4444444      2.0069324     60.0000000
                 WEIGHT      9    134.8888889     23.9501798    106.0000000

       1    13   HEIGHT     13     70.5384615      3.9075863     61.0000000
                 WEIGHT     13    189.0000000     22.0340645    150.0000000
------------------------------------------------------------------------------


                                   N
                        SEX      Obs    Variable        Maximum
                  ------------------------------------------------
                          0        9    HEIGHT       67.0000000
                                        WEIGHT      180.0000000

                          1       13    HEIGHT       75.0000000
                                        WEIGHT      235.0000000
                  ------------------------------------------------
```

or (this one needs a plot each for height and weight if you do both):

```
proc sgplot;
  vbox height / category=sex;
```



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



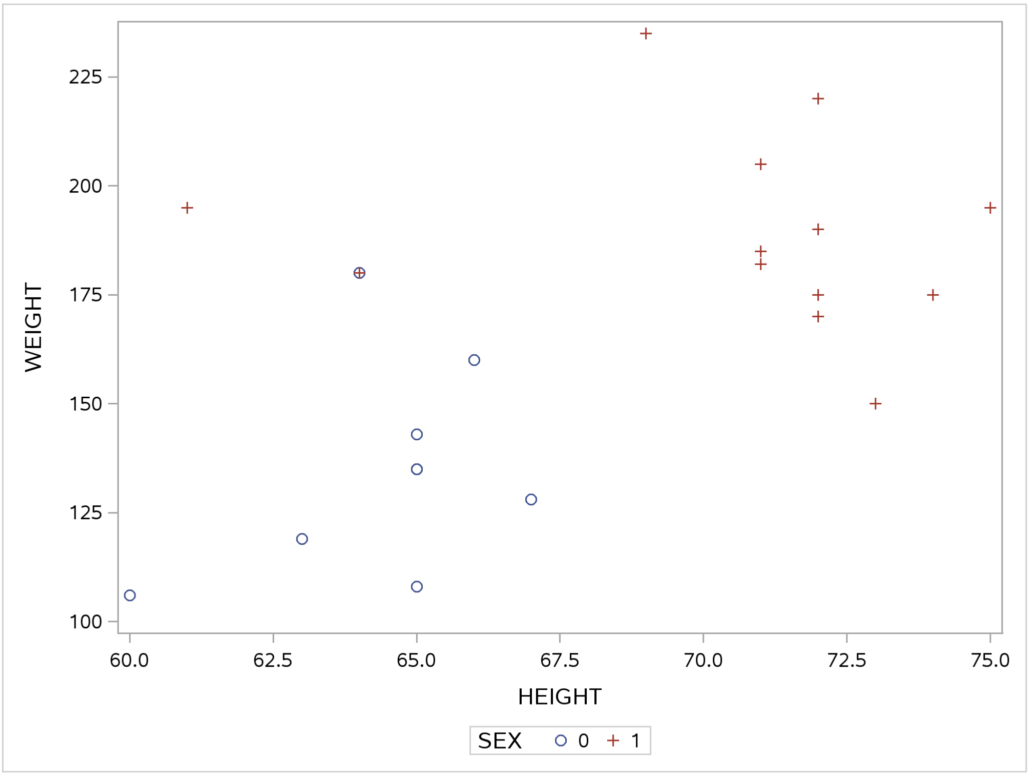
Whichever way you do it (and the minimum is to pick one variable, compare it somehow between genders, and then make a call), gender 1 is both taller on average and heavier on average than gender 0. So 1 is males and 0 is females.

I had an idea about getting both height and weight into one graph: plot height against weight as a *scatterplot*, and label the points differently according to whether they are gender 0 or gender 1. This is the idea: `proc sgplot` with `scatter` *and* `group=`:

```
proc sgplot;
  scatter x=height y=weight / group=sex;
```

This is, if you were keeping track, two quantitative variables and one categorical one.

There's no reason why the x and y-axes should be this way around; they could just as well be the other way around, since neither variable is a response to the other.



Gender 1, evidently the males, is up and to the right (taller *and* heavier), and gender 0 is down and to the left on the plot, evidently the females. There are a couple of males top left (short but heavier), but overall, the distinction is pretty clear.

4.4. Previously, we investigated whether children (aged 8–17) spend more time on electronic devices now than they did 10 years ago. Samples of 15 children aged 8–17 were taken in each of two years, 1999 and 2009, and the children (with their parents' help) were asked to keep a diary of the number of hours they spent using electronic devices on a certain day. The data are in the file `http://www.utsc.utoronto.ca/~butler/c32/pluggedin.txt`.

(a) Why are these data two independent samples rather than matched pairs? (Think about the way the data were collected.)

Solution: Children that appeared in the 1999 sample would have been *too old* to be in the 2009 sample. So it must have been a different group of children in 2009 than it was in 1999. Thus, this is two independent samples (and a two-sample *t*-test is coming up).

For this to have been matched pairs, we would have had to have the *same* 15 children assessed both times, or at least we would have had to have some natural pairing-up. (Even having siblings of the 1999 children be the sample for 2009 would have been difficult to arrange.)

The fact that there were 15 children in each group was meant to confuse you a little: if they were matched pairs, there would *have to be* the same number of children both times, but with two independent samples, there might be the same number of children or there might not be.[15]

(b) Read the data into SAS, and list out the values. Make sure you have 30 values altogether, and two different years.

Solution: Look at the file to see that the data values are separated by spaces (the clue is in the file extension `.txt`), and then use the version of `proc import` that reads space-delimited files. Copy an old one. That's what I do:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/pluggedin.txt";

proc import
  datafile=myurl
  dbms=dlm
  out=pluggedin
  replace;
  delimiter=' ';
  getnames=yes;


proc print;
```

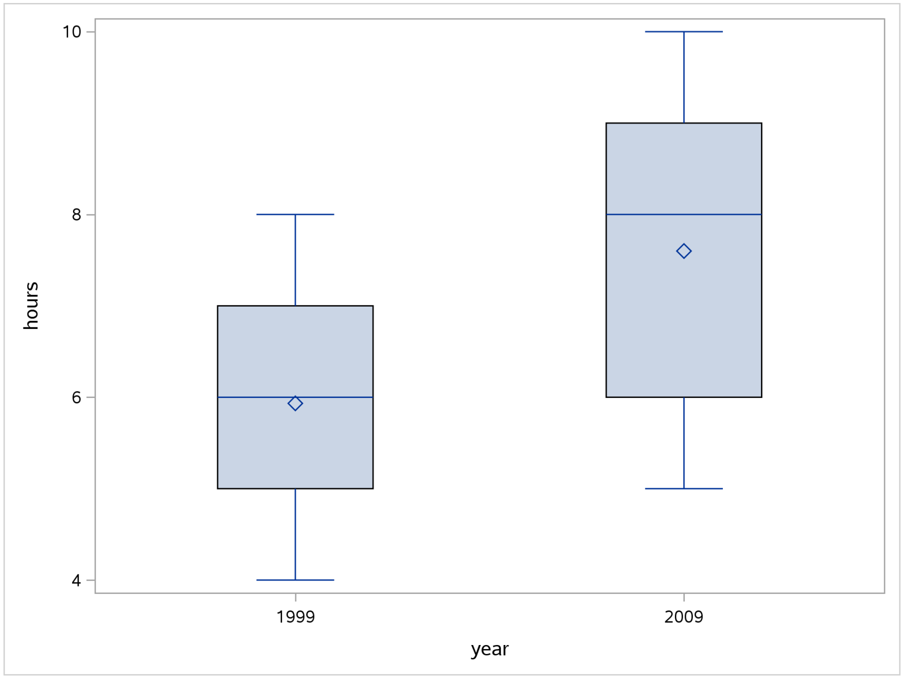
```
                  Obs           year          hours

                    1           1999              4
                    2           1999              5
                    3           1999              7
                    4           1999              7
                    5           1999              5
                    6           1999              7
                    7           1999              5
                    8           1999              6
                    9           1999              5
                   10           1999              6
                   11           1999              7
                   12           1999              8
                   13           1999              5
                   14           1999              6
                   15           1999              6
                   16           2009              5
                   17           2009              9
                   18           2009              5
                   19           2009              8
                   20           2009              7
                   21           2009              6
                   22           2009              7
                   23           2009              9
                   24           2009              7
                   25           2009              9
                   26           2009              6
                   27           2009              9
                   28           2009             10
                   29           2009              9
                   30           2009              8
```

$2 \times 15 = 30$ lines, years 1999 and 2009. Good. (Even though my data values ran off the bottom of the page.)

(c) Draw side-by-side boxplots of hours spent watching electronic devices for each year.

Solution: This is easy, unless you stop to think about it!

```
proc sgplot;
  vbox hours / category=year;
```



(d) What do your boxplots tell you? You're looking for a couple of things: how the means/medians compare, and whether you have any issues with skewness or outliers. Have a think about this before you look at my answer.

Solution: The mean and median for 2009 are quite a bit higher for 2009 compared to 1999, which suggests that the "average" number of hours really has increased.

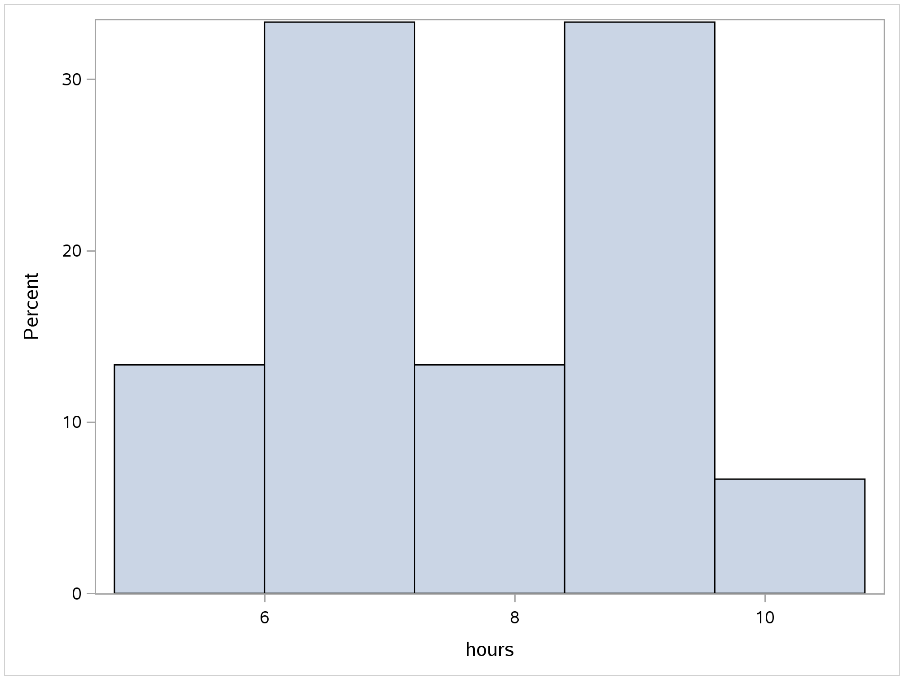
As for skewness and outliers, I really don't see any issues at all: the top and bottom whiskers are about the same length in both cases, and there are no outliers. You might be concerned about the median bar not being in the middle of the box for the 2009 data, but that doesn't really indicate a problem with skewness because that shows up in the *tails*: outliers if you have them, otherwise whisker length.[16]

This kind of thinking is to assess whether a t-test is the right thing to do, and for that we need data that are approximately normal within each group. By "approximately normal", it is generally enough to be able to say, as we have here, that the distributions are more or less symmetric and that we have no serious problems with outliers; in other words, the kind of things that boxplots will tell you.[17]

Extra: if you want to assess the 2009 data further, you can make a histogram via this trick:

```
proc sgplot;
  where year=2009;
  histogram hours;
```

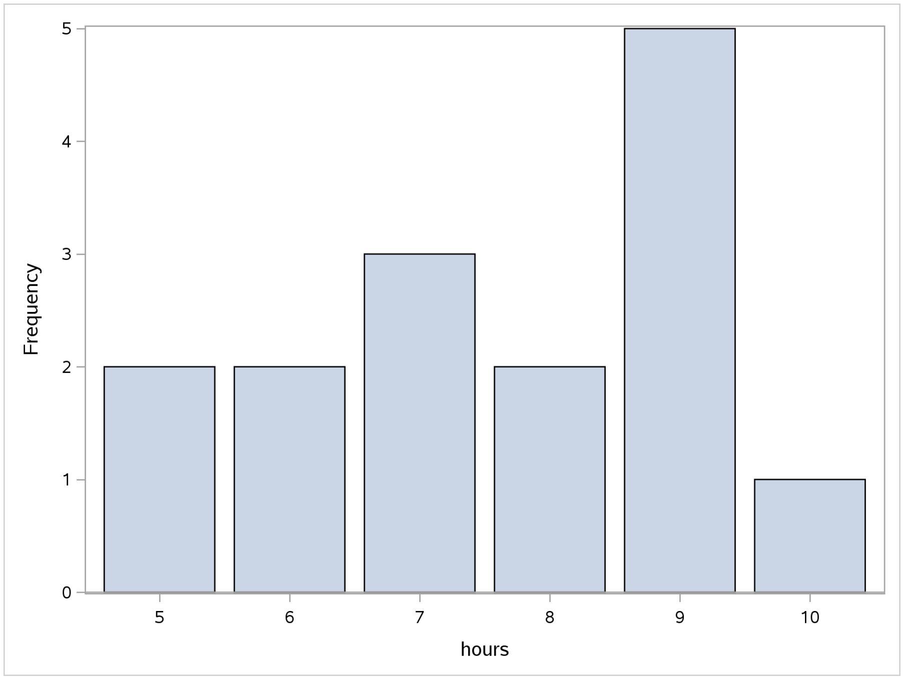
The `where` line says "only use data from year 2009 for this `proc`":


Percent
30
20
10
0
6
8
10
hours

The centre is indeed somewhere near 8, but look at the "hole" in the middle of the distribution (normally a histogram peaks in the middle). I suspect it's this that puts the median asymetrically in the box of the boxplot.

The numbers of hours are actually all integers, so you could reasonably think of the numbers of hours as being (ordered) categorical with a small number of categories:

```
proc sgplot;
  where year=2009;
  vbar hours;
```



This shows that the histogram is actually rather deceiving. The histogram bin between 8 and 10 has a lot of data because of all the 9s, but the histogram bin starting at 6 has a lot of data because that bin happens to contain *both* 6 and 7.

This came up before, and I might have to rewrite my advice about "one quantitative variable". Perhaps the story is "if the number of distinct values is not too much bigger than the number of bins you would use on a histogram, go with a bar chart".

Evidently the median here is 8, Q1 is 6 and Q3 is 9. Because of all the observations that are 9, the median is closer to Q3 than to Q1, and so the top "half" of the box is smaller than the bottom "half". Is this indicative of skewness? Well, maybe; it's enough to make the mean a bit smaller than the median. But it's not the kind of skewness that will cause trouble for the t-test; there is no *long* tail or lower-end outliers. One of those mushy ones that's not so clear.

(e) Carry out a test to determine if there is evidence that the mean number of hours spent per day watching electronic devices has *increased* since 1999. State your null and alternative hypotheses, and from your output, obtain a P-value and interpret it.

Solution: I like to start with the *alternative* hypothesis, since that is what we are trying to prove (which is usually the easiest thing to figure out). Here, that is that the mean in 1999 is *less* than the mean in 2009; in symbols that would be $H_a : \mu_{1999} < \mu_{2009}$. The null hypothesis is that the two means are the same, in symbols $H_0 : \mu_{1999} = \mu_{2009}$, or if this offends your logical sensibilities, $H_0 : \mu_{1999} \geq \mu_{2009}$. Either is good.

All right, getting some output. The only non-default thing here is the one-sidedness of the alternative. To figure out which `sides` you want, note that 1999 is before 2009 (alphabetically, actually, but numerically as well), so you have to express your alternative as how 1999 compares with 2009 *in that order*, as I did above. Or, you can try one of the `sides`, and if the answer makes no sense, try the other one. The intuition from the boxplots is that the P-value should be at least fairly small (since the story there is more in line with the alternative hypothesis than the null). I think the alternative is 1999 less than 2009, so we should have `sides=L`.[18] `side=` and `sides=` both work. The `var` and the `class` are the same as you would use on `proc means`:

```
proc ttest sides=L;
  var hours;
  class year;
```

```
year            N        Mean     Std Dev     Std Err     Minimum     Maximum

1999           15      5.9333      1.0998      0.2840      4.0000      8.0000
2009           15      7.6000      1.5946      0.4117      5.0000     10.0000
Diff (1-2)            -1.6667      1.3697      0.5002
   year          Method               Mean       95% CL Mean        Std Dev

   1999                             5.9333      5.3243   6.5424      1.0998
   2009                             7.6000      6.7169   8.4831      1.5946
   Diff (1-2)    Pooled            -1.6667     -Infty    -0.8158      1.3697
   Diff (1-2)    Satterthwaite     -1.6667     -Infty    -0.8121

               year          Method             95% CL Std Dev

               1999                             0.8052   1.7345
               2009                             1.1675   2.5149
               Diff (1-2)    Pooled             1.0870   1.8525
               Diff (1-2)    Satterthwaite
          Method           Variances        DF    t Value    Pr < t

          Pooled           Equal            28      -3.33    0.0012
          Satterthwaite    Unequal      24.861      -3.33    0.0013
                            Equality of Variances

              Method      Num DF    Den DF    F Value    Pr > F

              Folded F        14        14       2.10    0.1768
```

With SAS, you have to choose whether to use the pooled or the Satterthwaite test. The choice is whether you believe the two samples come from populations with the same SD (pooled) or not (Satterthwaite). It often doesn't make much difference, as here. I think the interquartile range for the 2009 figures is a bit bigger, so (in the absence of outliers) I would expect its SD to be a bit bigger also.[19] Thus here I would choose the Satterthwaite test, though (as I said) it won't make much difference to your conclusion if you disagree with me (and say that the two IQRs are not different enough to be worth worrying about). In any case, the P-value is 0.0013 or 0.0012, smaller than 0.05, and so you *reject* the null hypothesis[20] and conclude that the 2009 mean is indeed larger, for *all* children, not just the ones that happened to be sampled.

The bottom test, the one labelled `Folded F`, is a test for whether the SDs (variances) in the two groups are equal (vs. the alternative that they are not). This null is not rejected, suggesting that we would be entitled to use the pooled test because the two group SDs are not significantly different. It is a mistake, though, to make a formal procedure out of this: to look at the Folded F test first and then decide which two-sample t-test to do. This is because doing it this way messes with the type I error probability. See for example `https://onlinelibrary.wiley.com/doi/pdf/10.1348/000711004849222`.[21]

Extra: if you don't like the skewness in the distribution of hours for 2009, we learned in R that the right test is Mood's median test. We'll get to that in SAS as well, but looking ahead:

```
proc npar1way median;
  var hours;
  class year;
```

```
                         The NPAR1WAY Procedure

     Median Scores (Number of Points Above Median) for Variable hours
                       Classified by Variable year

                       Sum of      Expected       Std Dev          Mean
   year        N       Scores      Under H0      Under H0         Score

   1999       15     4.428571          7.50      1.310717      0.295238
   2009       15    10.571429          7.50      1.310717      0.704762

                     Average scores were used for ties.
                         Median Two-Sample Test

                       Statistic             4.4286
                       Z                    -2.3433
                       One-Sided Pr <  Z     0.0096
                       Two-Sided Pr > |Z|    0.0191
                          Median One-Way Analysis

                       Chi-Square            5.4911
                       DF                         1
                       Pr > Chi-Square       0.0191
```

Later, I get to how this corresponds to what we do in R, but the important point for now is the one-sided P-value of 0.0096. This is not as small as for the t-tests, but it is still significant at $\alpha = 0.01$.

Extra extra bit for those of you that took STAB57 (the rest of you should probably skip ahead): you probably derived the pooled t-test, because the theory is the same as for the one-sample t-test. That, as a reminder, starts like this: suppose you have one sample $x_1, x_2, \ldots, x_n$ from a normal distribution with mean μ and variance σ^2. Then, *if you know* σ^2, $\bar{x} = (1/n)\sum_{i=1}^n x_i$ is exactly normal with mean μ and variance σ^2/n, or equivalently, this thing:

$$z = \frac{\bar{x} - \mu}{\sigma/\sqrt{n}}$$

is exactly standard normal. But if you don't know σ^2, you have to estimate it using the usual unbiased estimate $s^2 = (1/(n-1))\sum_{i=1}^n (x_i - \bar{x})^2$, and then this thing

$$t = \frac{\bar{x} - \mu}{s/\sqrt{n}}$$

has exactly a t-distribution with $n-1$ degrees of freedom. Strictly, $(n-1)s^2/\sigma^2$ has a chi-squared distribution with $n-1$ degrees of freedom, and a normal divided by a suitably scaled chi-squared has a t distribution with the same df. (What you have done is to introduce an extra source of variability in that s is probably not exactly equal to σ, so a t distribution has slightly larger variance[22] and longer tails than the normal.)

Now we think about two samples, the first of size n_1 from a normal distribution with mean μ_1 and variance σ^2, and the second of size n_2 from another independent normal distribution with mean μ_2 and variance σ^2. (The samples can be different sizes.) Thus, the means are possibly different but *the variances are the same.* The usual thing is to compare the two means, so you would estimate $\mu_1 - \mu_2$ via the unbiased estimate $\bar{x}_1 - \bar{x}_2$, the difference of the two sample means. If you knew σ^2, that would have a normal distribution (exactly) with mean $\mu_1 - \mu_2$ and variance $\sigma^2/n_1 + \sigma^2/n_2$. Since there is only one σ, you can also write the variance like this:

$$\sigma^2\left(\frac{1}{n_1} + \frac{1}{n_2}\right)$$

and thus this would have a standard normal distribution (exactly):

$$z = \frac{\bar{x}_1 - \bar{x}_2}{\sigma\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

In the usual case where you don't know σ^2, you have to estimate it. Since you have *one* population variance σ^2 to estimate (common to the two groups), you can estimate it by *one* sample variance

$$s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}$$

(a weighted average of the two sample variances, with the larger sample getting more weight). Thus you have this:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

a normal thing with one σ^2 estimated by an s^2, which is therefore t with the right df, in this case $n_1 + n_2 - 2$.

When the two groups have *different* population SDs, the theory is a whole lot more complicated; in fact, there isn't even *any* exact answer.[23] What Satterthwaite and Welch did[24] was to say that you look at

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

In the second-last lecture of STAB22 this is introduced as "the two-sample t-test", since there they don't see the pooled test. This will have *approximately* a t-distribution. Welch and Satterthwaite independently looked at the thing on the bottom inside the square root:

$$W = \frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}$$

In the one-sample and pooled cases, the thing on the bottom can be scaled to have a chi-squared distribution, but not here, since there are two s^2 terms. The idea is to say that this sum is *approximately* a multiple of chi-squared with some unknown degrees of freedom: that is, you write it as $X = a\chi^2_b$. Since the mean and variance of a chi-squared distribution are both the degrees of freedom, we can work out the mean and variance of X: $E(X) = ab$, $var(X) = a^2 b$.

Now, we go back to the thing I called W above. What are its mean and variance? s_1^2 and s_2^2 are two independent sample variances, so we can get the mean and variance of W by finding the mean and variance of the two pieces and adding them together. So what are they? Well, $(n_i - 1)s_i^2/\sigma_i^2$ has a χ^2_{n-1} distribution, so it has mean and variance $n_i - 1$. Thus

$$E((n_i - 1)s_i^2/\sigma_i^2) = n_i - 1$$

and

$$var((n_i - 1)s_i^2/\sigma_i^2) = n_i - 1$$

and thus

$$E\left(\frac{s_i^2}{n_i}\right) = \frac{\sigma_i^2}{n_i}$$

(the $n_i - 1$ terms cancelling; note that this also says that s_i^2 is an unbiased estimator of σ_i^2), and

$$var\left(\frac{s_i^2}{n_i}\right) = \frac{\sigma_i^4}{n_i^2(n_i - 1)}$$

(hence s_i^2 is also a consistent estimator of σ_i^2: look at the powers of n_i.)

Thus W has mean and variance

$$E(W) = \frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}$$

and

$$var(W) = \frac{\sigma_1^4}{n_1^2(n_1 - 1)} + \frac{\sigma_2^4}{n_2^2(n_2 - 1)}$$

To make W be a multiple of a chi-squared with some df, we match the mean and variance of W with the mean and variance of X (method of moments is what we're doing), to get

$$ab = \frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}$$

and

$$ab^2 = \frac{\sigma_1^4}{n_1^2(n_1-1)} + \frac{\sigma_2^4}{n_2^2(n_2-1)}$$

and "simply" solve for a and b. The business end of this is the degrees of freedom b, and if you compare the left-hand sides of the two equations above, you'll see that we can get b by dividing the two equations by each other and the a will cancel:

$$b = \frac{\frac{\sigma_1^4}{n_1^2(n_1-1)} + \frac{\sigma_2^4}{n_2^2(n_2-1)}}{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}$$

The final problem here is that the σ_i are not known, so we have to approximate *those* using the sample variances s_i. Then, as the very last stage, this df number is used as the df for the t-distribution.

It is likely that I have made some errors here, but that's the idea. There are, as you see, approximations upon approximations: we don't *know* that $s_1^2/n_1 + s_2^2/n_2$ has approximately a chi-squared distribution, and even if that's all right, we are estimating parameters by method of moments rather than maximum likelihood, and even then we are replacing σ_i^2 by s_i which is (we hope) somewhere close to it. If you think that statistics is nothing more than rigorous mathematics, you are likely to be in for a disappointment.

(f) Obtain a 99% confidence interval for the difference in means. You'll have to get some more output for this.

Solution: The reason that we have to do some more work is that a confidence interval is by its nature a *two-sided* thing, so we have to do a two-sided test to get it. (The confidence intervals in the previous part were *one-sided*: they started at minus-something and went all the way down to minus infinity.) So we have to take out the `sides` thing and put in something that will get us a 99% CI, namely `alpha=0.01`, since that's what SAS works with:

```
proc ttest alpha=0.01;
  var hours;
  class year;
```

```
year            N        Mean     Std Dev     Std Err     Minimum     Maximum

1999           15      5.9333      1.0998      0.2840      4.0000      8.0000
2009           15      7.6000      1.5946      0.4117      5.0000     10.0000
Diff (1-2)            -1.6667      1.3697      0.5002
   year          Method                Mean       99% CL Mean         Std Dev

   1999                              5.9333      5.0880    6.7786      1.0998
   2009                              7.6000      6.3743    8.8257      1.5946
   Diff (1-2)    Pooled             -1.6667     -3.0487   -0.2846      1.3697
   Diff (1-2)    Satterthwaite      -1.6667     -3.0615   -0.2719

               year          Method              99% CL Std Dev

               1999                              0.7353    2.0386
               2009                              1.0662    2.9558
               Diff (1-2)    Pooled              1.0150    2.0532
               Diff (1-2)    Satterthwaite
```

```
Method           Variances        DF    t Value    Pr > |t|

Pooled           Equal            28      -3.33      0.0024
Satterthwaite    Unequal      24.861      -3.33      0.0027
                       Equality of Variances

        Method      Num DF    Den DF    F Value    Pr > F

        Folded F        14        14       2.10    0.1768
```

The confidence interval goes from -3.06 to -0.27 (hours). Note that SAS did indeed label it as a 99% interval, so that we have the right thing. I pulled out the Satterthwaite interval, since that's what I thought was best; if you think the pooled test was OK, then use the pooled interval here. (As for the tests, they are not very different).

(g) What does your confidence interval tell you, in the context of the data?

Solution: It says that the time spent by a child per day in front of an electronic device has *increased* by between 0.3 and 3.1 hours between 1999 and 2009. (The negative numbers mean that the 1999 values were less on average than the 2009 ones.)

(h) Does your confidence interval contain zero? Does that surprise you? Why, or why not?

Solution: No, it doesn't: all the values in the interval are negative. This does not surprise me, because we said from the test that the mean number of hours had significantly increased, and we would therefore expect a CI all on one side of zero (negative, because of the way the numbers were).

This is strictly speaking not quite right (though I didn't need you to observe this), because the correspondence is between a *two-sided* test and a confidence interval (or between the one-sided test and the one-sided confidence interval, which didn't contain zero either). We got away with it here, though, because the P-value was so small that even the two-sided P-value was still safely less than 0.01 (to go with the 99% CI).

The confidence interval was quite wide. This is partly because it was a 99% one; a 90% CI would be narrower. This says that we know that there *was* an increase in the mean number of hours spent watching electronic devices, but that we don't have a very precise idea of how big that increase was. This is unfortunately all too common.

4.5. How long does it take students to get to school? A survey was done of British secondary school students, and a similar survey of Ontario high-school students, with 40 students in each (which, you may assume, are a random sample of their respective populations). In both surveys, the "typical" time taken to get to school was recorded. The question of interest is whether there is a difference in the time students take to get to school in Ontario and the UK. The data are in `http://www.utsc.utoronto.ca/~butler/c32/to-school.csv`.

(a) Read the data into SAS. There should be two columns, `traveltime` and `location`. Obtain the mean travel time for each location. How many travel times do you have at each location?

Solution: The usual business for reading in a `.csv`:

```
filename myurl url 'http://www.utsc.utoronto.ca/~butler/c32/to-school.csv';

proc import
  datafile=myurl
  dbms=csv
  out=mydata
  replace;
  getnames=yes;
```

Normally you'd follow this with `proc print`, which you probably should for yourself, but there are 80 lines of data, a lot to hand in, so I asked you to summarize things, thus:

```
proc means;
  var traveltime;
  class location;
```

```
                              The MEANS Procedure

                         Analysis Variable : traveltime

              N
location    Obs    N          Mean         Std Dev         Minimum         Maximum
-----------------------------------------------------------------------------------
Ontario      40   40    17.0000000       9.6609178       2.0000000      47.0000000

UK           40   40    20.6500000      13.1276768       3.0000000      60.0000000
-----------------------------------------------------------------------------------
```

There are 40 travel times in each location; the mean for the Ontario students is 17 minutes, and the mean for the British students is 20.65 minutes. *You need to say this.*

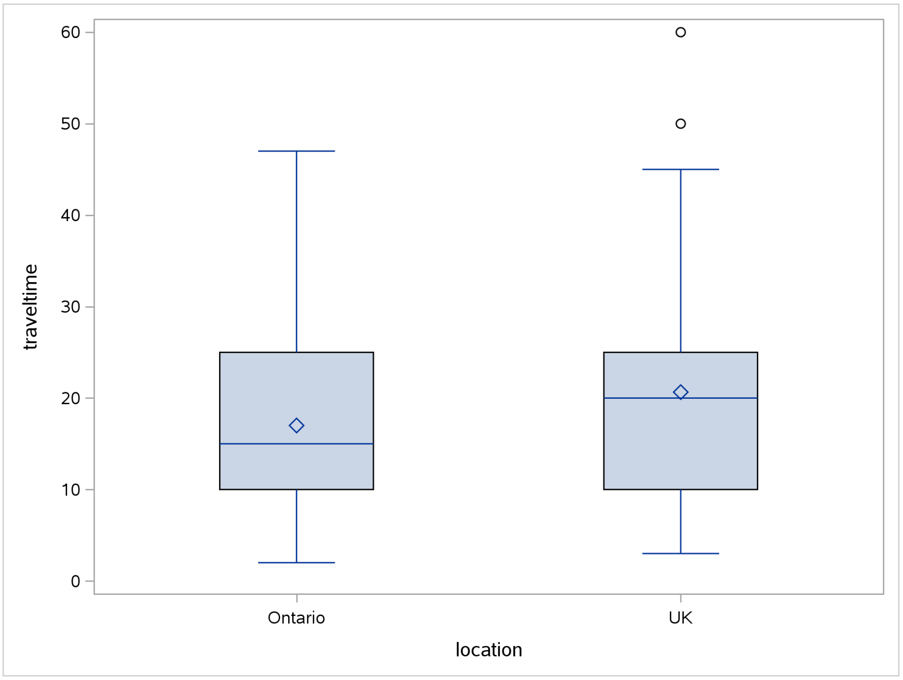
If something went astray with the reading in, it will probably show up here, and that would alert you to check what you did.

The first time I did this, I had the British times first in the data file, and the locations were labelled `UK` and `On` (it seems to take the maximum length from the first one it finds[25]). But I didn't want you to be dealing with that, so I switched things around in the data file.

(b) Make a suitable plot of travel times for each location. Describe the shapes of the distributions. Do they have similar spreads?

Solution: There is one quantitative variable here, travel time, and one categorical one, location, so the obvious thing is a side-by-side boxplot:

```
proc sgplot;
  vbox traveltime / category=location;
```


60
50
40
30
20
10
0
traveltime
Ontario
UK
location

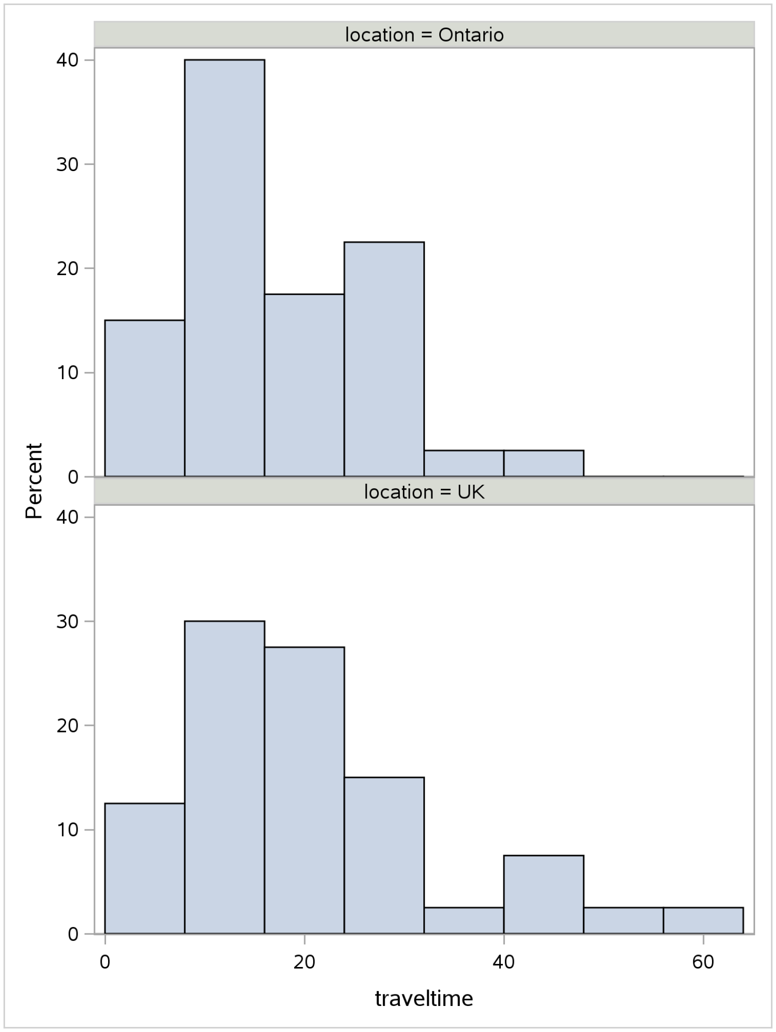
I think both of those distributions are skewed to the right: look at the longer upper whiskers, and the outliers on the UK distribution. However, the spreads, as measured by the heights of the boxes, look very similar to me.

Two points for a suitable graph, and one each for appropriate comment about shape and and about spreads.

Another possibility for a graph would be a paneled histogram, but this requires extra cajoling to come out above and below, like this:

```
proc sgpanel;
  panelby location / columns=1;
  histogram traveltime;
```

The `columns=1` arranges all the plots in one long vertical column, which is what we want:



The right-skewed shape is obvious enough (though less obvious than on the boxplots), but how are you going to assess spread well enough to conclude that it's about the same? Maybe looking at the bin from 40 to about 46 that has 3 observations in it for the UK data, and only one for the Ontario data. Or you can go back to the SDs in the output from `proc means`, where the SD for the UK measurements is a bit higher, but is that because of the outliers?

(c) Is there any evidence of a difference in mean travel time between the two locations? Run a suitable *t*-test. What do you conclude?

Solution: This code, the test being two-sided:

```
proc ttest;
  var traveltime;
  class location;
```

Note that the `var` and `class` are exactly as for the `proc means`.

```
location        N        Mean     Std Dev     Std Err     Minimum     Maximum

Ontario        40     17.0000      9.6609      1.5275      2.0000     47.0000
UK             40     20.6500     13.1277      2.0757      3.0000     60.0000
Diff (1-2)            -3.6500     11.5254      2.5772
   location      Method               Mean       95% CL Mean        Std Dev

   Ontario                         17.0000     13.9103  20.0897      9.6609
   UK                              20.6500     16.4516  24.8484     13.1277
   Diff (1-2)    Pooled            -3.6500     -8.7807   1.4807     11.5254
   Diff (1-2)    Satterthwaite     -3.6500     -8.7879   1.4879

               location      Method            95% CL Std Dev

               Ontario                          7.9138  12.4050
               UK                              10.7537  16.8564
               Diff (1-2)    Pooled             9.9662  13.6676
               Diff (1-2)    Satterthwaite
         Method           Variances        DF    t Value    Pr > |t|

         Pooled           Equal            78      -1.42      0.1607
         Satterthwaite    Unequal      71.663      -1.42      0.1610
                            Equality of Variances

              Method      Num DF    Den DF    F Value    Pr > F

              Folded F        39        39       1.85    0.0590
```

You should make a choice between the pooled and Satterthwaite tests, based on whether you thought the spreads were similar or noticeably different. My choice was that the spreads (IQRs) were almost the same, so I would go with the pooled P-value 0.1607. If you thought the spreads were different (eg. by considering the SDs), you need to use the Satterthwaite P-value 0.1610. I don't mind which way you go, but you need to be *consistent* with what you said before. That's the key. In either case, you fail to reject the null, and therefore conclude that there is no evidence of any difference between the mean travel times in the two places.

If you look at the two confidence intervals for the difference in mean, you'll see that they both include 0, which is another way of seeing that you are likely not going to be rejecting a hypothesis that the difference in population means is zero.

I guess another way of choosing between the two tests is to note that the P-values are almost identical, so that it doesn't matter which one you choose. This is, perhaps, surprising, given that equality of variances (the bottom test) is almost rejected, perhaps because the SD of the UK measurements is larger, inflated, perhaps, by the upper outliers in that distribution.

(d) Do you have any concerns about the t-test you just did? Explain briefly why or why not.

Solution: The critical assumption here is of approximately normal distributions *within* each group. Equal spreads does not matter. Go back to the boxplots you drew earlier, and make a call: does it look as if *both* groups have approximately normal distributions? I would say not, because I think they're both skewed to the right. That's a good answer.

Having said that, the skewness probably matters (somewhat) less than you think, because you have the Central Limit Theorem ($n = 40$ in each group) working in your favour. Is the sample size large enough to overcome the skewness that you see? That's a hard question to answer, without resorting to something like bootstrapping, but it's reasonable to answer that the skewness may not be causing as much of a problem as it appears because of the largish sample sizes.

Extra: there's a lot of hand-waving involved in all of this, and it may be difficult to get a definitive answer about whether we should do the two-sample t-test or something else (eg. Mood's median test, coming up later), but I want you to get at the issues in your answer: at least, that both distributions are skewed right, so that approximate normality fails, but possibly also that we have $n = 40$ in both groups so the Central Limit Theorem is in our favour (and the normality doesn't matter as much).

One other thing in among the infinity of issues here is that it helps if both groups are *skewed in the same direction*, as here, because whichever t-statistic you calculate, you subtract the sample means, and this allows the skewness to "cancel out", or at least get reduced. The idea is that you might get an unusually large travel time in either group, and those will *both* inflate the mean in that group upwards, so that when you subtract the means this effect is dampened down. (Compare if one group were skewed right and the other skewed *left*; then you'd have the sample means being pulled potentially opposite ways by unusual values and the difference could, if you were unlucky, be pulled a long way away from zero.)

One way to do less hand-waving, as I hinted above, is via the "bootstrap". The idea behind this is that you treat the observed data as populations, and then you sample from them *with* replacement.[26] You calculate the test statistic each time, and then this gives an idea of what the sampling distribution looks like. Since we're doing a test which is based on a null hypothesis, we can arrange things beforehand so that the means actually *are* equal, and then see how often we mistakenly reject. Here, though, I'm interested in the shape of the sampling distribution: if it's normal, using a t-test will have no problems.

As you might expect, R is the tool for this. Let's start with a small one where the distributions are skewed opposite ways, so we expect it to go wrong. But first:

```
library(tidyverse)

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

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

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

set.seed(457299)
```

and then some data to play with:

```
d=tribble(~value, ~gp,
          -3, "X",
          -2, "X",
          -1, "X",
          6, "X",
          -6, "Y",
          1, "Y",
          2, "Y",
          3, "Y")
d
```

```
## # A tibble: 8 x 2
##   value gp
##   <dbl> <chr>
## 1    -3 X
## 2    -2 X
## 3    -1 X
## 4     6 X
## 5    -6 Y
## 6     1 Y
## 7     2 Y
## 8     3 Y
```

The values in group X are skewed to the left, and the values in group Y are skewed to the right. Both groups have mean zero. So we know the means are actually equal, but the question is what kind of test statistic values we might get, or at least how different the sample means might be.

All right, to bootstrapping.

Let's play with the data in `d` while we get the feel for what's going on. There are two groups `X` and `Y` with four values in each:

```
d %>% count(gp)
```

```
## # A tibble: 2 x 2
##   gp        n
##   <chr> <int>
## 1 X         4
## 2 Y         4
```

but if you just randomly sample 8 rows with replacement, this kind of thing can happen:

```
d %>% sample_frac(replace=T)
```

```
## # A tibble: 8 x 2
##   value gp
##   <dbl> <chr>
## 1     3 Y
## 2    -3 X
## 3    -2 X
## 4    -6 Y
## 5     6 X
## 6     2 Y
## 7    -2 X
## 8    -2 X
```

Oops, five rows from `X` and only three from `Y`.

(This, if you have not seen the idea before, samples from all the rows with replacement. If you supply a number as the first input to `sample_frac`, it randomly samples that fraction of all the rows, the default fraction being 1.)

A bootstrap resample has to have four rows in each of the two groups, and this one doesn't. How to make sure each group gets represented the right number of times? The idea is to group_by the groups first, so that the sampling happens *within* each group:[27]

```
d %>% group_by(gp) %>%
  sample_frac(replace=T)
```

```
## # A tibble: 8 x 2
## # Groups:   gp [2]
##   value gp
##   <dbl> <chr>
## 1    -1 X
## 2     6 X
## 3     6 X
## 4    -2 X
## 5    -6 Y
## 6     1 Y
## 7     3 Y
## 8     2 Y
```

Now we have four rows from each group.

The next stage is to note that the calculation of the difference in sample means (our test statistic) is somewhat complicated, so we should probably have a function to do it. Our function will accept a data frame, a column of values to calculate with, and a grouping column (we'll assume there are exactly two groups):

```
mean_diff=function(d, x, group) {
   d %>% group_by({{ group }}) %>%
      summarize(m=mean({{ x }})) %>%
      pull(m) -> means
   means[1]-means[2]
}
mean_diff(d, value, gp)
```

```
## [1] 0
```

There are some mysterious curly brackets in there, which I'll get to in a minute.

The idea is to group by whatever you said the grouping variable was, summarize each group by the mean of whatever variable you're measuring for each group, pull this out as a vector and then return the first group's mean minus the second group's mean.

About those curly brackets: this is called "tidy evaluation" and is the mechanism behind referring to column names without putting quotes around them. In return for this, any time you use something in a function that might be an unquoted column name, you have to put the double curly brackets around it.[28]

The last line tests that it works for our mini data frame `d`, and it does, because the two means are the same.

Our bootstrap is a two-liner also, so I'll write a function for that as well, again using curly-curly:

```
boot2=function(d, x, group) {
   d %>% group_by({{ group }}) %>%
      sample_frac(replace=T)
}
z=boot2(d, value, gp)
z
```

```
## # A tibble: 8 x 2
## # Groups:   gp [2]
##   value gp
##   <dbl> <chr>
## 1    -1 X
## 2    -2 X
## 3     6 X
## 4    -1 X
## 5    -6 Y
## 6     1 Y
## 7    -6 Y
## 8     2 Y
```

There is one bootstrap sample, as a data frame, and we can work out the difference in means thus:

```
mean_diff(z, value, gp)
```

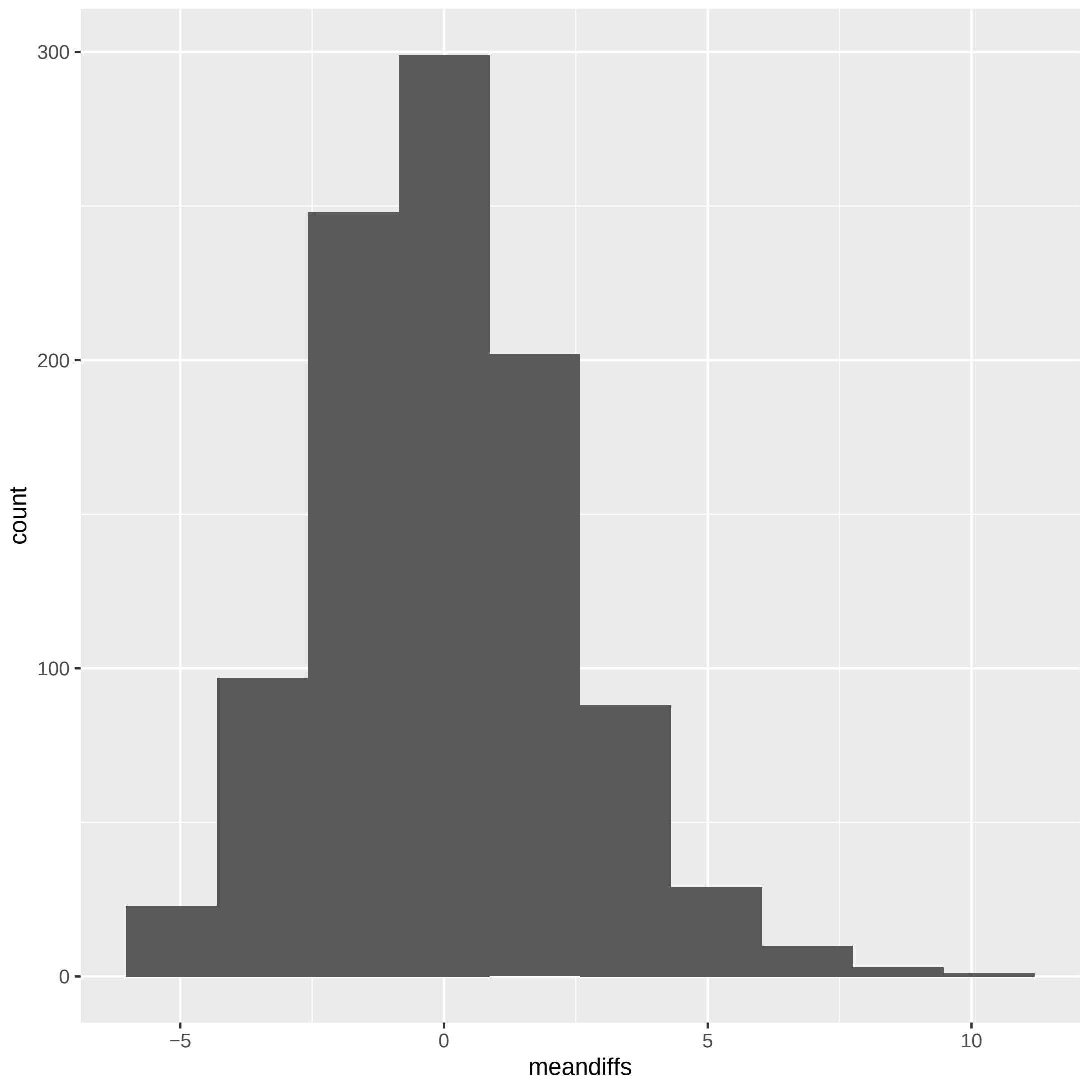
```
## [1] 2.75
```

So now we can do a whole bootstrap:

```
rerun(1000, boot2(d, value, gp)) %>%
  map_dbl(~mean_diff(., value, gp)) -> meandiffs
```

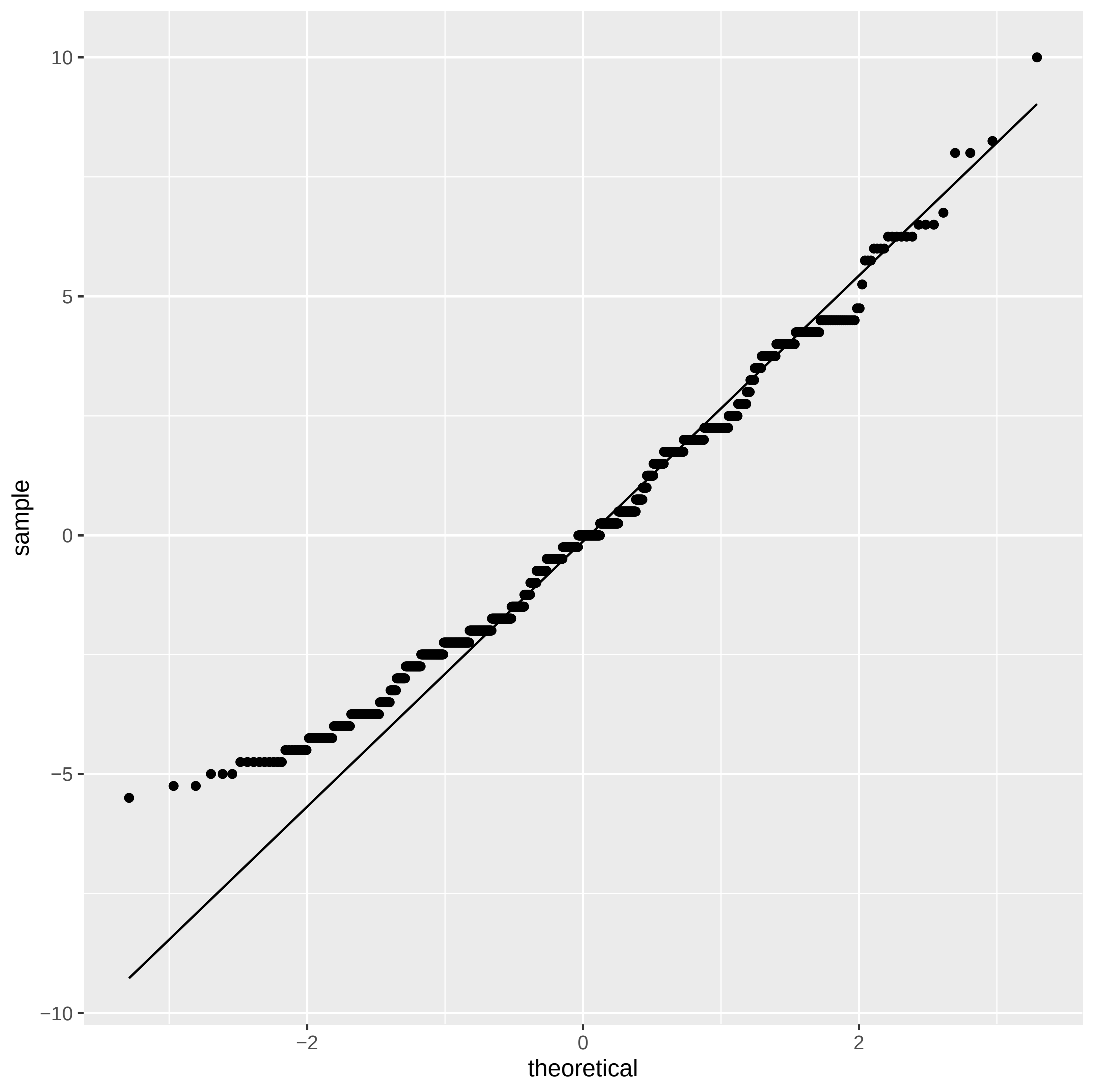
and take a look at that, and assess it for normality:

```
ggplot(tibble(meandiffs), aes(x=meandiffs)) + geom_histogram(bins=10)
```



Skewed to the right, because it is possible for the difference in means to be very positive but not very negative (group X has a very positive value in it and group Y a very negative one; if these values appear in the resamples, the difference could be very large and positive).

```
ggplot(tibble(meandiffs), aes(sample=meandiffs)) + stat_qq() + stat_qq_line()
```



This shows that the reason for the right skewness is, surprisingly enough, not that the right tail is too long but that the left tail is too *short*.

Now that we have the machinery, we can do all the same things again with the school travel times data set. There's not really any extra work, thanks to the functions we wrote; it's just a matter of changing some names:

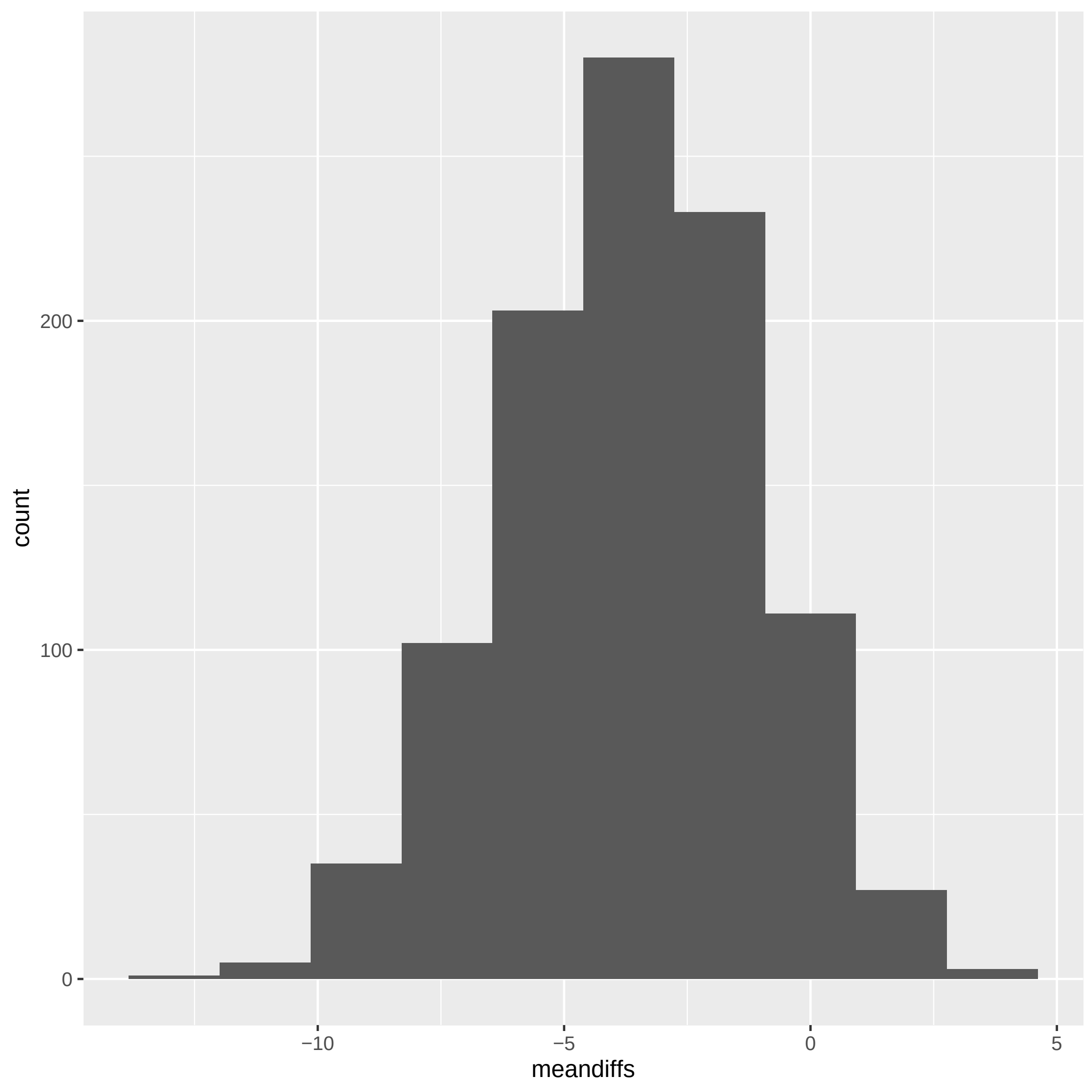
```
my_url="http://www.utsc.utoronto.ca/~butler/c32/to-school.csv"
to_school=read_csv(my_url)
```

```
## Parsed with column specification:
## cols(
##   traveltime = col_double(),
##   location = col_character()
## )
```

```
to_school
```

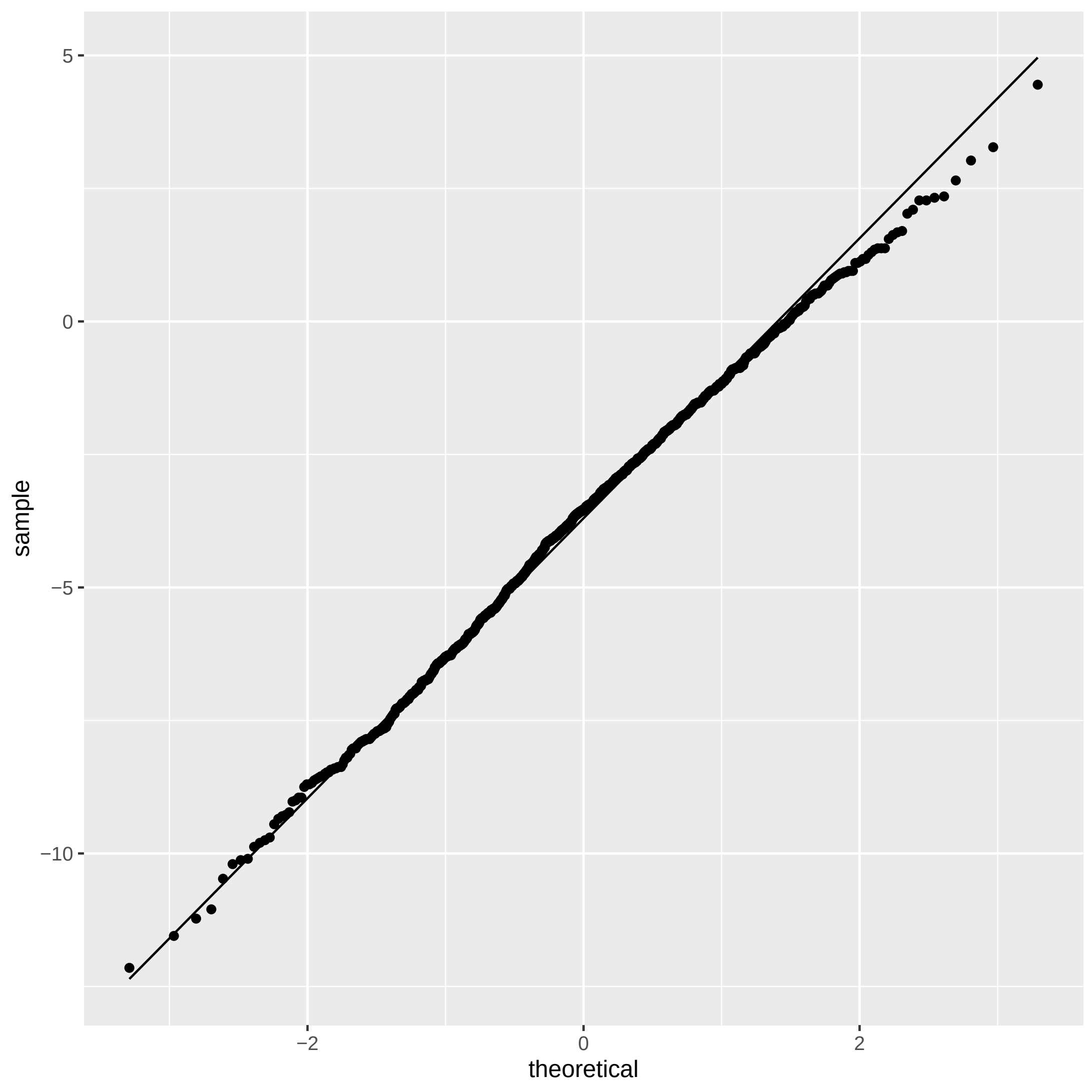
```
## # A tibble: 80 x 2
##    traveltime location
##         <dbl> <chr>
##  1         30 Ontario
##  2         10 Ontario
##  3          8 Ontario
##  4         30 Ontario
##  5          5 Ontario
##  6          8 Ontario
##  7          7 Ontario
##  8         15 Ontario
##  9         10 Ontario
## 10         35 Ontario
## # ... with 70 more rows
```

```
rerun(1000, boot2(to_school, traveltime, location)) %>%
   map_dbl(~mean_diff(., traveltime, location)) -> meandiffs
ggplot(tibble(meandiffs), aes(x=meandiffs)) + geom_histogram(bins=10)
```



This looks skewed to the left, but the normal quantile plot shows you that we really have nothing to worry about:

```
ggplot(tibble(meandiffs), aes(sample=meandiffs)) + stat_qq() + stat_qq_line()
```



That is to say, a two-sample t-test for these data is perfectly reliable, even with the skewness in the original data, because (apparently) the sample sizes are big enough. Thus, there is no need to go further for these data.

You can (with sufficient organization) do a bootstrap in SAS also, but I have no intention of making you struggle with that. If you really want to know, `https://blogs.sas.com/content/iml/2016/08/10/bootstrap-confidence-interval-sas.html` is a place to start.

4.6. Random samples of healthy men and women were taken, and their heart rates measured under normal conditions. The data are from an Excel spreadsheet, saved at `http://www.utsc.utoronto.ca/~butler/c32/heart-rates.csv`. We are interested in whether males and females differ in average heart rate.

(a) (2 marks) Read the data into SAS and display the data set.

Solution: This is a `.csv` file (it was created from a spreadsheet), so make sure the `dbms` line says that:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/heart-rates.csv";

proc import
  datafile=myurl
  out=heartrates
  dbms=csv
  replace;
  getnames=yes;

proc print;
```

with output

```
Obs    gender      heartrate

  1    male               74
  2    male               80
  3    male               75
  4    male               69
  5    male               58
  6    male               76
  7    male               78
  8    male               78
  9    male               86
 10    male               84
 11    male               71
 12    male               80
 13    male               75
 14    female             75
 15    female             66
 16    female             57
 17    female             87
 18    female             89
 19    female             65
 20    female             69
 21    female             79
 22    female             85
 23    female             59
 24    female             65
 25    female             80
 26    female             74
```

Extra: this was originally an Excel spreadsheet called `heart-rates.xlsx`, which you can find at `http://www.utsc.utoronto.ca/~butler/c32/heart-rates.xlsx`.[29] I thought you could read an `.xlsx` file directly from a URL, but SAS has the same problem with that as `read_excel` does. So to read that you would have to do two extra steps:

1. *download* the spreadsheet from the URL (change the `.csv` at the end to `.xlsx`; it goes into `Downloads` or wherever it goes on your computer)
2. *upload* the spreadsheet from your computer to SAS Studio.

Then you read it in via the `/home/username` thing. This is how it looks for me (you'd need to replace my username with yours). Note that there is no `filename` line this way because you are going to put the file's name directly on the `datafile` line:

```
proc import
  datafile='/home/ken/heart-rates.xlsx'
  out=heartrates
  dbms=xlsx
  replace;
  getnames=yes;
  sheet=Sheet1;

proc print;
```

Note that I needed to specify the worksheet name (since there might have been more than one, though in this case there wasn't).

This gives the same output (since it's the same data):

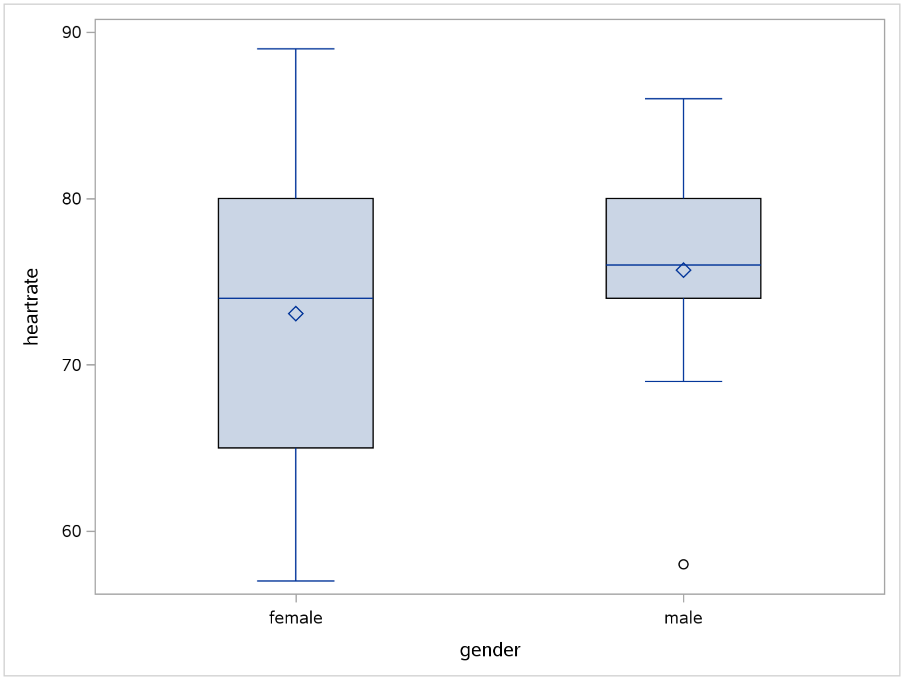
```
Obs    gender    heartrate

  1    male         74
  2    male         80
  3    male         75
  4    male         69
  5    male         58
  6    male         76
  7    male         78
  8    male         78
  9    male         86
 10    male         84
 11    male         71
 12    male         80
 13    male         75
 14    female       75
 15    female       66
 16    female       57
 17    female       87
 18    female       89
 19    female       65
 20    female       69
 21    female       79
 22    female       85
 23    female       59
 24    female       65
 25    female       80
 26    female       74
```

This is now the "most recently created" data set, so it's the one that will get used below, but it's the same as yours.

(b) (2 marks) Make a suitable plot of the two variables.

Solution: One quantitative and one categorical ought to suggest "boxplot" to you:

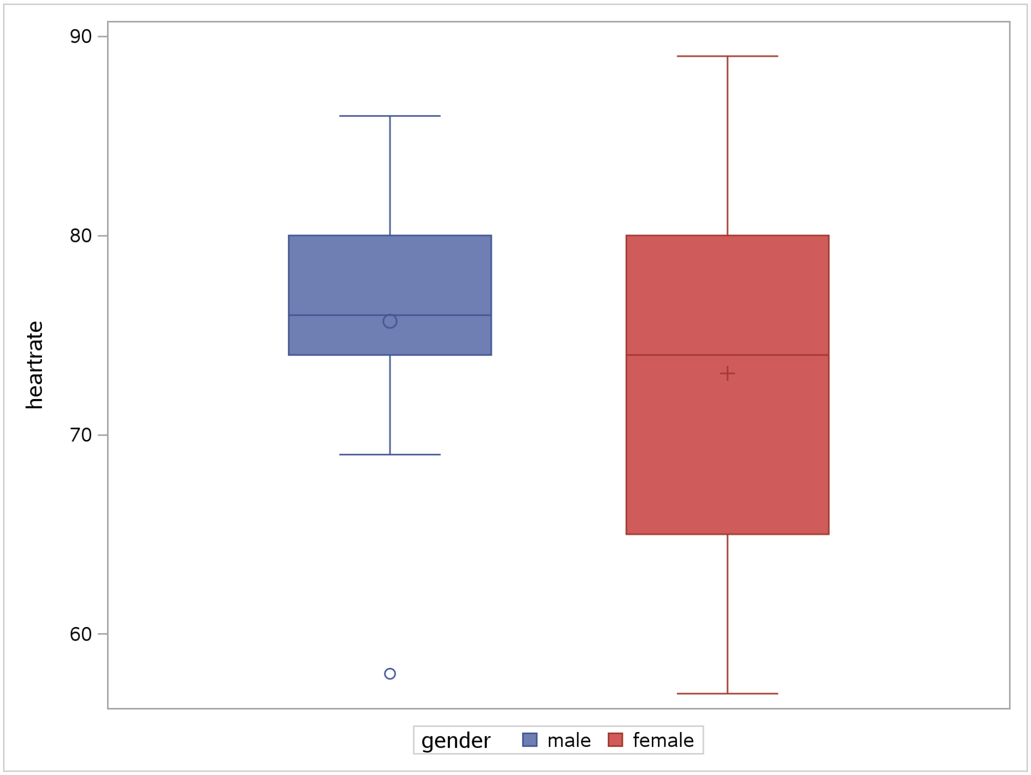
```
proc sgplot;
  vbox heartrate / category=gender;
```


90
80
70
60
heartrate
female
male
gender

I'm not asking you for comment yet, but looking ahead to a two-sample *t*-test, I hope you're looking for outliers, skewness and (in terms of which two-sample *t*-test to run) whether the genders have similar spread or not. That is coming up later.

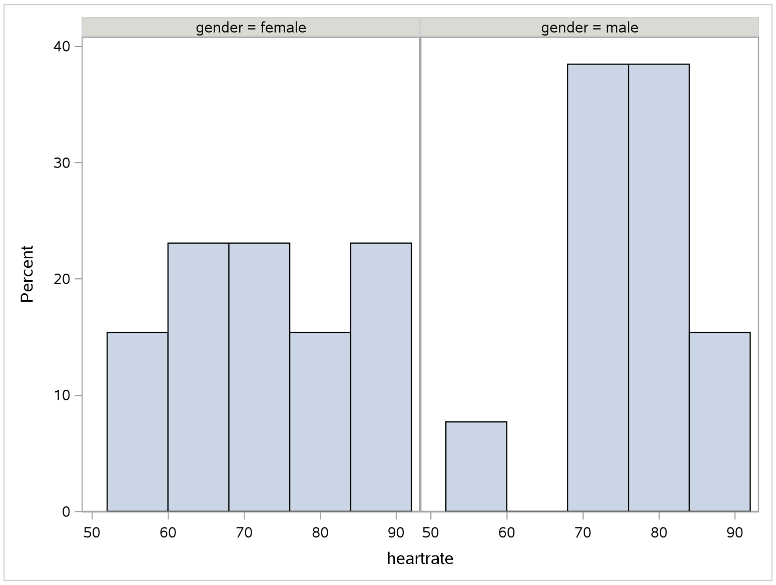
This gets you coloured boxplots, which it is up to you whether you prefer. For some reason, they also come out the other way around:

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



Faceted histograms would also work:

```
proc sgpanel;
  panelby gender;
  histogram heartrate;
```



(c) (3 marks) Run the most appropriate t-test to compare the mean heart rate for males and females. What do you conclude, in the context of the data?

Solution: This is `proc ttest`, used to do a two-sample t-test. SAS's approach is to give you both the Satterthwaite-Welch and the pooled tests, and leave it to you to choose which one you want:

```
proc ttest;
  var heartrate;
  class gender;
```

Here's the (text part of the) output:

```
gender          N        Mean     Std Dev     Std Err     Minimum     Maximum

female         13     73.0769     10.5314      2.9209     57.0000     89.0000
male           13     75.6923      7.1108      1.9722     58.0000     86.0000
Diff (1-2)            -2.6154      8.9854      3.5244
   gender        Method                Mean       95% CL Mean        Std Dev

   female                           73.0769     66.7129  79.4410     10.5314
   male                             75.6923     71.3953  79.9893      7.1108
   Diff (1-2)    Pooled             -2.6154     -9.8893   4.6585      8.9854
   Diff (1-2)    Satterthwaite      -2.6154     -9.9434   4.7127

               gender        Method             95% CL Std Dev

               female                           7.5519  17.3845
               male                             5.0991  11.7381
               Diff (1-2)    Pooled             7.0160  12.5000
               Diff (1-2)    Satterthwaite
         Method           Variances        DF    t Value    Pr > |t|

         Pooled           Equal            24      -0.74      0.4652
         Satterthwaite    Unequal      21.059      -0.74      0.4662
                            Equality of Variances

              Method      Num DF    Den DF    F Value    Pr > F

              Folded F        12        12       2.19    0.1881
```

I looked at the boxplot and said to myself "the females have a bigger spread than the males do", and so I chose the Satterthwaite test, with a P-value of 0.4662. This ignores the low outlier in the males, which will inflate the SD some, but if you look at the output from `proc ttest`, the females have SD 10.5 and the males 7.1 (smaller, even including the outlier), which are not very close to being equal in my opinion, though you are free to disagree. For instance, if you want to say "the group SDs are not all that different, so I use the pooled t-test and obtain a P-value of 0.4652", I'm good with that.

In summary, pick *one* of the P-values, say which one it is, and give some defensible reason why you chose it. If you say something like "the P-value is not less than 0.05", you are *wrong*, because there are two P-values (at least) and you haven't said which one you're looking at (or why). Saying *which* P-value you are using, if there is more than one, is *always* a good idea, on exams too. Otherwise it makes your work look sloppy. You might think that "none of the P-values are significant" would do it, but that includes the "folded F" that is a test for comparing *variances* as well, so that doesn't work. "Neither the Welch-Satterthwaite nor the pooled two-sample t-tests are significant" *would* do it, but if you're going to say that, you might as well pick *one* t-test and talk about that.

Either way, the P-value is not anywhere near small enough to reject with, so there is no evidence that males and females differ in mean heart rate.

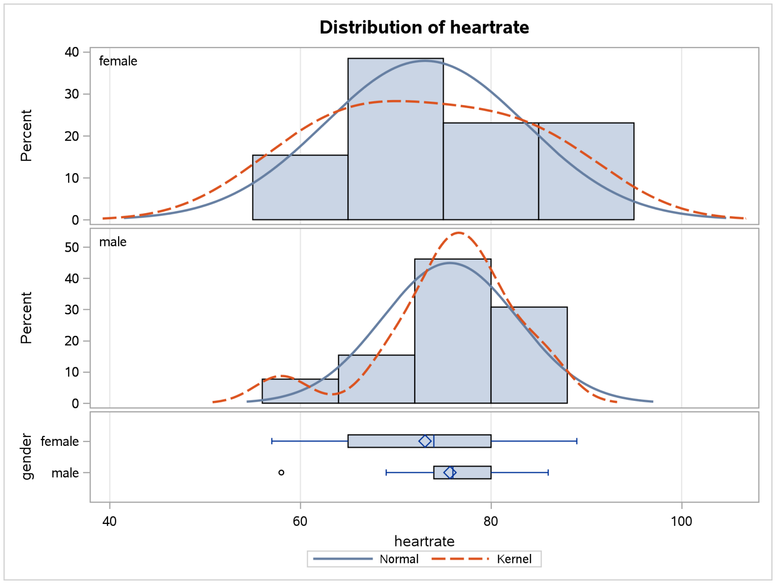
I carefully said "most appropriate t-test" in the question. If you don't think any kind of t-test is OK, I am asking you to choose the "least inappropriate" one, since the task here is to practice running a two-sample t-test, and *then* to think about what else you might do.

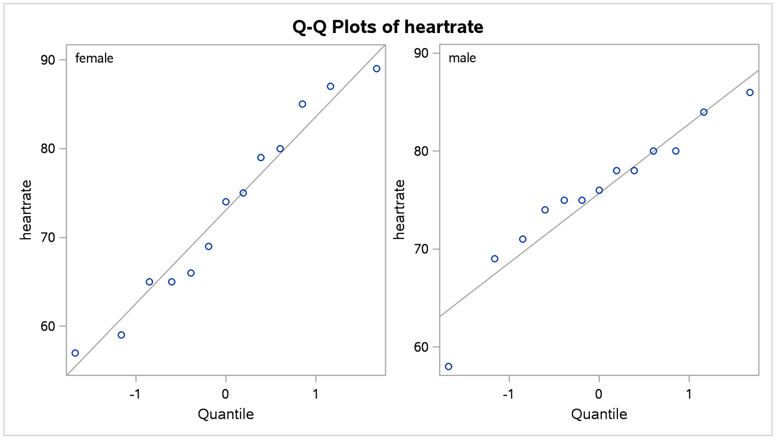
Extra: even though it looks that the groups are rather different in spread, the two P-values for the pooled and Satterthwaite tests are almost identical, as in almost all of the examples we've seen. I would even (here) accept "the two P-values are almost the same so it doesn't matter which test we do" (that is to say, Welch-Satterthwaite and pooled) as a defensible reason for *not* picking one over the other.

(d) (2 marks) Would you trust the result of your test? Explain briefly why or why not. Refer back to any of your previous output as necessary.

Solution: This is an invitation to go back to your graph and check for symmetry and outliers. Symmetry appears OK (all the whiskers are about the same length), but that low outlier on the males troubles me. That would be a reason to *not* trust a t-test here. Having said that, you could defend the t-test in a couple of ways: (i) despite the outlier, the mean is not pulled down very much compared to the median, (ii) the test is so far from being significant that abandoning the t-test and doing something like Mood's median test instead is very unlikely to change the P-value much. (I come back to this in a moment.)

Extra: you might have noticed that `proc ttest` produces graphs along with the text output. I would also (happily) accept a critique of your t-test based on one of these graphs, as long as you say which one(s) and why. (I'm looking for one reason, probably the outlier, why you don't like the t-test, but that might be supported by more than one of these graphs.)


Distribution of heartrate
female
male
Percent
gender
heartrate
Normal
Kernel
40
60
80
100


Q-Q Plots of heartrate
female
male
heartrate
Quantile

The top plot shows histograms for the males and females. It is a little hard to assess these for normality with only 13 observations in each group, though the male histogram looks a bit skewed left. The red kernel density curves are pretty close to the blue normal curves in each case, though the kernel density curve for the males has a "bump" where the outlier appears to be. I think the decision about whether the smallest value is an outlier will be very dependent on the choice of bins for the histogram; the outlier looks a lot clearer on the boxplot, reproduced below the histograms. (I ran into another example like this, where the outlier was a lot less clear on a histogram, which makes me wonder whether this happens often. Unless you tweaked the bins to make it show up, once you'd found where it was. Tweaking the bins like this has, to my mind, crossed over the line into cheating.)

The bottom plot shows normal quantile plots for each group. I have to say that the normality looks pretty acceptable with these sample sizes; I was expecting the low outlier on the males to stand out more, when here it's only somewhat lower, about 5 units lower, than you would expect on a normal distribution. So maybe the boxplot was actually exaggerating the outlierness of that lowest observation in the males. (Or, the fact that the default SAS line uses the standard deviation, which is inflated by the outlier, and so the line on the normal quantile plot for the males would be flatter on R's normal quantile plot and thus make the outlier stand out more.)

You know the drill: make a call and defend it. I don't mind whether you think that value is an outlier, causing trouble, or not, as long as you have decent reasons for your opinion.

Extra extra: I mentioned Mood's median test above, and I said that I suspected that it wouldn't be anywhere near significant either:

```
proc npar1way median;
  var heartrate;
  class gender;
```

```
                             The NPAR1WAY Procedure

      Median Scores (Number of Points Above Median) for Variable heartrate
                           Classified by Variable gender

                             Sum of      Expected       Std Dev          Mean
     gender        N         Scores      Under H0      Under H0         Score

     female       13       5.333333          6.50      1.231530      0.410256
     male         13       7.666667          6.50      1.231530      0.589744

                        Average scores were used for ties.
                             Median Two-Sample Test

                          Statistic              5.3333
                          Z                     -0.9473
                          One-Sided Pr <  Z      0.1717
                          Two-Sided Pr > |Z|     0.3435
                             Median One-Way Analysis

                          Chi-Square             0.8974
                          DF                          1
                          Pr > Chi-Square        0.3435
```

The P-value is a bit smaller, 0.3435, but still not anywhere near small enough to reject with. (Yes, I know I haven't talked about this in class yet, but the principle is the same as in R.)

5 Power analysis

5.1. We are planning a study to estimate a population mean. The population standard deviation is believed to be 20, and the population distribution is believed to be approximately normal. We will be testing the null hypothesis that the population mean is 100. Suppose the population mean is actually 110, and we want to determine how likely we are to (correctly) reject the null hypothesis in this case, using a two-sided (but one-sample) test with $\alpha = 0.05$.

This is the same situation as we investigated before with R.

(a) With a sample of size 30, what is the power of this test? You will need to specify the situation (`onesamplemeans`), the test `test=t`, the true mean (`mean`), the null mean `nullmean`, the population SD (`stddev`), the "total" sample size `ntotal` and the `power` that you are trying to find.

Solution: `proc power`, suitably modified. Guess or search for these:

```
proc power;
  onesamplemeans
  test=t
  mean=110
  nullmean=100
  stddev=20
  ntotal=30
  power=.;
```

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

                  Fixed Scenario Elements

            Distribution                Normal
            Method                       Exact
            Null Mean                      100
            Mean                           110
            Standard Deviation              20
            Total Sample Size               30
            Number of Sides                  2
            Alpha                         0.05
                      Computed Power

                          Power

                          0.754
```

The power is the same 0.754 as R (to 3 decimals rather than R's 7).

(b) Use SAS to find the sample size necessary to obtain a power of (i) 0.80 and (ii) 0.90. (This is doable in one step.) What sample sizes do you get?

Solution: Use the same SAS code as (b), and modify it to specify the power values (with a space between) and leave the sample size blank:

```
proc power;
  onesamplemeans
  test=t
  mean=110
  nullmean=100
  stddev=20
  ntotal=.
  power=0.80 0.90;
```

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

                   Fixed Scenario Elements

         Distribution                Normal
         Method                       Exact
         Null Mean                      100
         Mean                           110
         Standard Deviation              20
         Number of Sides                  2
         Alpha                         0.05
                  Computed N Total

                  Nominal    Actual         N
          Index     Power     Power     Total

              1       0.8     0.808        34
              2       0.9     0.900        44
```

Sample sizes of 34 (for power 0.80) and 44 (for power 0.90).

The first of these is what we got from R.[30]

5.2. You are designing an experiment to compare a treatment with a control. The response variable you are measuring, called y, is expected to have a mean of 40 in the treatment group and 30 in the control group. The standard deviation of y is expected to be about 16 in both groups. The researchers plan to use the Satterthwaite test, since they suspect the spreads of the treatment and control groups to be different.

Use SAS to answer the questions below. Bear in mind that you don't need any data; you just need to run the appropriate `proc` with the appropriate values.

(a) If you have funding to collect a sample size of 25 within each group, how likely are you to (correctly) reject a null hypothesis that the treatment and control means are equal, in favour of a one-sided alternative that the treatment mean is *higher*, at $\alpha = 0.05$?

Solution: Feed `proc power` the appropriate things. This would be `twosamplemeans` with `test=diff_satt`, `sides=1` (one-sided test), `stddev=15` (the same for both groups), `ntotal=50` (a sample size of 25 within each group, so 50 altogether). Leave `power` "missing", since that's what we're trying to find. There are no semicolons until right at the end, because this (as far as SAS is concerned) is all one line:

```
proc power;
  twosamplemeans
    test=diff_satt
    sides=1
    meandiff=10
    stddev=16
    ntotal=50
    power=.;
```

My `meandiff` is 10 because $40 - 30 = 10$. How much power do we have to detect this kind of alternative with this kind of sample size?

```
                      The POWER Procedure
  Two-Sample t Test for Mean Difference with Unequal Variances

                    Fixed Scenario Elements

                 Distribution                Normal
                 Method                       Exact
                 Number of Sides                  1
                 Mean Difference                 10
                 Standard Deviation              16
                 Total Sample Size               50
                 Null Difference                  0
                 Nominal Alpha                 0.05
                 Group 1 Weight                   1
                 Group 2 Weight                   1
                          Computed Power

                          Actual
                           Alpha    Power

                          0.0499    0.703
```

The power is 0.703.

There are actually several ways to do this. Another way is this:

```
proc power;
  twosamplemeans
    test=diff_satt
    sides=1
    groupmeans=40|30
    stddev=16
    ntotal=50
    power=.;
```

which gives the same result:

```
                     The POWER Procedure
    Two-Sample t Test for Mean Difference with Unequal Variances

                   Fixed Scenario Elements

             Distribution                Normal
             Method                       Exact
             Number of Sides                  1
             Group 1 Mean                    40
             Group 2 Mean                    30
             Standard Deviation              16
             Total Sample Size               50
             Null Difference                  0
             Nominal Alpha                 0.05
             Group 1 Weight                   1
             Group 2 Weight                   1
                       Computed Power

                       Actual
                        Alpha    Power

                       0.0499    0.703
```

If you have done something plausible-looking, I am good with it. You can also specify the group SDs separately via `groupstddevs` (which you would want to do if they are different from each other), and you can also specify the sample size within each group via `npergroup`. (If you do the latter in the next part, SAS will tell you how many subjects to use in each group rather than altogether.) My reference for these is `https://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_power_sect013.htm`.

About the `sides=1` thing, that being a number 1. When you are doing a test and getting a P-value, if your test is one-sided, you have to say *which* one side you want via `sides=U` or `sides=L`. You can also do that in `proc power`, but it seems not to be intuitive which way around it is. Sometimes it gives you a very small power, which is an indication that you were actually looking in the wrong tail. The safe way around this, *for `proc power` only*, is to use `sides=1`, which says that you're doing a one-sided test and you want the power *in the same direction as the effect*. In this case the true mean for group 1 was (expected to be) 40 and for group 2 was 30. The obvious thing to obtain power for here is the one-sided test with a null of equal means and an alternative that group 1's mean is bigger than group 2's, because the true mean for group 1 *is* bigger than for group 2. The advantage of using `sides=1` is that it will do this, whichever way around the groups are listed. That is to say, this works, as above:

```
proc power;
  twosamplemeans
    test=diff_satt
    sides=1
    meandiff=10
    stddev=16
    ntotal=50
    power=.;
```

```
                         The POWER Procedure
         Two-Sample t Test for Mean Difference with Unequal Variances

                        Fixed Scenario Elements

                  Distribution                Normal
                  Method                       Exact
                  Number of Sides                  1
                  Mean Difference                 10
                  Standard Deviation              16
                  Total Sample Size               50
                  Null Difference                  0
                  Nominal Alpha                 0.05
                  Group 1 Weight                   1
                  Group 2 Weight                   1
                            Computed Power

                            Actual
                             Alpha    Power

                            0.0499    0.703
```

but this also works, *without changing `sides`*:

```
proc power;
  twosamplemeans
    test=diff_satt
    sides=1
    meandiff=-10 /* switch the groups around */
    stddev=16
    ntotal=50
    power=.;
```

```
                         The POWER Procedure
         Two-Sample t Test for Mean Difference with Unequal Variances

                        Fixed Scenario Elements

                  Distribution                Normal
                  Method                       Exact
                  Number of Sides                  1
                  Mean Difference                -10
                  Standard Deviation              16
                  Total Sample Size               50
                  Null Difference                  0
                  Nominal Alpha                 0.05
                  Group 1 Weight                   1
                  Group 2 Weight                   1
                            Computed Power

                            Actual
                             Alpha    Power

                            0.0499    0.703
```

(b) Some extra research funding becomes available, and the principal investigator is curious how large a sample size would be needed to increase this power to 0.80, with everything else remaining the same. Find out how large a sample size would be needed, so that the principal investigator can determine whether the funding is available to support the collection of the larger sample.

Solution: Use the same code as before, but make two changes: set `ntotal` to missing, and fill in 0.80 for the power:

```
proc power;
  twosamplemeans
    test=diff_satt
    sides=1
    meandiff=10
    stddev=16
    ntotal=.
    power=0.80;
```

This gives

```
                        The POWER Procedure
           Two-Sample t Test for Mean Difference with Unequal Variances

                      Fixed Scenario Elements

                 Distribution                Normal
                 Method                       Exact
                 Number of Sides                  1
                 Mean Difference                 10
                 Standard Deviation              16
                 Nominal Power                  0.8
                 Null Difference                  0
                 Nominal Alpha                 0.05
                 Group 1 Weight                   1
                 Group 2 Weight                   1
                          Computed N Total

                      Actual    Actual        N
                       Alpha     Power    Total

                      0.0499     0.807       66
```

Read off “N Total” from the output, 66. Remember that this is the sample size needed *altogether*, so half of these should be in each group: that is, 33 in the treatment group and 33 in the control group. So, you tell the principal investigator that 33 subjects are needed in each group, and they can see whether they can get funding to support that.

As per the discussion above, this also works:

```
proc power;
  twosamplemeans
    test=diff_satt
    sides=1
    meandiff=10
    stddev=16
    npergroup=.  /* this was changed */
    power=0.80;
```

This gives

```
                      The POWER Procedure
     Two-Sample t Test for Mean Difference with Unequal Variances

                    Fixed Scenario Elements

               Distribution                Normal
               Method                       Exact
               Number of Sides                  1
               Mean Difference                 10
               Standard Deviation              16
               Nominal Power                  0.8
               Null Difference                  0
               Nominal Alpha                 0.05
                      Computed N per Group

                    Actual    Actual    N per
                     Alpha     Power    Group

                    0.0499     0.807       33
```

This gives you the number of observations required in each group, 33, so that you again need $33 + 33 = 66$ observations altogether.

You might notice that when SAS does a sample size calculation, it rounds up the sample size for you (unlike R, which gives you a decimal-number sample size that you have to round up yourself). SAS tells you about this using the words “nominal power” for the 0.8 you were aiming for, and the “actual power” of an experiment with 33 subjects in each group, which will be a little bigger than 0.8 (here it is 0.807). The implication is that 32 subjects per group gives power a little bit *less* than 0.8, and so we’re erring on the side of caution, so to speak, by rounding the sample size up. Likewise, with 33 subjects per group, the actual probability of a type I error (“actual alpha”) is a tiny bit less than the 0.05 we were aiming for (“nominal alpha”).

5.3. A study is being done to test whether a population mean is 100. The population standard deviation is believed to be about 10, and the population is believed to be normal in shape. Use SAS for this question.

(a) (4 marks) If a sample of size 35 is taken, and the population mean is actually 103, how likely is it that the null mean of 100 will be correctly rejected, against a two-sided alternative?

Solution: This is a one-sample *t*-test (there will be only one sample of observations to estimate one mean), so you need this kind of thing:

```
proc power;
  onesamplemeans
  test=t
  mean=103
  nullmean=100
  sttdev=10
  ntotal=35
  power=.;
```

Leave the power missing, since that's what you're trying to find.

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

                  Fixed Scenario Elements

            Distribution                Normal
            Method                       Exact
            Null Mean                      100
            Mean                           103
            Standard Deviation              10
            Total Sample Size               35
            Number of Sides                  2
            Alpha                         0.05
                      Computed Power

                          Power

                          0.407
```

You can also have `mean` be the difference 3 between the null and actual means, and omit `nullmean` completely. (You can also put in `sides=2`, but that is the default, so you don't need to worry about that.)

The power is 0.407, so that's the chance of correctly rejecting the null: not as big as you might like.

(b) (3 marks) Under the same conditions as the previous part, how big a sample size is needed to obtain power 0.75?

Solution: Two changes to your code: set `ntotal=.`, since you are trying to find a sample size, and put in the power value you are shooting for:

```
proc power;
  onesamplemeans
  test=t
  mean=103
  nullmean=100
  sttdev=10
  ntotal=.
```

```
        power=0.75;
```

This is the output:

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

                         Fixed Scenario Elements

                  Distribution                Normal
                  Method                       Exact
                  Null Mean                      100
                  Mean                           103
                  Standard Deviation              10
                  Nominal Power                 0.75
                  Number of Sides                  2
                  Alpha                         0.05
                           Computed N Total

                           Actual        N
                            Power    Total

                            0.755       80
```

A sample size of 80 is needed. (There is only one sample, so this is how big the one sample has to be.)

(c) (2 marks) SAS gave you an "actual power" that is slightly different from what you asked for. Explain briefly why it did that.

Solution: You can only get exactly the power you asked for by allowing a fractional sample size (which is what R does). But in real life the sample size must be a whole number, so you either get slightly more power than you wanted or slightly less, depending on whether you round up or down. The "safe" way to proceed is to round *up*, which gives you at least as much power as you asked for (usually a little more: here it's 0.755 vs. 0.75). SAS tells you (under `Actual Power`) exactly how much power you got. The line `Nominal Power` tells you what power you were aiming for.

The implication of this is that a sample size of 80 would give you power slightly greater than 0.75 and a sample size of 79 would give you power slightly *less* than 0.75. When we were working with R, this is why I told you always to round sample sizes up, even if the answer was something like 79.1: rounding down would give you power slightly less than your target.

"Because the sample size has to be an integer" is only one point because it's not enough insight. The second point is for noting the implication of this: that the actual power you get is either a bit too much or not quite enough.

(d) (3 marks) After further study, the researchers found that the population SD is about 12 rather than about 10. Calculate the sample size now required to get a power of 0.75, and explain briefly why the change in results from part (b) is not surprising.

Solution: For the calculation, repeat the previous part but with the `stddev` changed from 10 to 12, using another dose of copy and paste (or editing of what you had before):

```
proc power;
  onesamplemeans
  test=t
  mean=103
  nullmean=100
  sttdev=12
  ntotal=.
  power=0.75;
```

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

                     Fixed Scenario Elements

               Distribution                Normal
               Method                       Exact
               Null Mean                      100
               Mean                           103
               Standard Deviation              12
               Nominal Power                 0.75
               Number of Sides                  2
               Alpha                         0.05
                        Computed N Total

                        Actual        N
                         Power    Total

                         0.750      113
```

The sample size needed is now 113, about 40% bigger than before.

This makes sense because if the population SD is bigger, the data we get are likely to be more variable, and thus we will have a harder time rejecting the null hypothesis then before, all else being equal. (A little more precisely, a larger population SD will tend to mean a larger sample SD, and thus a *smaller* test statistic, not so far away from zero, and thus a less small P-value.) In order to keep the power the same, without changing anything else, we have to take a bigger sample size.

Extra: I was actually surprised it came out this much bigger, but it's not always easy to predict what will happen in a power calculation. If we had been thinking about confidence interval length instead, it would have been easier: increasing the SD by a factor of 1.2 (as here) would also increase the length of the confidence interval by a factor of 1.2, all else equal. So if we wanted to keep the confidence interval length the same, we'd have to increase the sample size by a factor of $1.2^2 = 1.44$. So maybe the 40% increase in sample size in our power calculation ought not to have surprised me.

Extra extra: it looks as if we hit our target power of 0.75 exactly this time, which we may have been lucky enough to do, but I think it is more likely that the "actual power" is something like 0.7501, a teeny bit bigger than 0.75, if enough decimal places were shown.

6 Sign test

6.1. In 1999, the National Basketball Association introduced some rule changes that were intended to open up the game and increase scoring. You can read about them here: `https://www.nytimes.com/1999/10/`

31/sports/1999-2000-nba-preview-playing-by-the-new-rules.html. I especially like the phrase "borderline thuggery". Before 1999 the average number of points per game (both teams together) was 183.2. There were 25 games played during the period December 10–12, 1999 (after the rule changes were implemented), which you can take as being a random sample of "all possible games" played under the new rules. The data are in http://www.utsc.utoronto.ca/~butler/c32/nbapoints.csv.

You previously analyzed these data with R. This analysis will be similar but slightly different.

(a) (2 marks) Read the data into SAS and display the data set.

Solution: This is the usual kind of thing, with a .csv file:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/nbapoints.csv";

proc import
  datafile=myurl
  out=nba
  dbms=csv
  replace;
  getnames=yes;

proc print;
```

with output

```
         Obs    Date                                     Points

           1                  19991210T000000+0000          196
           2                  19991210T000000+0000          198
           3                  19991210T000000+0000          205
           4                  19991210T000000+0000          163
           5                  19991210T000000+0000          184
           6                  19991210T000000+0000          224
           7                  19991210T000000+0000          206
           8                  19991210T000000+0000          190
           9                  19991210T000000+0000          140
          10                  19991210T000000+0000          204
          11                  19991211T000000+0000          200
          12                  19991211T000000+0000          190
          13                  19991211T000000+0000          195
          14                  19991211T000000+0000          180
          15                  19991211T000000+0000          200
          16                  19991211T000000+0000          180
          17                  19991211T000000+0000          198
          18                  19991211T000000+0000          243
          19                  19991211T000000+0000          235
          20                  19991211T000000+0000          200
          21                  19991211T000000+0000          188
          22                  19991211T000000+0000          197
          23                  19991212T000000+0000          191
          24                  19991212T000000+0000          194
          25                  19991212T000000+0000          196
```

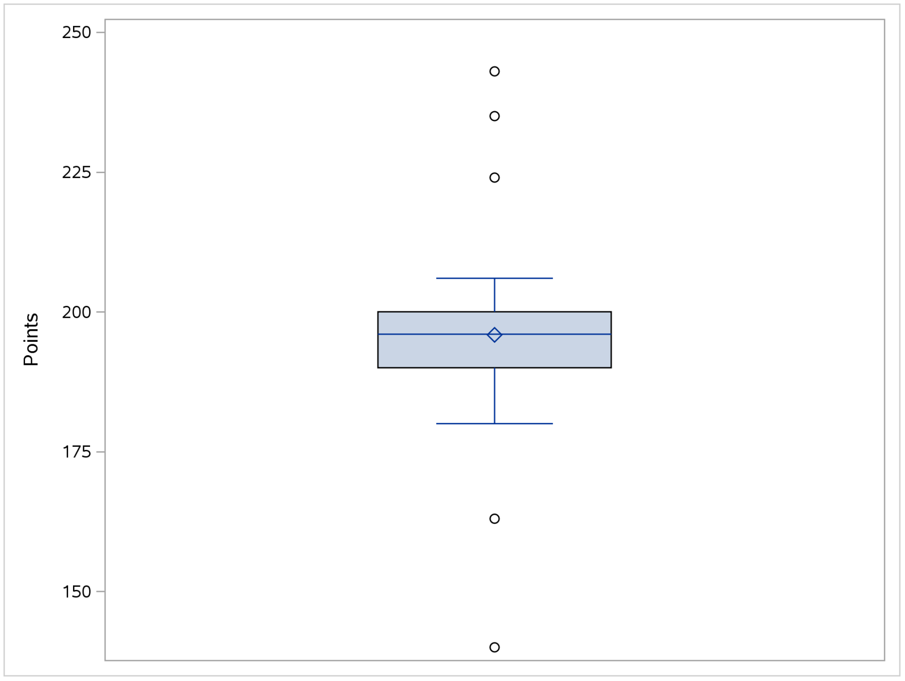
That column `Date` is rather odd-looking, but those are SAS date-times, with year-month-day, a T for Time, then a six-digit time (hours-minutes-seconds, midnight here) with time zone (this one being UTC).

We're not going to be doing anything with `Date`, but the `Points` column looks believable.

(b) (2 marks) Make a boxplot of the numbers of points. (This is a boxplot without any categorical variable, so you should get just one boxplot, rather than several side by side.)

Solution: There's a reason I asked for a boxplot, which I'll get to shortly, but first:

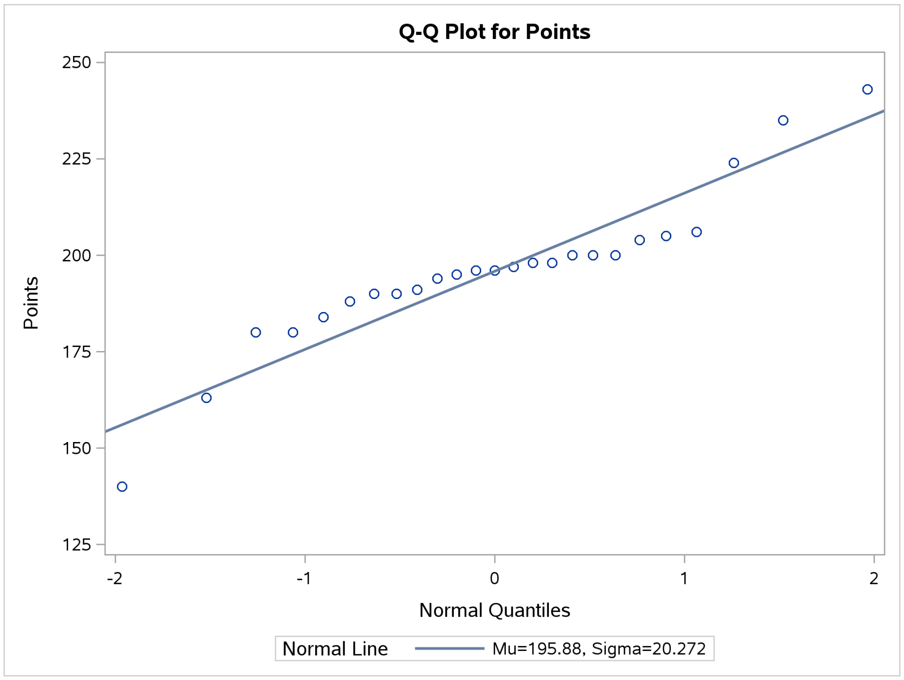
```
proc sgplot;
  vbox Points;
```



This clearly shows the outliers at both ends (jumping the gun a bit, since that's the answer to the next part).

Extra: the reason I didn't ask for a normal quantile plot here was that I suspected that the high and low outliers wouldn't show up so clearly. Let's draw it and see whether I was right:

```
proc univariate noprint;
  qqplot Points / normal(mu=est sigma=est);
```



Yeah, normality doesn't look so bad here, as I suspected.

The reason for that is that SAS estimates σ using the sample standard deviation. Here, though, this is inflated because of the outliers at both ends, so the standard deviation is larger than it ought to be (those outliers are a long way from the mean). This in turn makes the line steeper than it ought to be, which makes it appear to go more nearly through the points.

If you look carefully at this plot, it has a kind of S-bend, in that the points in the middle are less steep than the line is. But I didn't want you grappling with that, so I had you draw a boxplot instead, where the outliers are a lot clearer.

The way to fix up the normal quantile plot, as you might have guessed, is to estimate `mu` and `sigma` using the median and IQR. But first we have to find those:

```
proc means median qrange;
  var Points;
```

```
                    The MEANS Procedure

                 Analysis Variable : Points

                                  Quartile
                   Median            Range
               ----------------------------
                196.0000000      10.0000000
               ----------------------------
```

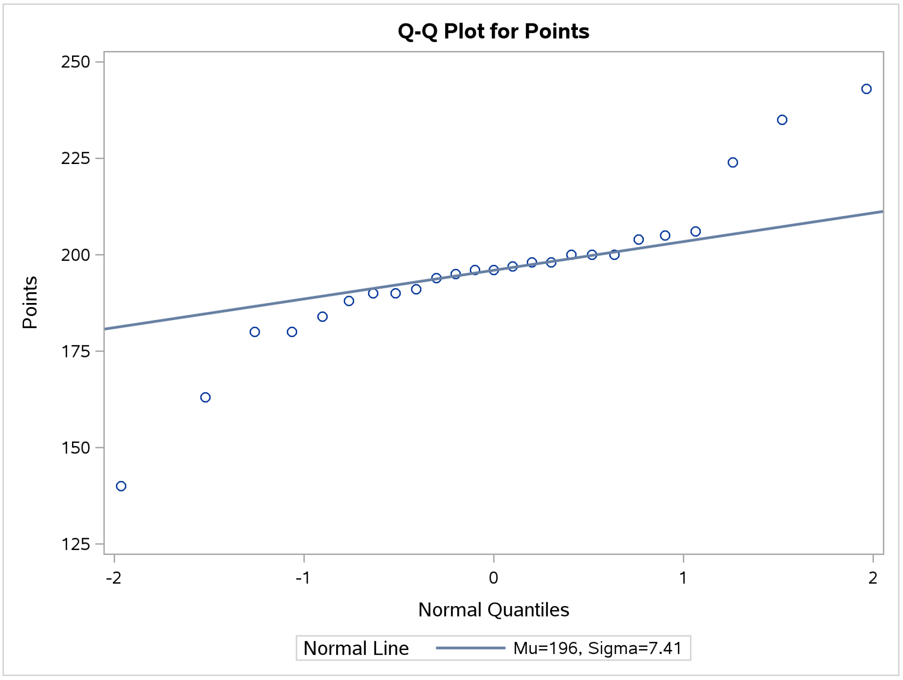
The estimate of `sigma` is 10 divided by 1.35:

```
10/1.35

## [1] 7.407407
```

and the estimate of `mu` is the median, so:

```
      proc univariate noprint;
        qqplot Points / normal(mu=196 sigma=7.41);
```

and thus



This shows the outliers much more clearly, and the line does a much better job of going through the bulk of the points. It also looks a lot more like the R normal quantile plot.

I'm inclined to say that we should *never* use SAS's normal quantile plot with `mu=est sigma=est`. That would perhaps be too much of a blanket statement, but the reason I'm thinking this way is that when the normal distribution is inappropriate in the usual ways, such as skewness, outliers, or long tails, the mean and standard deviation are also inappropriate to measure centre and spread and will typically make the normal quantile plot look less bad than it should (as compared to basing our estimation on the median and IQR), in exactly the cases when we want it to look bad (because normality is under question).

(c) (2 marks) What do you notice on your boxplot?

Solution: As I indicated before, outliers at both ends. (The two marks are (i) outliers and (ii) at the high *and* low ends.) This is an indication of long tails rather than skewness (in that case you would see outliers at only one end, typically the end with the longer whisker.)

Note that the mean and median are almost exactly equal to each other. The effect of the outliers is not to distort the mean (as they would if they were only at one end), but to distort the standard deviation by making it bigger (as I explained in the normal quantile plot discussion above).

The outliers imply that we should be doing a sign test rather than a t-test, but I get ahead of myself again.

(d) (2 marks) Run a procedure that will get you both a sign test and a t-test for the appropriate null mean/median.

Solution: `proc univariate`, in the Tests for Location section, will get both tests. The appropriate null value is the pre-1999 average (which I think is actually a mean, but we ignore that):

```
proc univariate location=183.2;
  var Points;
```

```
                    The UNIVARIATE Procedure
                       Variable:  Points

                 Tests for Location: Mu0=183.2

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

        Student's t    t  3.127507    Pr > |t|    0.0046
        Sign           M       8.5    Pr >= |M|   0.0009
        Signed Rank    S     116.5    Pr >= |S|   0.0006
```

That's all I was after here. As an extra, though, note that this one is unusual in that the P-value for the sign test is *smaller* than the one for the t-test. (That may be tied in with my comment earlier that this might be one of those cases where the sign test is more powerful than the t-test.)

(e) (3 marks) Which test is more appropriate? What do you conclude from it, bearing in mind the NBA's aims with the rule changes? (The answer to this should be the same as you got with R when you used this data set before. Feel free to use your previous results to check your work here.)

Solution: As we concluded when we did this with R, the appropriate test is the sign test because of the outliers. (The t-test will not work because we don't have a large enough sample size to overcome the effect of the outliers.) One point for that.

We should go back to the question and remind ourselves of the NBA's aim here: it was to *increase* the average number of points scored in a game. So we should be using a one-sided test. But this one is two-sided, so we have to turn it into a one-sided one. We need to check first whether we are on the correct side. From the boxplot, the sample median is more than 190 (my calculations elsewhere found it to be 196), definitely more than the previous value of 183.2. So we are justified in halving the two-sided P-value, getting (to this accuracy) $0.0009/2 = 0.00045$. One point altogether for making a case for a one-sided test and for providing a reasonable justification for halving the two-sided P-value. (Half of 0.0009 is to this accuracy the same P-value that `smmr` gave in R.)

The last point is for saying that we *do* have evidence that the median number of points in a game has increased. (If you correctly interpret the two-sided test: that is, if you conclude from it that the median number of points in a game has *changed*, expect to get two out of three for this part, since this was not what we were trying to demonstrate.)

7 Matched pairs

7.1. Can students throw a baseball farther than a softball? A statistics class, containing 24 students, went out to a football field to try to answer this question. Each student warmed up and then threw each type of ball as far as they could. The order of ball types was randomized: some students threw the baseball first, and some threw the softball first. (A softball is bigger than a baseball, so we might expect that a softball would be harder to throw a long way than a baseball.) The data are in `http://www.utsc.utoronto.ca/~butler/c32/throw.txt` in three columns: the first is a number identifying the student, the second is the distance thrown with the baseball (in yards) and the third is the distance thrown with the softball (also in yards).

(a) Read the data into SAS. *There are no column headers*, which you'll need to take into account.

Solution: The file extension suggests that the data values are separated by spaces, which is correct, but there are *no* variable names, so `getnames=no`:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/throw.txt";

proc import
  datafile=myurl
  dbms=dlm
  out=throw
  replace;
  delimiter=' ';
  getnames=no;
```

There are no variable names, so SAS had to invent some:

```
proc print;
```

```
             Obs          VAR1          VAR2          VAR3

               1             1            65            57
               2             2            90            58
               3             3            75            66
               4             4            73            61
               5             5            79            65
               6             6            68            56
               7             7            58            53
               8             8            41            41
               9             9            56            44
              10            10            70            65
              11            11            64            57
              12            12            62            60
              13            13            73            55
              14            14            50            53
              15            15            63            54
              16            16            48            42
              17            17            34            32
              18            18            49            48
              19            19            48            45
              20            20            68            67
              21            21            30            27
              22            22            26            25
              23            23            28            25
              24            24            26            31
```

The data values look OK, and there are correctly 24 rows. The column names are `VAR1`, the student IDs, `VAR2`, the distance thrown with a baseball, and `VAR3`, the distance thrown with a softball.

(b) Calculate a column of differences, baseball minus softball.

Solution: Remember how SAS wants you to do this: create a new data set, copy in everything from the previous one, and *then* create your new variable. Don't forget to use SAS's variable names:

```
data throw2;
  set throw;
  diff=VAR2-VAR3;
```

and for completeness check that it worked, bearing in mind that the most-recently created data set is the new one, `throw2`, so this will do the right thing:

```
proc print;
```

```
         Obs            VAR1            VAR2            VAR3    diff

           1               1              65              57       8
           2               2              90              58      32
           3               3              75              66       9
           4               4              73              61      12
           5               5              79              65      14
           6               6              68              56      12
           7               7              58              53       5
           8               8              41              41       0
           9               9              56              44      12
          10              10              70              65       5
          11              11              64              57       7
          12              12              62              60       2
          13              13              73              55      18
          14              14              50              53      -3
          15              15              63              54       9
          16              16              48              42       6
          17              17              34              32       2
          18              18              49              48       1
          19              19              48              45       3
          20              20              68              67       1
          21              21              30              27       3
          22              22              26              25       1
          23              23              28              25       3
          24              24              26              31      -5
```

which it did.

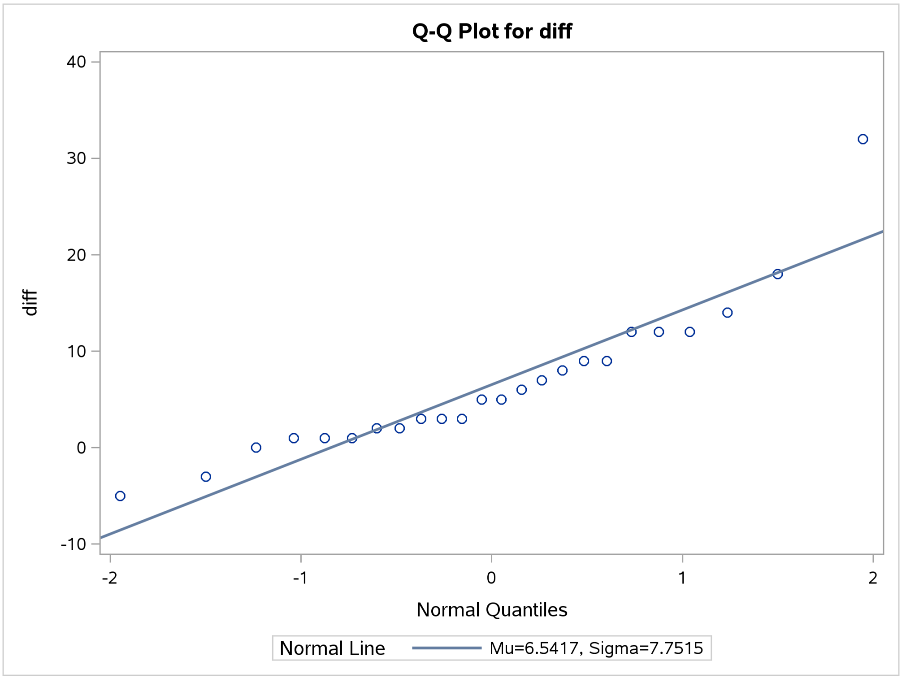
(c) Make a normal quantile plot of the differences. On your plot, add a line (using a μ and σ estimated from the data). What do you conclude from the plot, and thus why would a sign test be more appropriate than a matched-pairs t-test?

Solution: This kind of thing:

```
proc univariate noprint;
```

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

with result



These differences are mostly normal, except for the outlier at the upper end. The outlier makes us doubt normality, which is assumed for a *t*-test, so a sign test would be more appropriate.

You could also reasonably see a curve in the normal quantile plot, with the lowest values being a bit too high and the outlier at the top end. I'm not sure, myself, that those low-end values are all that bunched-up, but this is a reasonable way of looking at the plot, and leads to the same conclusion.

I have made noises elsewhere about not using this line. Does it look much different if we use median and IQR?

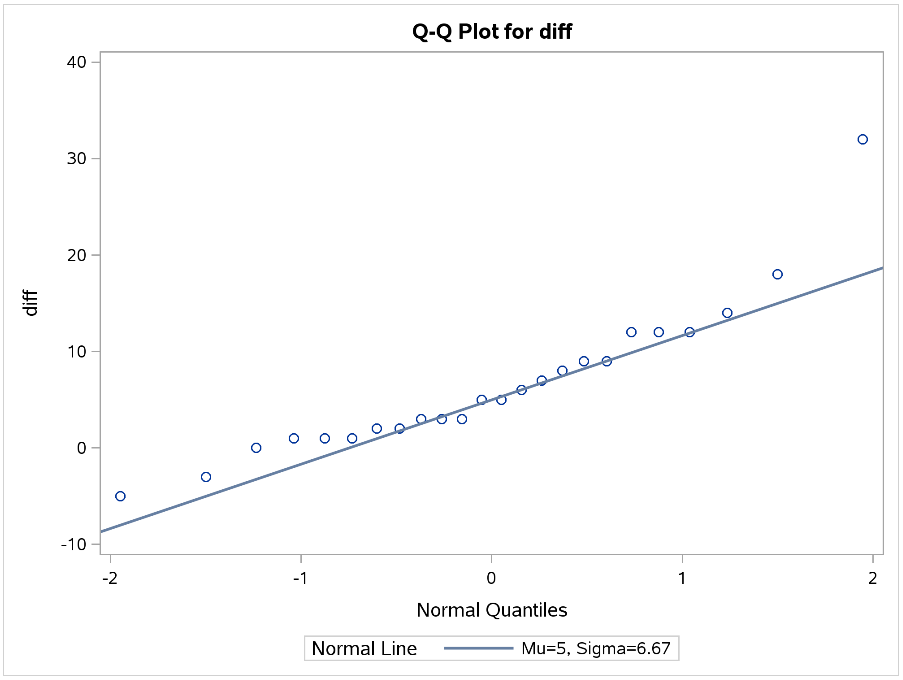
```
      proc means median qrange;
        var diff;
```

```
                        The MEANS Procedure

                      Analysis Variable : diff

                                          Quartile
                        Median               Range
                  ----------------------------
                     5.0000000           9.0000000
                  ----------------------------
```

```
9/1.35

## [1] 6.666667
```

and so:

```
proc univariate noprint;
  qqplot diff / normal(mu=5 sigma=6.67);
```



Not much different. Certainly, I think the best conclusion is that there is an outlier at the top end.

(d) Think about how you would use a sign test in this matched-pairs situation. Run an appropriate sign test in SAS, bearing in mind the null and alternative hypotheses that you wish to test. What do you conclude, in the context of the data?

Solution: In the matched-pairs context, our null hypothesis is that there is no difference between how far students can throw a baseball and a softball: that is, that the median difference is zero. We wanted to see whether students can throw a baseball further on average than a softball: that is, whether the median difference is *greater* than zero (the way around I calculated it: if you did softball minus baseball, the median difference would be *less* than zero).

Thus the SAS code is something like this:

```
proc univariate mu0=0;
  var diff;
```

This will get us, remember, a two-sided test:

```
                        The UNIVARIATE Procedure
                            Variable:  diff

                       Tests for Location: Mu0=0

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

           Student's t    t  4.134381    Pr > |t|    0.0004
           Sign           M       9.5    Pr >= |M|   <.0001
           Signed Rank    S     119.5    Pr >= |S|   <.0001
```

The two-sided P-value is less than 0.0001. But we wanted a one-sided P-value, for testing that the median difference is *greater* than zero. So we ought first to check that the median difference in the sample is greater than zero, which is also on the `proc univariate` output:

```
                          Basic Statistical Measures

                Location                    Variability

            Mean     6.541667     Std Deviation            7.75146
            Median   5.000000     Variance                60.08514
            Mode     1.000000     Range                   37.00000
                                  Interquartile Range      9.00000

      Note: The mode displayed is the smallest of 3 modes with a count of 3.
```

The median difference is 5, so we are "on the correct side", and our one-sided P-value is half the two-sided one, less than 0.00005. This is definitely small enough to reject the null with, and we can conclude that students really can throw a baseball farther than a softball.

When we did this in R, we got a P-value of 0.000033, which is consistent with this one. (You might argue that "less than 0.00005" is as accurate as you need to be, since it points to a really small P-value; knowing how much smaller than that it is is not really very informative.)

For a complete answer, you need in your discussion to say that SAS's P-value is two-sided and we need a one-sided one. Simply halving the two-sided one is not the best (you really ought to convince yourself that you are "on the correct side"), but is acceptable. An answer simply using SAS's P-value, even though "less than 0.0001" is the right answer, is not the right answer for the right reason, and so is incomplete.

(e) Obtain a 95% confidence interval for the median. Compare with what you got before from R.

Solution: The magic word is `cipctldf`:

```
proc univariate cipctldf;
  var diff;
```

```
                     The UNIVARIATE Procedure
                         Variable:  diff

                     Quantiles (Definition 5)

                      Level         Quantile

                      100% Max          32.0
                      99%               32.0
                      95%               18.0
                      90%               14.0
                      75% Q3            10.5
                      50% Median         5.0
                      25% Q1             1.5
                      10%                0.0
                      5%                -3.0
                      1%                -5.0
                      0% Min            -5.0

                    Quantiles (Definition 5)

              95% Confidence Limits    -------Order Statistics-------
Level           Distribution Free      LCL Rank  UCL Rank    Coverage

100% Max
99%               .                .         .         .           .
95%              14               32        22        24       59.21
90%              12               32        20        24       83.52
75% Q3            6               18        14        23       96.96
50% Median        2                9         8        18       95.67
25% Q1           -3                3         2        11       96.96
10%              -5                1         1         5       83.52
5%               -5                0         1         3       59.21
1%                .                .         .         .           .
0% Min
```

The confidence interval for the median difference is from 2 to 9. This is how much further, on average, students can throw a baseball than a softball. (This is the same interval that came out of R.)

7.2. Previously, we looked at a parking survey designed to address whether men or women were better at parallel parking. Let's revisit these data, and see what might be a better test that the two-sample *t*-test we did before. The data were in `http://www.utsc.utoronto.ca/~butler/c32/parking.xlsx`.

(a) Now we'll do Mood's median test in SAS (which has it built in). First read the data into SAS and summarize the values.

Solution: This is completely copied from what I did before:[31]

```
proc import
  datafile='/home/ken/parking.xlsx'
  dbms=xlsx
  out=mydata
  replace;
  sheet=Sheet2;
  getnames=yes;

proc means;
```

```
  var distance;
  class gender;
```

```
                                The MEANS Procedure

                         Analysis Variable : distance distance

              N
 gender     Obs     N           Mean         Std Dev         Minimum         Maximum
 -----------------------------------------------------------------------------
 female      47    47      9.3085106       5.3258529       2.0000000      25.0000000

 male        46    46     11.1413043       7.7729324       0.5000000      48.0000000
 -----------------------------------------------------------------------------
```

The same number of males and females that we had before, and a slightly smaller mean for the females. Or, find the median and quartiles and compare with the boxplots:

```
proc means q1 median q3;
  var distance;
  class gender;
```

```
                                The MEANS Procedure

                         Analysis Variable : distance distance

                         N           Lower                            Upper
         gender        Obs        Quartile          Median         Quartile
         -------------------------------------------------------------
         female         47       5.0000000       8.5000000       13.0000000

         male           46       7.0000000      10.0000000       14.0000000
         -------------------------------------------------------------
```

Bearing in mind that the SAS and R definitions of quartiles do differ, so you may not get *exactly* the same thing, these appear to be the same as the boxplots.

(b) Run Mood's median test. What do you conclude here, and do you get the same result as R (either the way you did it or the way `smmr` does it)?

Solution: This is `proc npar1way` with option `median` (*not* `mood`!):

```
proc npar1way median;
  var distance;
  class gender;
```

```
                               The NPAR1WAY Procedure

       Median Scores (Number of Points Above Median) for Variable distance
                            Classified by Variable gender

                            Sum of      Expected       Std Dev          Mean
     gender         N       Scores      Under H0      Under H0         Score

     male          46    25.666667     22.752688      2.353169      0.557971
     female        47    20.333333     23.247312      2.353169      0.432624

                          Average scores were used for ties.
```

```
                    Median Two-Sample Test

               Statistic             25.6667
               Z                      1.2383
               One-Sided Pr >  Z      0.1078
               Two-Sided Pr > |Z|     0.2156
                  Median One-Way Analysis

               Chi-Square             1.5334
               DF                          1
               Pr > Chi-Square        0.2156
```

This gives the same *conclusion* as before (no difference between the medians for males and females), but a different P-value (look in the Median One-way Analysis at the end of the output). I think the difference is yet another way of handling those observations that are exactly equal to 9. If you go back up to the table of median scores at the top of the output, the Sum of Scores column is the key.

If there are no observations exactly equal to the overall median, this will be the numbers in our `FALSE` columns above: the number of values *above* the overall median. If there are values equal to the overall median, something else happens. In this case, there are 93 data values altogether. 43 of them are strictly less than the median, 44 are strictly greater and the other 6 are exactly equal to the median. If those values exactly equal to the median were in fact different from each other, they would have ranks $44, 45, \ldots 49$ from the bottom. The median would have rank $(93+1)/2 = 47$, so the first four of these are less than or equal to the median, and the last two are strictly greater.

Now, we have two groups, so if those observations had actually been different from each other, we don't know which ones of them would have been greater than the median and which $\leq$. So we pretend that $2/6 = 1/3$ of them were greater than the median in each group.

There were two male observations equal to 9, so SAS pretends that $2(1/3) = 2/3 = 0.67$ of them were greater than equal to 9, giving a total of $25 + 0.67 = 25.67$. There were four female observations equal to 9, and 19 strictly greater, giving a total of $19 + 4(1/3) = 20.33$. Those match the sums of scores in the output.

8 Analysis of variance

8.1. How do you learn and remember somebody's name when you meet them for the first time? Psychologists at Lancaster University[32] evaluated three methods of name retrieval. 139 students were randomly divided into three groups. Each group used a different method to learn the names of the other students in their group:

simple In the simple name game:

- the first student states their name
- the second student states their name and the name of the first student,
- the third student states their name and the names of the first two students,

and so on.

elaborate The elaborate name works the same way as the simple name game, except that each student states their name and also their favourite activity, and students had to repeat the favourite activities as well as the names of the students before them. (The idea was to see whether this made the names easier to remember.)

pairwise Students are divided into pairs, and each student must learn the name of the other student in their pair well enough to introduce that student to the group (of students doing pairwise introductions).

One year after this, all the participants in the study were sent pictures of all the other students in their group, and asked to name those students. The response variable was the percentage of names correctly recalled.

The data are in `https://www.utsc.utoronto.ca/~butler/c32/namegame.txt`.

(a) (3 marks) Read in the data, and determine how many subjects were given each different way of learning names.

Solution: First, read in the data. *Look at the data file before that* to see that the data values are separated by single spaces, and the first line is variable names as you would expect.

```
filename myurl url "https://www.utsc.utoronto.ca/~butler/c32/namegame.txt";
proc import
  datafile=myurl
  out=names
  dbms=dlm
  replace;
  getnames=yes;
  delimiter=" ";
```

You should use `proc print` *for yourself* to check that you have 139 rows of data with the right variables, but *don't* think about handing that in!

To get the number of students in each group, the easy way is to use `proc means` (which will get you the mean recall for each group as well, but no problem about that):

```
proc means;
  var recall;
  class game;
```

```
                        The MEANS Procedure

                    Analysis Variable : recall

          N
game     Obs     N           Mean        Std Dev        Minimum        Maximum
-------------------------------------------------------------------------------
elabor    42    42     26.2142857     23.7019457              0     86.0000000

pairwi    47    47     15.1276596     15.7032495              0     66.0000000

simple    50    50     30.6400000     20.0354380              0     99.0000000
-------------------------------------------------------------------------------
```

Another way of doing this, which is more in the spirit of just counting the number of observations in each group, is to use `proc freq`:

```
proc freq;
  tables game;
```

with output

```
                              The FREQ Procedure

                                                   Cumulative    Cumulative
           game      Frequency     Percent     Frequency      Percent
           -----------------------------------------------------------
           elabor          42       30.22            42        30.22
           pairwi          47       33.81            89        64.03
           simple          50       35.97           139       100.00
```

I'm good with either of these. Whichever way you go, you *need to say* this:

There are 42 students in the `elaborate` group, 47 in the `pairwise` group, and 50 in the `simple` group.

Expect to lose a mark if you don't.

Two extras:

extra 1: you might have wondered why some of the game names got shortened. This is the issue that we ran into before (I think) about reading in text: SAS by default uses the first 20 lines of the data file to determine how long the game names are, and the longest one it sees there is `simple`, six letters, and so it thinks that *all* the game names are no more than six letters long. (The data are in order, `simple` first.) I could have gotten around this by putting the `elaborate` data first in the file, but I thought you should see this. The other way to get around this is to use `guessingrows`, as we saw before.)

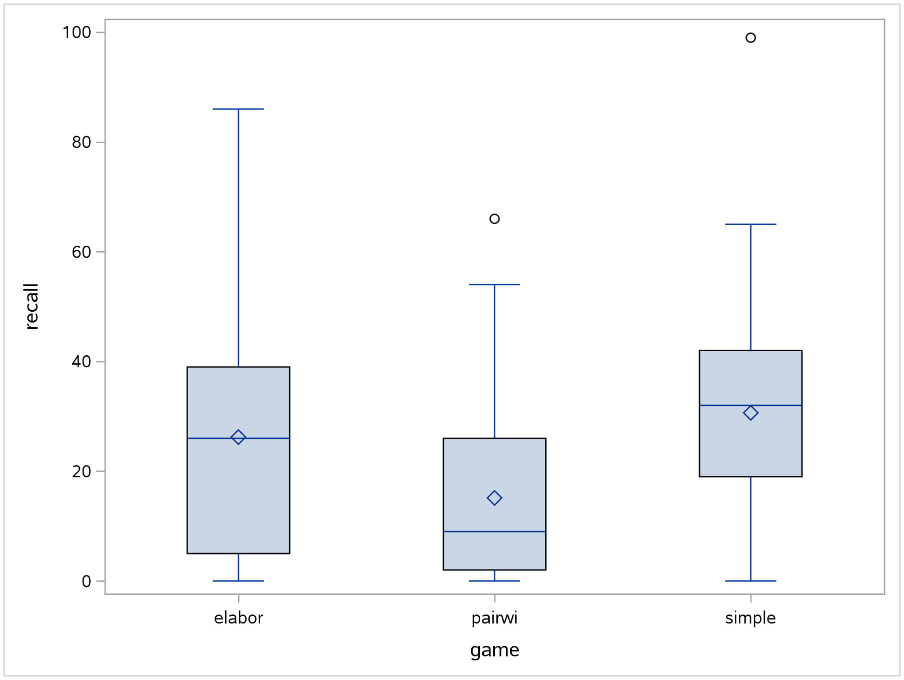
extra 2: why are the groups different sizes? It's customary, if you are comparing a number of groups, to have the same number of observations in each. This is to maximize the power of the ANOVA that you will shortly be doing: if you have the same number of observations in each group, you maximize your chance of rejecting the null hypothesis (that all the means are the same), *if* the population means are really not all the same.

I'm guessing (since I don't have any more information than I gave you here) that the groups started out the same size, say 50 students in each, and some of the students dropped out. When you're using humans in a study, you have to get their "informed consent", meaning that they understand what they will be doing and agree to do it, and a standard piece of the consent form is that participants can withdraw from the study at any time.

(b) (2 marks) Make a suitable plot of this data set.

Solution: One quantitative and one categorical variable, so a boxplot:

```
proc sgplot;
  vbox recall / category=game;
```



Comment on this is coming up later, but I think you should be feeling some nagging doubts at this point.

(c) (3 marks) Run a suitable analysis of variance, with Tukey, and display all the (text) results (ie., not the graphs, if you get any).

Solution: My idea here is to have you get the Tukey results first, and later decide whether you need them:

```
proc anova;
  class game;
  model recall=game;
  means game / tukey;
```

```
                             The ANOVA Procedure

                           Class Level Information

                  Class         Levels    Values

                  game               3    elabor pairwi simple
                     Number of Observations Read         139
                     Number of Observations Used         139
                             The ANOVA Procedure

                         Dependent Variable: recall

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

  Model                        2       6109.71410      3054.85705       7.69    0.0007

  Error                      136      54045.82547      397.39578

  Corrected Total            138      60155.53957
```

```
             R-Square     Coeff Var      Root MSE    recall Mean

             0.101565      82.86290      19.93479       24.05755
Source                       DF       Anova SS    Mean Square   F Value   Pr > F

game                          2    6109.714097    3054.857049      7.69   0.0007
                                  The ANOVA Procedure

              Tukey's Studentized Range (HSD) Test for recall

       NOTE: This test controls the Type I experimentwise error rate.


                 Alpha                                   0.05
                 Error Degrees of Freedom                 136
                 Error Mean Square                   397.3958
                 Critical Value of Studentized Range  3.35119
      Comparisons significant at the 0.05 level are indicated by ***.


                                Difference
               game                Between     Simultaneous 95%
            Comparison               Means    Confidence Limits

          simple - elabor            4.426      -5.462   14.313
          simple - pairwi           15.512       5.915   25.110  ***
          elabor - simple           -4.426     -14.313    5.462
          elabor - pairwi           11.087       1.056   21.117  ***
          pairwi - simple          -15.512     -25.110   -5.915  ***
          pairwi - elabor          -11.087     -21.117   -1.056  ***
```

(d) (3 marks) Run a suitable Mood's median test (you don't need to do Tukey or anything equivalent to it), displaying the printed results.

Solution: `proc npar1way median` with the same `var` and `class` that you would use in `proc means`:

```
proc npar1way median;
  var recall;
  class game;
```

with output

```
                             The NPAR1WAY Procedure

        Median Scores (Number of Points Above Median) for Variable recall
                             Classified by Variable game

                           Sum of      Expected       Std Dev          Mean
   game          N         Scores      Under H0      Under H0         Score

   simple       50           34.0     24.820144      2.839221      0.680000
   elabor       42           23.0     20.848921      2.716625      0.547619
   pairwi       47           12.0     23.330935      2.798737      0.255319

                        Average scores were used for ties.
```

```
                       Median One-Way Analysis

                      Chi-Square         17.9797
                      DF                       2
                      Pr > Chi-Square     0.0001
```

(e) (4 marks) Which test do you prefer, ANOVA or Mood's median test, for these data? Explain briefly. For your preferred test, give as complete a conclusion as your output will permit.

Solution: You can justify doing either test here, I think, but the starting point should be the boxplots.

I look at the boxplots and see right-skewedness in all three groups: the two with upper outliers and the one with a long right tail (`elaborate`). This makes me doubt the validity of the ANOVA, because these distributions are not that close to normal. That would be a good reason for preferring Mood's median test.

That said, however, you can also note the sample sizes: over 40 in each group, so we can expect a lot of help from the Central Limit Theorem: the distributions within each group do not have to be all that normal for ANOVA to be OK. So you can make the case that what we have here is "moderate skewness" of the kind that the Central Limit Theorem with these sample sizes can overcome, and therefore that the regular ANOVA is OK. (The data set came from the ANOVA section of a textbook, so clearly the authors thought that ANOVA was defensible for these data.)

So, make a choice for a good reason (two marks for that), and then follow through with your choice (the remaining two marks).

If you went with Mood's median test, you have it easier, because there is only one test to interpret. The P-value for that is 0.0001, so there is strong evidence that the group medians are *not* all equal: that is, it makes a difference to average (median) recall which name memorization game was used.

We haven't officially done a multiple-comparisons test for Mood's median test (to determine which groups differ from which in median recall). I have a suggestion below, but I didn't ask you to do it, so for the question this is as far as you need to go.

If you were, all things considered, happy with the ANOVA, then you have two things to do: (i) interpret the F-test, (ii) interpret the Tukey (if appropriate). One point for doing each of those correctly.

The P-value for the F-test is 0.0007 (from the `game` line in the second table, though for a one-way analysis like this, from the `Model` line of the first table is also good.) This means that the mean recalls for the three games (memorization methods) are not all the same, and thus we need to look at the Tukey to find out which are better or worse.

This time, the Tukey output is a lot more like R's, with each pair of groups being compared. (This is because there are not the same number of observations in each group, so SAS does what is called Tukey-Kramer instead: the comparison between each pair of groups depends on how many observations there are in each group.) So you interpret it the same way as R's: all the comparisons involving `pairwise` are significant, and the only non-significant one is between `elaborate` and `simple`. If you disentangle which way around the means are (or go back to your table of means, if you made one), this means that the `pairwise` game leads to significantly *worse* recall on average than either of the other two games, which are not significantly different from each other.

Each pair of groups is compared both ways around, so there are six comparisons rather than only three.

In the usual way that SAS does it, `simple` and `elaborate` would be joined by AAAAAA, and `pairwise` would have a B next to it and be off by itself.

I'm guessing that `pairwise` was something like a control group here, and the study was intended to show the superiority of the `simple` and `elaborate` ways of learning names, with the aim of finding out whether the `elaborate` version was better for recall (it wasn't). In D29, we learn about "contrasts", which would provide a way of testing precisely that.

Extra: I think I mentioned elsewhere that a way to do something Tukey-like for Mood's median test is to do the test for *pairs* of groups, and then adjust the P-values to allow for having done three tests at once. This works naturally in SAS using a `where` line and a lot of copying and pasting:

```
proc npar1way median;
  where game='elabor' | game='pairwi';
  var recall;
  class game;

proc npar1way median;
  where game='elabor' | game='simple';
  var recall;
  class game;

proc npar1way median;
  where game='pairwi' | game='simple';
  var recall;
  class game;
```

I thought first that using the full names for the games would be OK, because SAS would only compare the first six letters to what's in its data set (as far as SAS is concerned, the games are only six letters long). But I was wrong. I have to truncate them to six letters myself.

```
                          The NPAR1WAY Procedure

       Median Scores (Number of Points Above Median) for Variable recall
                         Classified by Variable game

                         Sum of      Expected       Std Dev          Mean
     game         N      Scores      Under H0      Under H0         Score

     elabor      42        23.0     20.764045      2.367963      0.547619
     pairwi      47        21.0     23.235955      2.367963      0.446809

                        Average scores were used for ties.
                            Median Two-Sample Test

                          Statistic            23.0000
                          Z                     0.9443
                          One-Sided Pr >  Z     0.1725
                          Two-Sided Pr > |Z|    0.3450
```

```
                     Median One-Way Analysis

                  Chi-Square              0.8916
                  DF                           1
                  Pr > Chi-Square         0.3450
                      The NPAR1WAY Procedure

   Median Scores (Number of Points Above Median) for Variable recall
                    Classified by Variable game

                       Sum of      Expected       Std Dev          Mean
 game          N       Scores      Under H0      Under H0         Score

 simple       50         28.0          25.0      2.375671      0.560000
 elabor       42         18.0          21.0      2.375671      0.428571

                Average scores were used for ties.
                     Median Two-Sample Test

                  Statistic              18.0000
                  Z                      -1.2628
                  One-Sided Pr <  Z       0.1033
                  Two-Sided Pr > |Z|      0.2067
                     Median One-Way Analysis

                  Chi-Square              1.5947
                  DF                           1
                  Pr > Chi-Square         0.2067
                      The NPAR1WAY Procedure

   Median Scores (Number of Points Above Median) for Variable recall
                    Classified by Variable game

                       Sum of      Expected       Std Dev          Mean
 game          N       Scores      Under H0      Under H0         Score

 simple       50         36.0     24.742268      2.473690      0.720000
 pairwi       47         12.0     23.257732      2.473690      0.255319

                Average scores were used for ties.
                     Median Two-Sample Test

                  Statistic              12.0000
                  Z                      -4.5510
                  One-Sided Pr <  Z       <.0001
                  Two-Sided Pr > |Z|      <.0001
                     Median One-Way Analysis

                  Chi-Square             20.7115
                  DF                           1
                  Pr > Chi-Square         <.0001
```

Only the last one is significant here, and it remains significant after I adjust its P-value by multiplying it by three (to account for my having done three tests). That is, the only significant difference here is between `simple` and `pairwise`, with `simple` being better; the difference between `elaborate` and `pairwise` is no longer anywhere close to being significant.

I smelled a rat[33] here, because if you look at the boxplots, the means of `elaborate` and `pairwise` are different enough to be significantly different, and the medians differ by *more*, and yet are apparently not significantly different.

Let me create a data set with just the `elaborate` and `pairwise` values, and investigate using `proc freq`. Step one:

```
data names2;
  set names;
  if game='elabor' | game='pairwi';
```

Quick check that this is the right thing:

```
proc means;
  var recall;
  class game;
```

```
                              The MEANS Procedure

                         Analysis Variable : recall

              N
game        Obs     N          Mean         Std Dev         Minimum         Maximum
-----------------------------------------------------------------------------------
elabor       42    42     26.2142857      23.7019457               0      86.0000000

pairwi       47    47     15.1276596      15.7032495               0      66.0000000
-----------------------------------------------------------------------------------
```

This should check with my proc means earlier, and does, for the two groups I have left. Now, I am going to check SAS's Mood median test, so I first need the overall median of recall, for these two groups:

```
proc means median;
  var recall;
```

```
                         The MEANS Procedure

                     Analysis Variable : recall

                               Median
                         ------------
                           13.0000000
                         ------------
```

This seems a bit small, but I guess it's OK. Next, make a new column (making a new data set) that contains above and below this:

```
data names3;
  set names2;
  if (recall>13) then above_below='above';
  else above_below='below';

proc freq;
  tables game*above_below / chisq;
```

I printed this out first to check that it was OK, and it was. Then I made a cross-tabulation of above/below against `game`, as if we were building this ourselves, and let SAS run its battery of chi-squared tests on it. Results:

```
                                The FREQ Procedure

                           Table of game by above_below

                       game      above_below

                       Frequency|
                       Percent  |
                       Row Pct  |
                       Col Pct  |above   |below   |  Total
                       ---------+--------+--------+
                       elabor   |     23 |     19 |     42
                                |  25.84 |  21.35 |  47.19
                                |  54.76 |  45.24 |
                                |  52.27 |  42.22 |
                       ---------+--------+--------+
                       pairwi   |     21 |     26 |     47
                                |  23.60 |  29.21 |  52.81
                                |  44.68 |  55.32 |
                                |  47.73 |  57.78 |
                       ---------+--------+--------+
                       Total          44       45       89
                                   49.44    50.56   100.00
                   Statistics for Table of game by above_below

              Statistic                     DF       Value      Prob
              ------------------------------------------------------
              Chi-Square                     1      0.9017    0.3423
              Likelihood Ratio Chi-Square    1      0.9032    0.3419
              Continuity Adj. Chi-Square     1      0.5435    0.4610
              Mantel-Haenszel Chi-Square     1      0.8916    0.3450
              Phi Coefficient                       0.1007
              Contingency Coefficient               0.1002
              Cramer's V                            0.1007
```

```
       Fisher's Exact Test
----------------------------------
Cell (1,1) Frequency (F)        23
Left-sided Pr <= F          0.8774
Right-sided Pr >= F         0.2306

Table Probability (P)       0.1080
Two-sided Pr <= P           0.3988
         Sample Size = 89
```

The row and column totals check out: 42 in `elaborate`, 47 in `pairwise`, and as even a split as we can manage between above and below. But look at the frequencies *in* the table: both groups have an almost even split of recall values above and below 13, and so it is scarcely surprising that the result would not be significant.

We can go one step further with our checking: if we go back to the `proc npar1way` output for elaborate vs. pairwise, the numbers in the "sums of scores" column are 23 and 21. These are the numbers of values strictly above the overall median of 13. The table we just made contains the same frequencies, so it looks as if Mood's median test is correct. (The output from `proc npar1way` is the same as the Mantel-Haenszel here, which we previously discovered is what it's doing for a 2×2 table.)

In short, the means are clearly different between these two groups, but the medians *are not.* Weird but true.

Yet further analysis: what is it about those values that makes this happen? Let's count how many are in some classes. I just found out how to do this:

```
proc format;
  value myrecall
  0 - 13 = 'less than 13'
  14 - 25 = '14-25'
  26 - 40 = '26-40'
  41 - 60 = '41-60'
  61 - 100 = '61 up'
;

proc freq;
  tables game*recall;
  format recall myrecall.;
```

```
                                The FREQ Procedure

                             Table of game by recall

         game      recall

         Frequency|
         Percent  |
         Row Pct  |
         Col Pct  |less tha|14-25   |26-40   |41-60   |61 up   |  Total
                  |n 13    |        |        |        |        |
         ---------+--------+--------+--------+--------+--------+
         elabor   |     19 |      1 |     13 |      5 |      4 |     42
                  |  21.35 |   1.12 |  14.61 |   5.62 |   4.49 |  47.19
                  |  45.24 |   2.38 |  30.95 |  11.90 |   9.52 |
                  |  42.22 |  10.00 |  61.90 |  62.50 |  80.00 |
         ---------+--------+--------+--------+--------+--------+
         pairwi   |     26 |      9 |      8 |      3 |      1 |     47
                  |  29.21 |  10.11 |   8.99 |   3.37 |   1.12 |  52.81
                  |  55.32 |  19.15 |  17.02 |   6.38 |   2.13 |
                  |  57.78 |  90.00 |  38.10 |  37.50 |  20.00 |
         ---------+--------+--------+--------+--------+--------+
         Total          45       10       21        8        5       89
                     50.56    11.24    23.60     8.99     5.62   100.00
```

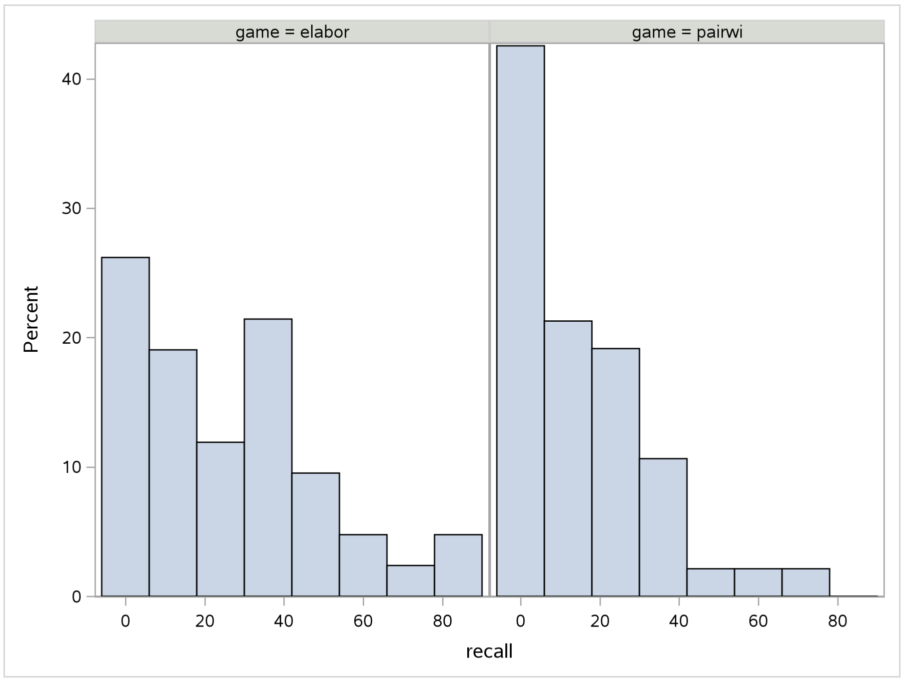
This is a way of categorizing a continuous variable (which is normally a bad idea, but I have my reasons here).

I am interested in the values bigger than 13 (which is the overall median): both groups have about equal numbers of values above and below 13 (so that the Mood median test was not significant.

Now let's think about where the medians are. For `pairwise`, the median is a bit below 13, since a bit more than half the observations are less than 13. But for `elaborate`, the median is up in the 26–40 class, because there are $19 + 1 = 20$ observations less than 25, and we haven't quite reached halfway up the data yet. So the medians are very different (and so are the means, since `elaborate` has more high values), but the split above/below 13 is about the same for both groups. That explains what we saw (and is an example of Mood's median test not explicitly comparing the medians of the two groups).

Another way to look at this is via the *shapes* of the distributions. The `pairwise` distribution is skewed right, but the `elaborate` distribution is *bimodal*: there are very few observations between 14 and 25, and a lot more either side. We didn't see this in the boxplots, because boxplots don't show bimodality; we'd have to look at paneled histograms for that:

```
proc sgpanel;
  panelby game;
  histogram recall;
```



The bimodality definitely shows up, and the median is in that tall bar between 30 and 40, whereas if there had not been a second mode, the median and mean would have been less.

Well, I didn't expect that to be so long, but now you know.

9 Reports in SAS

9.1. So far, we've been talking about reproducible research with R. SAS also has a system, called `statrep`, that works with LaTeX (which is another reason to learn it). The Statrep procedure is like this:

- Construct a specially-formatted LaTeX file that contains the Statrep version of code chunks.
- Recompile this file. This produces a `.sas` file, containing SAS commands, some of which you will recognize if you look at it.
- Upload the `.sas` file to SAS Studio. Run that file.
- Find the output, download it, and upload it to Overleaf.
- Recompile. This produces a document with code and output.

Let me take you through this. I'll assume you are using online SAS Studio along with Overleaf. The procedure is less fiddly if you have SAS (eg. University Edition) and LaTeX on your own computer, but that is not necessary. Most of the fiddliness is in downloading and uploading files.

(a) Download the files you need from SAS: `http://support.sas.com/rnd/app/papers/statrep/statrep.zip`. This is a "zip archive". Extract all the files from it. (It is likely that double-clicking on `statrep.zip` will display the files in it, and you will have the option to Extract the files somewhere.)

(b) Open up Overleaf. Create a new project, which you can call `statrep1`, with an "empty document" (which is actually a template called `main.tex` with some boilerplate in it).

(c) Find these four files that you extracted from `statrep.zip`:

- `longfigure.sty`
- `statrep.dtx`
- `statrep.ins`
- `statrep.sty`

Upload these from wherever they are on your computer to Overleaf. (The third icon below Menu is Upload.) When you have done this, the files should be listed on the left below `main.tex`.

(d) Now put the following in the `.tex` file that you created before. You can copy-paste this and delete what is currently there, as we did in Overleaf before. This is the soap data that we looked at before:

```
\documentclass{article}
\usepackage{statrep}
\def\SRrootdir{/home/megan3}
\def\SRmacropath{/home/megan3/statrep_macros.sas}

\begin{document}

Let us enter some data and print it:

\begin{Datastep}
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/soap.txt";

proc import
  datafile=myurl
  out=soap
  dbms=dlm
  replace;
  getnames=yes;
  delimiter=" ";
\end{Datastep}

Now we print it out:

 \begin{Sascode}[store=printout]
 proc print;
 \end{Sascode}

Now we have one more step to actually see it:

\Listing[store=printout,caption={The printed out data}]{a}
\end{document}
```

Solution: Go through this and replace the username `megan3` with yours.

The exact details don't actually matter; the point is that we needs to read in some data and print it out by way of testing.

As to what this file contains: the usual `\documentclass` and the beginning and ending of the document. We also need to load a package called `statrep` (that was the purpose of uploading all those files called `statrep`-dot-something into this project).

You'll see that the rest of the `.tex` file has some things that look like code chunks. In Statrep, there are actually three different kinds of things:

- A `Datastep` environment, which contains a SAS data step. This is where you'd read in data from a file. The point of this environment is that reading in a file doesn't produce any output.
- A `Sascode` environment. This contains a SAS `proc`-something (or more than one). This one is about the simplest one imaginable. Note that it has an "optional" argument `store=printout`, which is not really optional, since it is a name by which you will refer to this later.
- A `Listing` function. This is how you actually get to show (text) output, using that `store=` name that you put on the `Sascode` environment.

Note that all of these things have *initial capital* letters.

(e) Go ahead and recompile this. What do you see?

Solution: It doesn't work, yet, but for a good reason. The structure of the file is right, but where the SAS output should be, there is a "missing file". This is because we haven't actually run SAS yet! All the processing so far has been on Overleaf.

(f) There should be a file in your Overleaf project called `output_SR.sas`. This contains SAS code that will produce your output. It is rather difficult to find. Next to the green Recompile button, you'll see a little button that looks like a rendering of a piece of paper. Its tooltip is "Logs and output files". Click it. Look for "Other logs and files" on the right. Click *that*. This is where `output_SR.sas` is. Select it. That will download it. Do the same thing with `output_SR_preamble.sas`. The two files are now in your Downloads folder, or wherever things get downloaded to on your computer. (If you look at `output_SR.sas`, you should see all your code plus some other lines beginning with percent signs.) Running LaTeX on your document the first time produced these files.

(g) Open up SAS Studio. Use the Upload button to upload both files to SAS Studio. Don't forget to click Files (Home) first so that the Upload up-arrow is not greyed out. You can open `output_SR.sas`, but don't run it yet.

(h) In the `.zip` file from which you extracted the other `statrep` files earlier, there is also a subfolder called `sas` with two SAS files named `statrep_macros.sas` and `statrep_tagset.sas`. Upload these to your working folder on SAS Studio also. (You only need to do this once, regardless of how many times you use Statrep.)

(i) Now you can run `output_SR.sas`. It should run without errors, but you *won't see any output.* (If you have errors, go back to your file in Overleaf and check that it is *absolutely correct.* When you have corrected any errors that you see, compile again, and upload the files to SAS Studio again, before repeating this part.)

(j) Where did the output go? In the subfolder `lst` in SAS Studio, you should see a file called `a.lst`. (`a` was the name on the very end of the `Listing` line.) Download that file to your computer, by finding it in the folder, right-clicking and selecting Download. (Folders are at the top of the list under Files (Home), above the files, so look there.)

(k) Now go back to Overleaf. Make a folder called `lst` in your project, and upload `a.lst` into it. (The second button under Menu creates a new subfolder in the current project.) To make sure it uploads to the right place, click on your folder `lst` so that its line is green. You can check that the file ended up there by clicking on the arrow at the left end of the `lst` line so it points *down.* `a.lst` should be listed there.

(l) Cross your fingers and recompile. The listing of the data set should appear on the second page: 27 lines of data, with variables `case`, `scrap`, `speed` and `line`, the last of which is categorical.

Solution:

Here's the second page of my output:

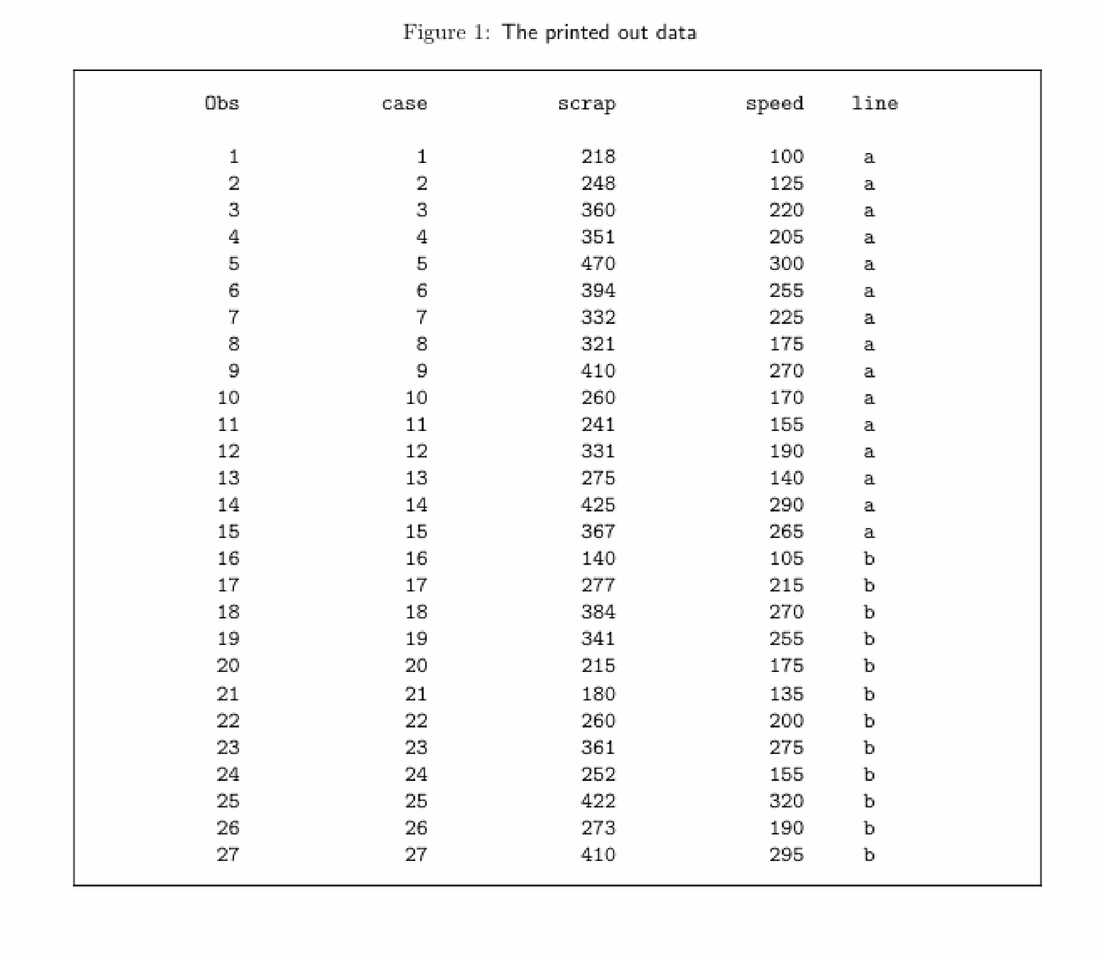


Figure 1: The printed out data

```
Obs     case     scrap     speed    line

  1        1       218       100     a
  2        2       248       125     a
  3        3       360       220     a
  4        4       351       205     a
  5        5       470       300     a
  6        6       394       255     a
  7        7       332       225     a
  8        8       321       175     a
  9        9       410       270     a
 10       10       260       170     a
 11       11       241       155     a
 12       12       331       190     a
 13       13       275       140     a
 14       14       425       290     a
 15       15       367       265     a
 16       16       140       105     b
 17       17       277       215     b
 18       18       384       270     b
 19       19       341       255     b
 20       20       215       175     b
 21       21       180       135     b
 22       22       260       200     b
 23       23       361       275     b
 24       24       252       155     b
 25       25       422       320     b
 26       26       273       190     b
 27       27       410       295     b
```

This is reproducible just as the output from R Markdown is, in that we know that the output came from the code, because it was literally produced by running the code.

If you wish to download the output so that you can be sure it worked, look for the second button to the right of Recompile, whose tooltip is Download PDF. Clicking that will download the output file so that you can inspect it at your leisure.

(m) Let's try something a bit more ambitious. This time, we're going to shoot for the output from `proc means` plus a boxplot. But the procedure is what we just did. First, go back to your file `main.tex` in Overleaf, and add these lines between the end of the previous and the `\end{document}`:

```
Next, we run \texttt{proc means} and obtain a boxplot.
That is done with this code, on the most recently created
(or only) data set:

 \begin{Sascode}[store=means]
 proc means;
   var scrap;
   class line;

 proc sgplot;
```

```
   vbox scrap / category=line;
 \end{Sascode}

Now we look at the output of these separately.
First the printed output from \texttt{proc means}:

\Listing[store=means,caption={Mean scrap for each line}]{b}

and then the boxplot. Note that this is produced by a
different command, because it is a graphic and not text:

\Graphic[store=means,caption={Boxplots of scrap for each line}]{c}
```

Solution: Not much new here. A `Sascode` block can have more than one `proc` in it; in this case it has two. As for getting hold of the output, well, we have to get the text and the graphics separately. Note the use of a new function `\Graphic` to obtain the boxplot. (This will exist, after you've run SAS, in a file `c.png` because of the `{c}` on the end of the line.)

(n) Recompile your new code, and note the two extra "missing files" in the output.

(o) Download the *new* `output_SR.sas` and `output_SR_preamble.sas` files from Overleaf, and upload them to SAS Studio. They might have gotten downloaded with different names (since you already downloaded files with those names), so make sure you upload the most recent versions.

(p) Run `output_SR.sas`, or whichever name the file acquired. It should run with no errors but produce no visible output. (To be sure that you ran the right code, take a look at the Code tab and make sure there's something that looks like boxplot code at the bottom.)

(q) In SAS Studio, look in the `lst` subfolder. There should now be two files, `a.lst` (the `proc print` output) and `b.lst` (the `proc means` output). Download those. (My `a.lst` got saved as `a (1).lst` so as not to overwrite the previous one. Make sure you open the right one, although in this case it doesn't matter because they are both the same.) Upload these files to the subfolder `lst` on Overleaf, using the names `a.lst` and `b.lst`. If that's not the names the files have, upload them anyway, and then delete/rename the files by right-clicking on them.

(r) In SAS Studio, you should now have a subfolder called `png` with a picture `c.png` in it. Download it to your computer.

(s) Go back to Overleaf. Create a new folder called `png`, and upload `c.png` into it.

(t) Recompile your document. If everything is where it should be, you'll get a table of means and a boxplot.

Solution: In my output, I got this, on page 3:

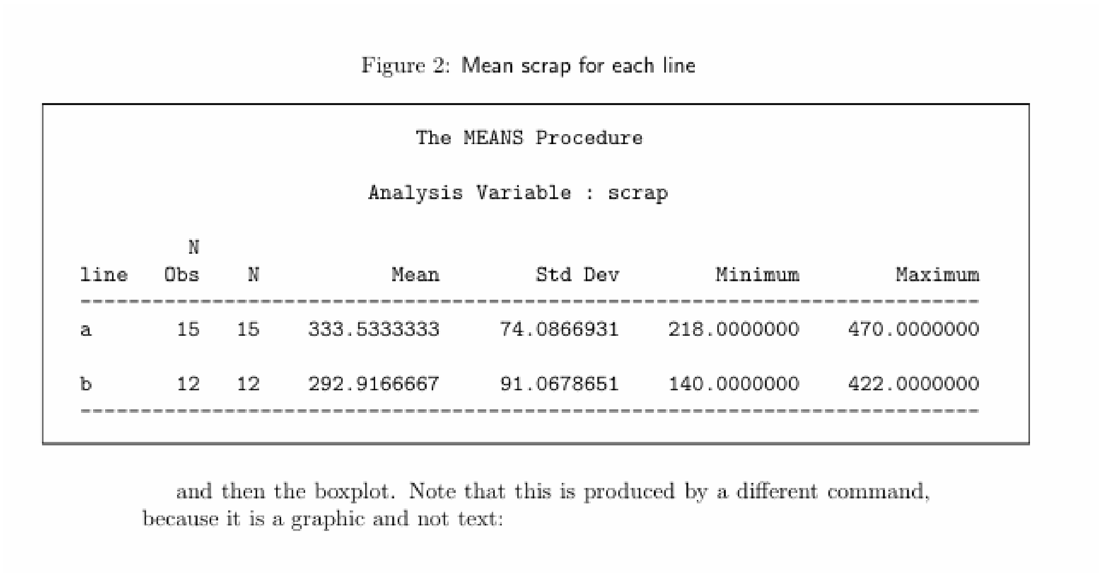


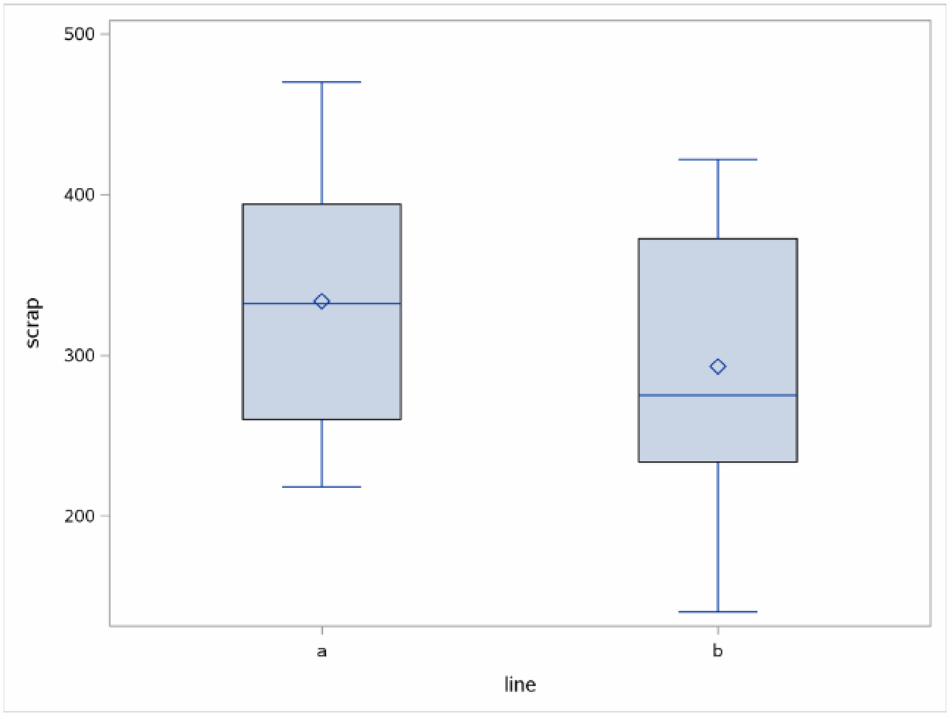
Figure 2: Mean scrap for each line

```
                          The MEANS Procedure

                       Analysis Variable : scrap

          N
line    Obs    N          Mean        Std Dev        Minimum        Maximum
----------------------------------------------------------------------------
a        15   15   333.5333333     74.0866931    218.0000000    470.0000000

b        12   12   292.9166667     91.0678651    140.0000000    422.0000000
----------------------------------------------------------------------------
```

and then the boxplot. Note that this is produced by a different command, because it is a graphic and not text:

and this, on page 4:

Figure 3: Boxplots of scrap for each line



9.2. This again uses the movie rating data at `http://www.utsc.utoronto.ca/~butler/c32/movie-lengths.`

csv.

(a) (3 marks) Read the data into SAS, and run Mood's median test. Does it give similar results to R's?

Solution: No credit for reading in the data, since you did that before:

```
filename myurl url 'http://www.utsc.utoronto.ca/~butler/c32/movie-lengths.csv';

proc import
  datafile=myurl
  dbms=csv
  out=movies
  replace;
  getnames=yes;
```

and then, noting that the `var` and `class` are the same as on `proc means`:

```
proc npar1way median;
  var length;
  class rating;
```

```
                           The NPAR1WAY Procedure

       Median Scores (Number of Points Above Median) for Variable length
                        Classified by Variable rating

                            Sum of      Expected       Std Dev          Mean
     rating        N        Scores      Under H0      Under H0         Score

     G            15          2.00          7.50      1.662778      0.133333
     PG-13        15         12.00          7.50      1.662778      0.800000
     PG           15          7.50          7.50      1.662778      0.500000
     R            15          8.50          7.50      1.662778      0.566667

                        Average scores were used for ties.
                             Median One-Way Analysis

                            Chi-Square         13.9701
                            DF                       3
                            Pr > Chi-Square     0.0029
```

The "sum of scores" for G and PG-13 is the same number of values above 100 that we had before. There are two scores of exactly 100, so they count half above and half below in each group (hence the 7.5 and 8.5).

The P-value of 0.0029 is very similar to whichever of my R variants you had, and so the conclusion is the same: the medians are not all equal. This P-value is slightly different from any of the ones I had in R, because of two things: (i) the way the exactly-100 values are counted, and (ii) the way, that I still haven't figured out, that SAS calculates the `Chi-square` test statistic, which is not the same as the usual observed minus expected, squared, divided by expected, that you might know from elsewhere.

OK, two points for doing the test properly (you got credit for reading in the data elsewhere), and one point for saying that the P-value is different but almost the same (or that the test statistic is different but almost the same).

If you want to do the multiple comparisons in SAS, you do as I did before, using a `where` line to compare only pairs of ratings, and then Bonferroni-ize the P-values at the end: that is, only reject if each P-value is less than 0.05/6. This is a lot of work because there are six pairs, but here is one of them:

```
proc npar1way median;
  where rating='G' or rating='PG-13';
  var length;
  class rating;
```

```
                          The NPAR1WAY Procedure

      Median Scores (Number of Points Above Median) for Variable length
                        Classified by Variable rating

                           Sum of      Expected       Std Dev          Mean
     rating       N        Scores      Under H0      Under H0         Score

     G           15           2.0          7.50      1.392715      0.133333
     PG-13       15          13.0          7.50      1.392715      0.866667

                        Average scores were used for ties.
                           Median Two-Sample Test

                         Statistic              2.0000
                         Z                     -3.9491
                         One-Sided Pr <  Z      <.0001
                         Two-Sided Pr > |Z|     <.0001
                           Median One-Way Analysis

                         Chi-Square            15.5956
                         DF                          1
                         Pr > Chi-Square        <.0001
```

$0.05/6 = 0.0083$, so these two are definitely different.

SAS also has loops, so you could do something along the lines of the Python-like solution that I did in R.

10 Mood's median test

10.1. This question is about the Blue Jays data set (that I used in class), in SAS. The data can be found at `http://www.utsc.utoronto.ca/~butler/c32/jays15-home.csv`.

(a) Read the data into SAS, and use `proc print` to verify that you have the right attendances and day/night values (don't show the other variables).

Solution: This is the usual thing for reading in a `.csv` file. By now, I'm sure you have a lot of examples. I copied this one from the North Carolina births and changed some names:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/jays15-home.csv";

proc import
  datafile=myurl
  dbms=csv
  out=jays
  replace;
  getnames=yes;
```

To display just those two variables, put them on a `var` line in the `proc print`:

```
proc print;
  var attendance Daynight;
```

```
                    Obs      attendance    Daynight

                      1         48414         N
                      2         17264         N
                      3         15086         N
                      4         14433         N
                      5         21397         N
                      6         34743         D
                      7         44794         D
                      8         14184         N
                      9         15606         N
                     10         18581         N
                     11         19217         N
                     12         21519         N
                     13         21312         N
                     14         30430         N
                     15         42917         D
                     16         42419         D
                     17         29306         D
                     18         15062         N
                     19         16402         N
                     20         19014         N
                     21         21195         N
                     22         33086         D
                     23         37929         D
                     24         15168         N
                     25         17276         N
```

That indeed looks right.

(b) Obtain the mean and standard deviation of attendance for each of day and night games. (If this is difficult, you are thinking about it too much!)

Solution: Just proc means:

```
proc means;
  var attendance;
  class daynight;
```

```
                             The MEANS Procedure

                        Analysis Variable : attendance

              N
Daynight   Obs    N            Mean         Std Dev         Minimum         Maximum
--------------------------------------------------------------------------------
D            7    7        37884.86         5775.18        29306.00        44794.00

N           18   18        20086.67         8083.64        14184.00        48414.00
--------------------------------------------------------------------------------
```

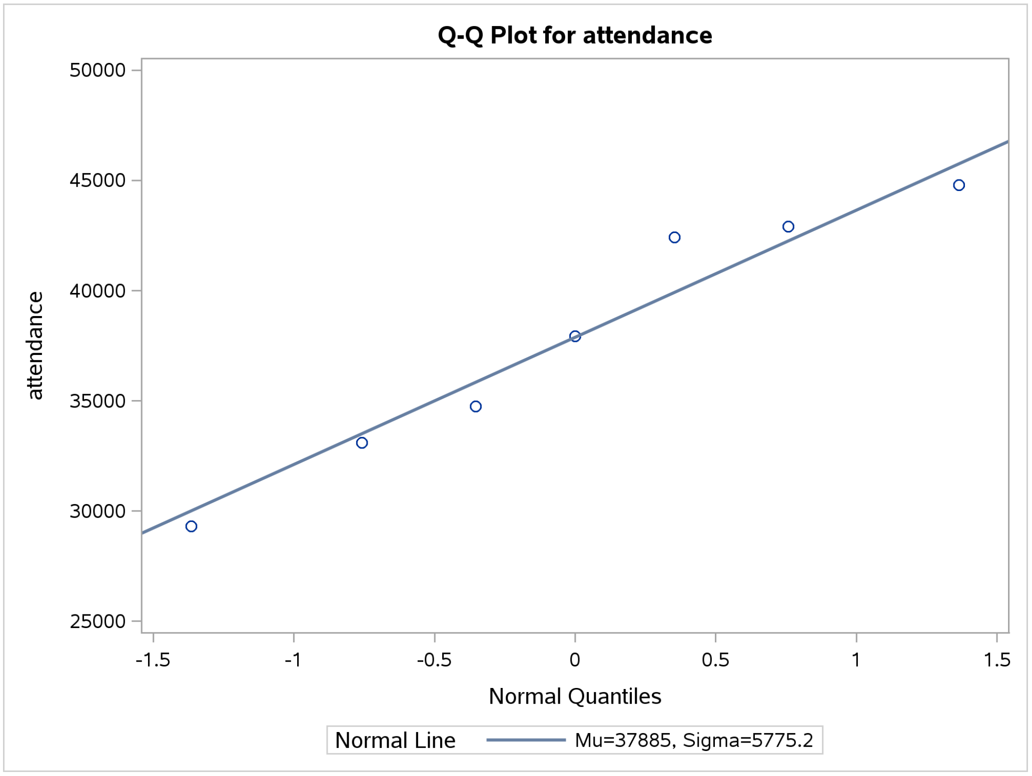
These are identical with R's values.

(c) Using SAS, make a normal quantile plot of just the day games' attendances. You'll have to select out just the day attendances first. This uses the `where` idea, applied to a `proc` (see `http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#a001000758.htm`).

Solution:

I add the `where` part to `proc univariate`, also using the `noprint` option so I don't get all the numerical stuff (that I don't want):

```
proc univariate noprint;
  where daynight='D';
  qqplot attendance / normal(mu=est sigma=est);
```

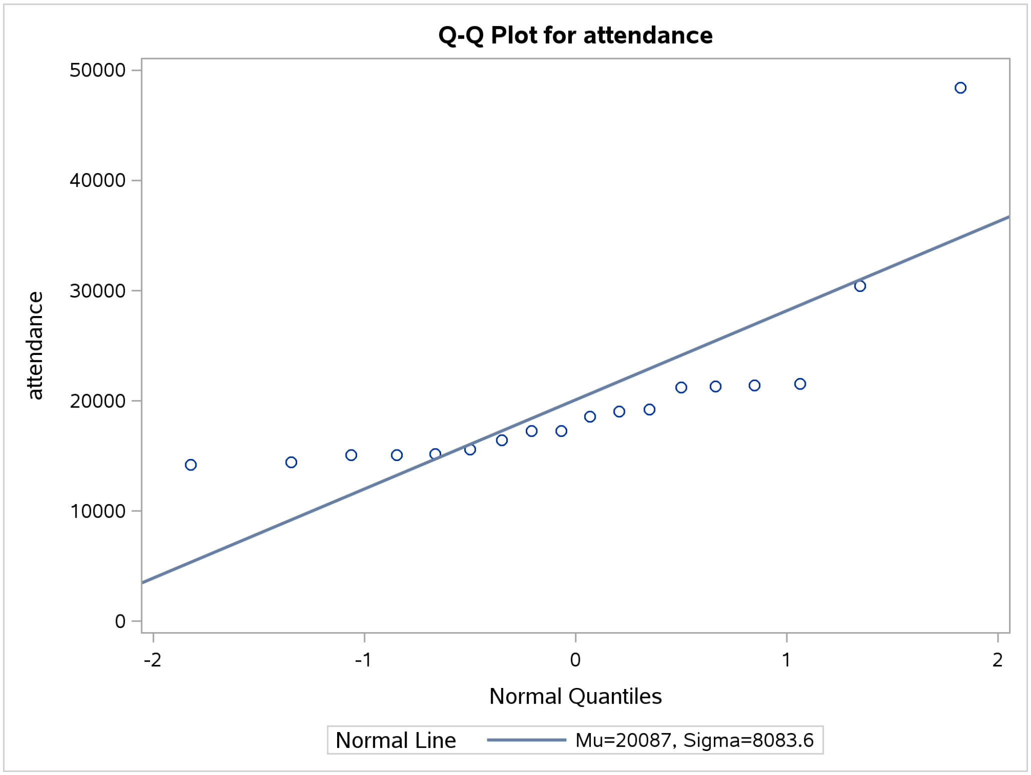


This looks a lot like R's normal quantile plot, and tells the same story: insofar as we can judge with only seven observations, these look pretty normal.

(d) Now use SAS to make a normal quantile plot of just the night games' attendances. (Copy and edit the code you just used.)

Solution: Same idea, only change the `where`:

```
proc univariate noprint;
  where daynight='N';
  qqplot attendance / normal(mu=est sigma=est);
```


Q-Q Plot for attendance
attendance
Normal Quantiles
Normal Line
Mu=20087, Sigma=8083.6

This one looks more obviously like a curve than R's version of the same thing, but the same points have been plotted. The only thing that's different is the line, which goes a bit higher at the end towards the high outlier (making it look less like an outlier), and it goes almost through the second-highest point, suggesting that this one is all right (when R said it was too high as well).

R and SAS's normal quantile plot lines are different, because they are constructed differently. R's goes through the first and third quartiles, and so it will tend to be consistent with the bulk of the data and not be swayed by outliers. SAS's, on the other hand, is based on the sample mean and SD, and these (especially the standard deviation) can be badly affected by outliers. So when the normal quantile plot has outliers, the lines can be very different.

We saw elsewhere how to estimate `mu` and `sigma` using the median and interquartile range. So let's get the median and interquartile range of the night attendances:

```
proc means median qrange;
  var attendance;
  class daynight;
```

```
                        The MEANS Procedure

                   Analysis Variable : attendance

                       N                            Quartile
        Daynight     Obs          Median               Range
        ----------------------------------------------------
        D              7        37929.00             9831.00

        N             18        17928.50             6144.00
        ----------------------------------------------------
```

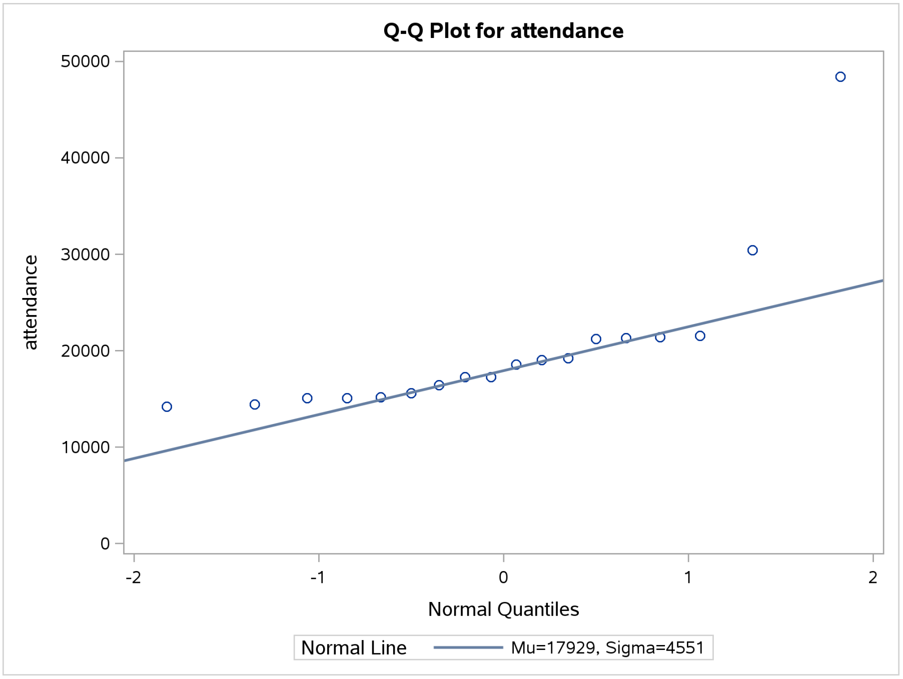
The interquartile range is 6144 and the median is 17928.5. The latter is our estimate of μ and our estimate of σ is

```
6144/1.35

## [1] 4551.111
```

The sample *mean* is 20086.7, noticeably bigger than the median, and the sample *SD* is 8084, which is a lot bigger than our estimate of σ that came from the IQR.

So now we can re-draw SAS's normal quantile plot. Instead of saying `=est`, we replace `est` with the appropriate number:

```
proc univariate noprint;
  where daynight='N';
  qqplot attendance / normal(mu=17928.5 sigma=4551);
```



This line looks a lot more like R's: the highest value is obviously an outlier, maybe the second highest one is too, and those values at the bottom are too clustered together to be the bottom-most few values in a normal distribution.

(e) Run Mood's median test to see whether the median attendances at day and night games are significantly different. What do you conclude?

Solution:

```
proc npar1way median;
  var attendance;
  class daynight;
```

That gives this output:

```
                        The NPAR1WAY Procedure

    Median Scores (Number of Points Above Median) for Variable attendance
                      Classified by Variable Daynight

                          Sum of      Expected       Std Dev          Mean
Daynight       N          Scores      Under H0      Under H0         Score

N             18             5.0         8.640      1.144727      0.277778
D              7             7.0         3.360      1.144727      1.000000
                        Median Two-Sample Test

                    Statistic             7.0000
                    Z                     3.1798
                    One-Sided Pr >  Z     0.0007
                    Two-Sided Pr > |Z|    0.0015
                      Median One-Way Analysis

                    Chi-Square           10.1111
                    DF                         1
                    Pr > Chi-Square       0.0015
```

As you might expect, the (population) medians *are* different, and decisively so: a P-value of 0.0015 (or half that if you could have justified a one-sided alternative hypothesis before looking at the data). There are no issues about whether this conclusion is trustworthy or not, because it does not depend on anything being normal. So we can conclude that the population medians are different, even though the night attendances have those outliers.

One thing the Mood Median Test does *not* give us is a confidence interval for the difference between medians. We were able to make a CI for the median from the sign test by saying something like "what if the population median were x?" and deciding whether you would reject a median of x or not (and making your interval be all of the values that you wouldn't reject). That kind of idea doesn't work here because Mood's test is designed to test only whether two medians are *the same*: it doesn't have any mechanism for testing whether the two medians differ by x. If it did, we could test a whole bunch of median differences this way, and the ones we don't reject would make up our confidence interval, which would be analogous to our procedure with the sign test. But, unfortunately, we are out of luck. (If you want to explore, SAS includes "Hodges-Lehmann estimation of location", but that assumes that the two groups have equal spread, an assumption we may well not be willing to make.)

10.2. A biology graduate student exposed each of 32 tomato plants to one of four different colours of light (8 plants to each colour). The growth rate of each plant, in millimetres per week, was recorded. The data are in `http://www.utsc.utoronto.ca/~butler/c32/tomatoes.txt`.

We did this one before with R. One of the parts had you save the tidy data to a file which I called `tomatoes2.csv`. If you did that problem, find the file on `rstudio.cloud`, download it to your computer and upload it to SAS Studio. If you didn't, use the copy at `https://www.utsc.utoronto.ca/~butler/c32/tomatoes2.csv`. Download the file from there to your computer (it might open in Excel, in which case save it as `.csv` to somewhere on your computer, or it might get downloaded to a folder with a name like `Downloads`, in which case it will already be on your computer). From your computer, upload it to SAS Studio.

(a) Get the data file into a SAS dataset. This will mean something like (i) uploading the file from where it is now to SAS Studio, and (ii) reading it in to SAS from there. You should make it a habit of running `proc print` *for yourself* to check that the data values were read correctly, before you do anything else.

Solution: When you have gotten the file to SAS Studio, something like this should work, substituting your username for mine:

```
proc import
  datafile='/home/ken/tomatoes2.csv'
  out=tomatoes
  dbms=csv
  replace;
  getnames=yes;
```

I like to check what I got:

```
proc print;
```

with output

```
                 Obs           plant    colour      growthrate

                   1               1    blue              5.34
                   2               2    blue              7.45
                   3               3    blue              7.15
                   4               4    blue              5.53
                   5               5    blue              6.34
                   6               6    blue              7.16
                   7               7    blue              7.77
                   8               8    blue              5.09
                   9               1    red              13.67
                  10               2    red              13.04
                  11               3    red              10.16
                  12               4    red              13.12
                  13               5    red              11.06
                  14               6    red              11.43
                  15               7    red              13.98
                  16               8    red              13.49
                  17               1    yellow            4.61
                  18               2    yellow            6.63
                  19               3    yellow            5.29
                  20               4    yellow            5.29
                  21               5    yellow            4.76
                  22               6    yellow            5.57
                  23               7    yellow            6.57
                  24               8    yellow            5.25
                  25               1    green             2.72
                  26               2    green             1.08
                  27               3    green             3.97
                  28               4    green             2.66
                  29               5    green             3.69
                  30               6    green             1.96
                  31               7    green             3.38
                  32               8    green             1.87
```

Success! You should certainly run `proc print` until you are happy that the data got read in properly.

(b) Find the mean growth rate for each colour. Would you expect an analysis of variance to come out significant? Explain briefly.

Solution: This is just `proc means`:

```
proc means;
  var growthrate;
  class colour;
```

with output

```
                               The MEANS Procedure

                          Analysis Variable : growthrate

                N
colour   Obs    N          Mean        Std Dev        Minimum        Maximum
------------------------------------------------------------------------------
blue       8    8     6.4787500      1.0467697      5.0900000      7.7700000

green      8    8     2.6662500      0.9934922      1.0800000      3.9700000

red        8    8    12.4937500      1.4096194     10.1600000     13.9800000

yellow     8    8     5.4962500      0.7480439      4.6100000      6.6300000
------------------------------------------------------------------------------
```

These means are *very* different from each other, so I would certainly expect an ANOVA to come out significant (or, a null of "all the means are equal" should be rejected).

Extra: as a yardstick, compare the differences in means with the standard deviations (which are all near 1). Most of the means differ by a lot more than 1. (You can make a more sophisticated yardstick out of standard errors, but this will do.)

(c) Run an analysis of variance to see whether there are any differences in mean growth rate among the different colours. What do you conclude?

Solution: This:

```
proc anova;
  class colour;
  model growthrate=colour;
  means colour / tukey;
```

(I did the Tukey line on the end, which I will show you in a minute. You need, as a minimum, the first three lines.)

The ANOVA output is:

```
                               The ANOVA Procedure

                            Class Level Information

                Class         Levels    Values

                colour             4    blue green red yellow

                   Number of Observations Read          32
                   Number of Observations Used          32
```

```
                                 The ANOVA Procedure

                            Dependent Variable: growthrate

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

 Model                        3     410.4687000    136.8229000    118.22   <.0001

 Error                       28      32.4054500      1.1573375

 Corrected Total             31     442.8741500
            R-Square     Coeff Var      Root MSE    growthrate Mean

            0.926829      15.85843      1.075796           6.783750
 Source                      DF        Anova SS    Mean Square   F Value   Pr > F

 colour                       3     410.4687000    136.8229000    118.22   <.0001
```

The null hypothesis is "all the means are equal", that is, the mean growth rate is the same for plants of each colour. With a P-value much smaller than 0.05, this null is rejected, in favour of the vague alternative "not all the means are equal". That is, the mean growth rate is different somehow for the different colours.

We cannot say more, yet.

(d) Explain briefly whether or not you need to obtain the output for Tukey's method. If you *do* need it, obtain it and explain briefly what it tells you.

Solution: All I know so far is that the mean growth rates are not all the same, but I don't know which colours differ from which, so I need to run Tukey's method.

You can do this two ways: put the `means colour / tukey` line on the end of your `proc anova` just in case (as I did above), and ignore the output if you don't need it, or you can re-run the entire `proc anova` with the extra line on the end. Either way is good: there is no penalty for making SAS work harder!

Here's mine. I have a way of saving it up from earlier and showing it to you now:

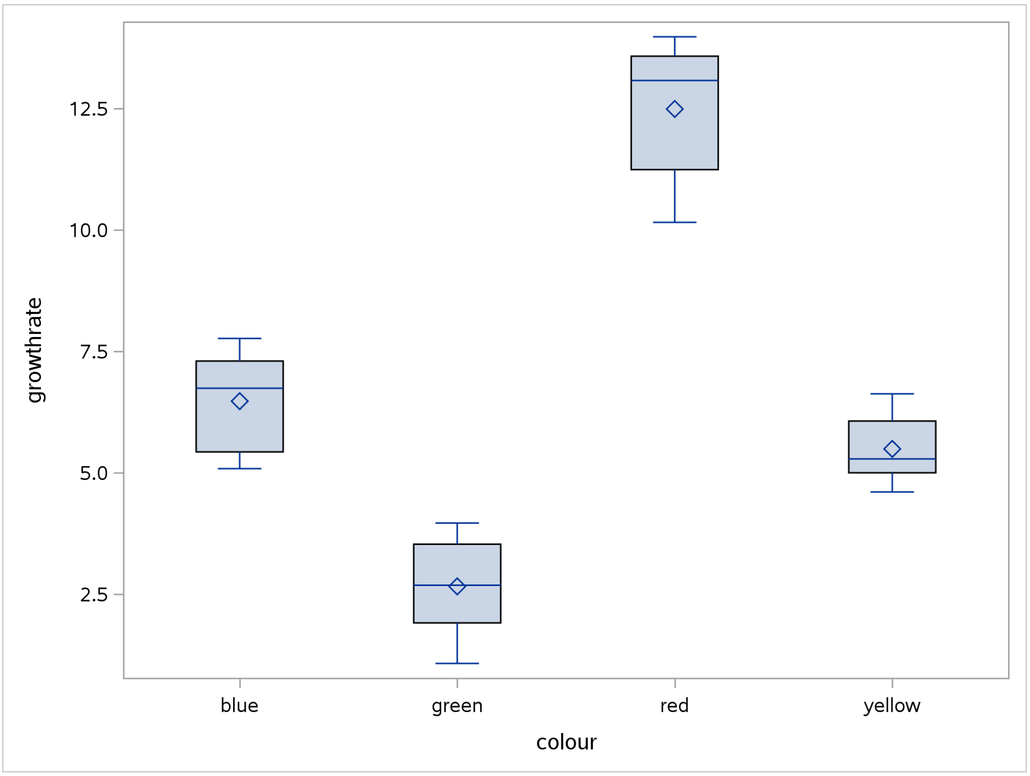
```
                                 The ANOVA Procedure

                    Tukey's Studentized Range (HSD) Test for growthrate

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


                  Alpha                                     0.05
                  Error Degrees of Freedom                    28
                  Error Mean Square                     1.157338
                  Critical Value of Studentized Range  3.86124
                  Minimum Significant Difference        1.4686
           Means with the same letter are not significantly different.


           Tukey Grouping          Mean      N    colour

                        A       12.4938      8    red

                        B        6.4788      8    blue
                        B
                        B        5.4963      8    yellow

                        C        2.6663      8    green
```

The mean growth rate is significantly higher for red than for any other colour, and significantly lower for green than any other colour. Yellow and blue are not significantly different from each other.

What I have not asked you to do here (since the question is long enough already) is to check the assumptions. These are like the ones for the two-sample pooled t-test: the observations within each group should be approximately normal with approximately equal spreads (variances). Something like side-by-side boxplots would be sufficient to check that (looking for any obvious asymmetry or outliers). I don't think there are any problems here: if you look back at the output from `proc means` in (e), the standard deviations are almost identical, and the min and max observations are nowhere more than about 2 standard deviations from their means, so it doesn't look as if we have any problems with outliers.

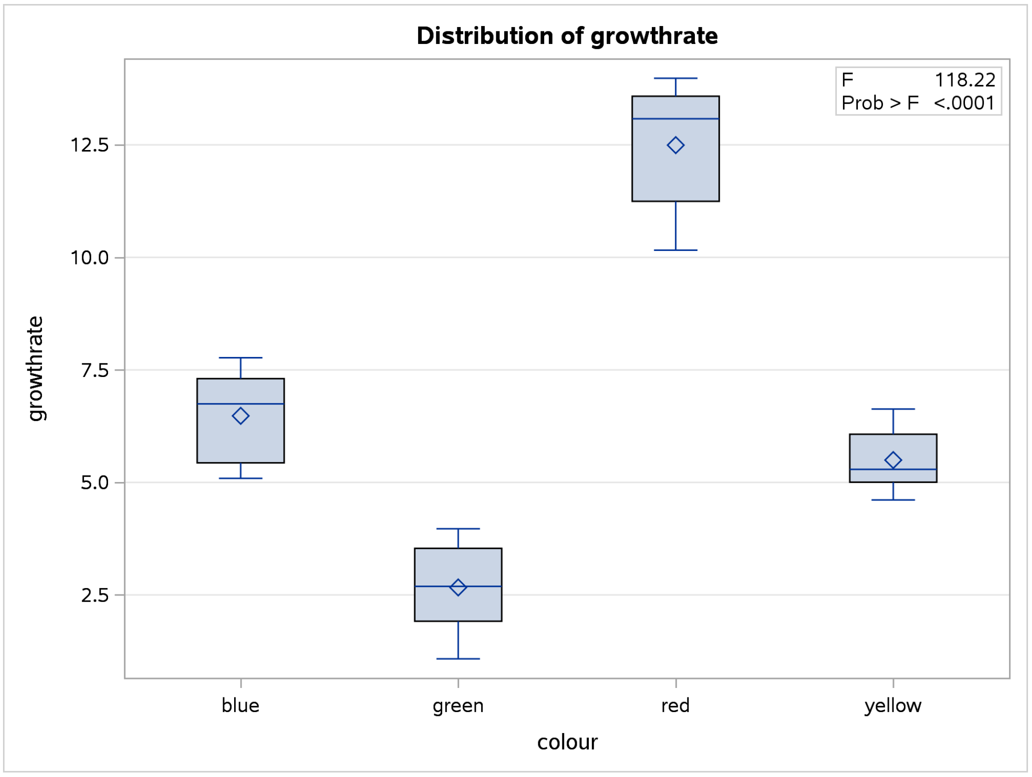
The obvious way to confirm this, should you wish to (I didn't ask for it) is side-by-side boxplots (which are also easiest to produce here).[34] They won't tell us completely about normal shape, but they will tell us about any problems with asymmetry or outliers, which is what we really care about:

```
proc sgplot;
  vbox growthrate / category=colour;
```



Now, bearing in mind that we have only 8 observations in each group, I think these are not bad at all. The red ones are a bit left-skewed, and the yellow ones have slightly smaller spread. But I don't think these are worth worrying about. I am perfectly happy with the ANOVA.

The boxplots in the ANOVA output are these:



Get them from there, or draw them yourself. Either is good.

10.3. The data in `http://www.utsc.utoronto.ca/~butler/c32/migraine.txt` are from a study of pain relief in migraine headaches. Specifically, 27 subjects were randomly assigned to receive *one* of three pain relieving drugs, labelled A, B and C. Each subject reported the number of hours of pain relief they obtained (that is, the number of hours between taking the drug and the migraine symptoms returning). A higher value is therefore better. Can we make some recommendation about which drug is best for the population of migraine sufferers?

We did this before with R.

(a) Read in and display the data. This will *not* work with `proc import`; see the solution for how to do it.

Solution:

We'll not be reading data like this into SAS in this course, but you might like to know how it goes (for future reference). It uses a `data` step, like we've been using to create new variables, thus:

```
filename myurl url 'http://www.utsc.utoronto.ca/~butler/c32/migraine.txt';
```

```
data pain;
  infile myurl firstobs=2;
  input druga drugb drugc;

proc print;
```

```
Obs    druga    drugb    drugc

 1       4        6        6
 2       5        8        7
 3       4        4        6
 4       3        5        6
 5       2        4        7
 6       4        6        5
 7       3        5        6
 8       4       11        5
 9       4       10        5
```

That has worked. Unlike `proc import`, which figures out what the columns are called, this way needs you (i) to specify names for all the columns and (ii) to skip the row of column names (which seems kind of backwards written like that), starting reading the data from line 2.

(b) What is wrong with the current format of the data as far as doing a one-way ANOVA analysis is concerned? (This is related to the idea of whether or not the data are "tidy".)

Solution: For our analysis, we need one column of pain relief time and one column labelling the drug that the subject in question took.

Or, if you prefer to think about what would make these data "tidy": there are 27 subjects, so there ought to be 27 rows, and all three columns are measurements of pain relief, so they ought to be in one column.

(c) "Tidy" the data to produce a data frame suitable for your analysis.

Solution:

The SAS version of `gather` goes like this.

```
data pain2;
  set pain;
  array drug_array [3] druga drugb drugc;
  do i=1 to 3;
    painrelief=drug_array[i];
    drug=vname(drug_array[i]);
    output;
  end;
  keep painrelief drug;
```

Did the "gather" work? `proc print` will print the most recently created data set, the tidy one:

```
proc print;
```

```
                              Obs    painrelief    drug

                                1         4        druga
                                2         6        drugb
                                3         6        drugc
                                4         5        druga
                                5         8        drugb
                                6         7        drugc
                                7         4        druga
                                8         4        drugb
                                9         6        drugc
                               10         3        druga
                               11         5        drugb
                               12         6        drugc
                               13         2        druga
                               14         4        drugb
                               15         7        drugc
                               16         4        druga
                               17         6        drugb
                               18         5        drugc
                               19         3        druga
                               20         5        drugb
                               21         6        drugc
                               22         4        druga
                               23        11        drugb
                               24         5        drugc
                               25         4        druga
                               26        10        drugb
                               27         5        drugc
```

It seems that it did. You can compare this with the R data frame that I called `migraine2`. It has all the same data values, but in a different order. R works column-by-column in the original data frame `migraine`, but SAS works one *row* at a time, dealing with all the drug values on each row of the input file before moving on to the next.

The order won't affect the analysis, though. Both ways give the "same" data as far as that is concerned.

(d) Go ahead and run your one-way ANOVA (and Tukey if necessary). Assume for this that the pain relief hours in each group are sufficiently close to normally distributed with sufficiently equal spreads.

Solution:

My last sentence is absolving us of the need to check for the usual ANOVA assumptions (or, you can take it for granted that I already did this and declared that I was happy). The easiest way to do this in SAS is to ask for Tukey up front, and just ignore it if you happen not to need it:

```
proc anova;
  class drug;
  model painrelief=drug;
  means drug / tukey;
```

Let me grab the whole output (well, the text part of it anyway):

```
                              The ANOVA Procedure

                            Class Level Information

                   Class         Levels    Values

                   drug               3    druga drugb drugc
                    Number of Observations Read          27
                    Number of Observations Used          27
                              The ANOVA Procedure

                       Dependent Variable: painrelief

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

 Model                       2     41.1851852     20.5925926      7.83   0.0024

 Error                      24     63.1111111      2.6296296

 Corrected Total            26    104.2962963
           R-Square     Coeff Var      Root MSE    painrelief Mean

           0.394886      30.19556      1.621613           5.370370
 Source                     DF       Anova SS    Mean Square   F Value   Pr > F

 drug                        2    41.18518519    20.59259259      7.83   0.0024
                              The ANOVA Procedure

               Tukey's Studentized Range (HSD) Test for painrelief

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


                  Alpha                                   0.05
                  Error Degrees of Freedom                  24
                  Error Mean Square                    2.62963
                  Critical Value of Studentized Range  3.53170
                  Minimum Significant Difference         1.909
          Means with the same letter are not significantly different.


          Tukey Grouping          Mean      N    drug

                       A        6.5556      9    drugb
                       A
                       A        5.8889      9    drugc

                       B        3.6667      9    druga
```

First off, the drugs are not all the same in terms of pain relief (P-value 0.0003, eg. from the `drug` line).[35]

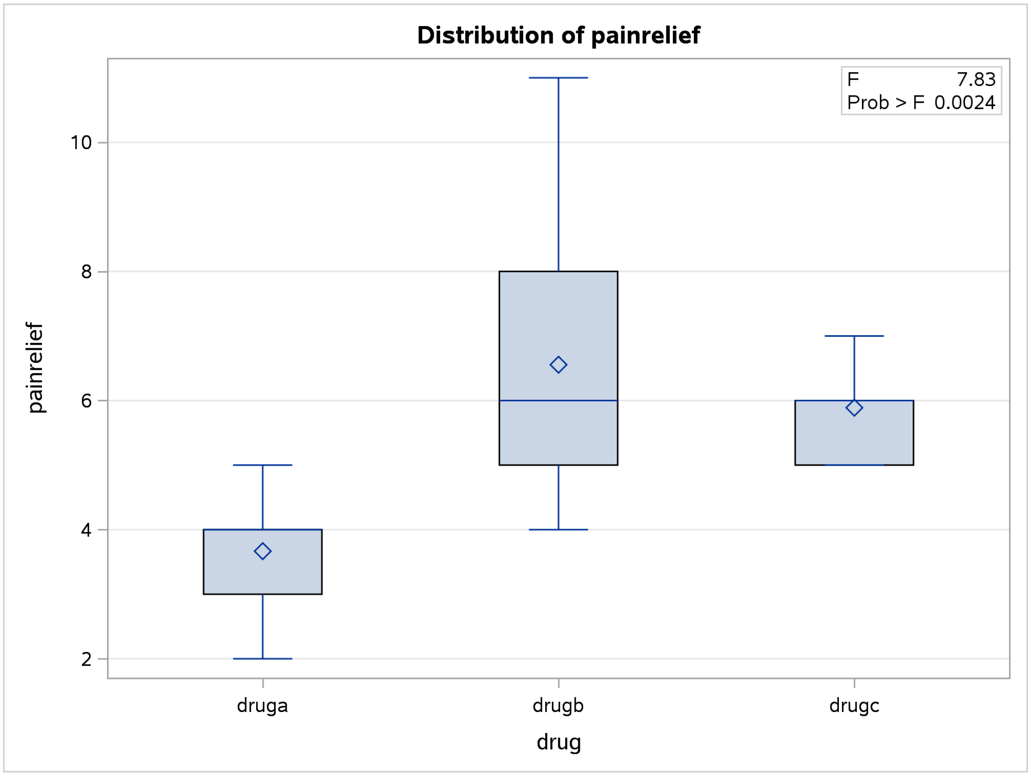
Having found differences, we look at the Tukey output. First off, don't get confused by the A and B on the *left* and the drug names, which are on the *right*. (The ones on the left are saying which groups differ significantly from others.) The top two drugs, B and C, are not significantly different in terms of mean pain relief (they have the same letter in the left column), but the bottom drug A is significantly worse than the other two.

(e) What recommendation would you make about the best drug or drugs? Explain briefly.

Solution: Drug A is significantly the worst, so we eliminate that. But there is no significant difference between drugs B and C, so we have no reproducible reason for preferring one rather than the other. Thus, we recommend "either B or C".

You should *not* recommend drug C over drug B on this evidence, just because its (sample) mean is higher than B's. The point about significant differences is that they are supposed to stand up to replication: in another experiment, or in real-life experiences with these drugs, the mean pain relief score for drug A is expected to be worst, but between drugs B and C, sometimes the mean of B will come out higher and sometimes C's mean will be higher, because there is no significant difference between them.[36][37]

Extra: another way is to draw a boxplot of pain-relief scores, which is actually in the output we just obtained:



The medians of drugs B and C are actually exactly the same. Because the pain relief values are all whole numbers (and there are only 9 in each group), you get that thing where enough of them are equal that the median and third quartiles are equal, actually for all three groups.

Despite the outlier, I'm willing to call these groups sufficiently symmetric for the ANOVA to be OK (but I didn't ask you to draw the boxplot, because I didn't want to confuse the issue with this. The point of this question was to get the data tidy enough to do an analysis.) Think about it for a moment: that outlier is a value of 8. This is really not that much bigger than the value of 7 that is the highest one on drug C. The 7 for drug C is not an outlier. The only reason the 8 came out as an outlier was because the IQR was only 1. If the IQR on drug B had happened to be a bit bigger, the 8 would not have been an outlier.

As I said, I didn't want you to have to get into this, but if you are worried, you know what the remedy is — Mood's median test.

```
proc npar1way median;
  var painrelief;
  class drug;
```

```
                         The NPAR1WAY Procedure

   Median Scores (Number of Points Above Median) for Variable painrelief
                        Classified by Variable drug

                        Sum of      Expected       Std Dev          Mean
   drug        N        Scores      Under H0      Under H0         Score

   druga       9      0.333333      4.333333      1.117078      0.037037
   drugb       9      5.666667      4.333333      1.117078      0.629630
   drugc       9      7.000000      4.333333      1.117078      0.777778

                    Average scores were used for ties.
                       Median One-Way Analysis

                        Chi-Square         13.2968
                        DF                       2
                        Pr > Chi-Square     0.0013
```

The P-value is a little bigger than came out of the F-test, but the conclusion is still that there are definitely differences among the drugs in terms of pain relief. The table at the top of the output again suggests that drug A is worse than the others, but to confirm that you'd have to do Mood's median test on all three *pairs* of drugs, and then use Bonferroni to allow for your having done three tests.

More extra: the test statistic and P-value are slightly different in SAS than in R, but not enough to change the decision (or, indeed, enough to change the P-value from being bigger than the ANOVA but still significant). This is because of how SAS counts values exactly equal to the overall median: rather than ignoring them, it counts them as a fraction of an observation above the overall median. When we analyzed these data with R, we found that six observations were exactly equal to the median (one for drug A, two for drug B and three for drug C), 11 were strictly above the median and 10 were strictly below. This means that ranks 12 through 17 (counting from the high end) were observations that were actually equal to the median. Ranks 12 and 13, that is, two out of the six, would actually be above the median if the values were all distinct. Thus, as far as SAS is concerned, each observation counts as $2/6 = 1/3$ above and $2/3$ below-or-equal. So when figuring out the "scores", SAS counts $1/3$ for each observation exactly equal to the median, and thus an extra $1/3$ for drug A, $2/3$ for drug B and $3/3 = 1$ for drug C.

The effect of this difference between how SAS does it and how `smmr` does it in R is usually pretty small, especially in terms of the conclusion you end up drawing, but the answers won't be exactly the same.

10.4. My cars data file can be found at `http://www.utsc.utoronto.ca/~butler/c32/cars.csv`. This question is the same one we did before using R, so you should be able to see yourself get the same results both ways (comparing your results from before). The values in the data file are separated by commas; the car names are up to 29 characters long. For all the parts after (a), creating and displaying a new data set each time, or (where applicable) using `where`.

(a) Read the data into SAS and list the values. (Listing them all is OK for yourself.)

Solution:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/cars.csv";

proc import
  datafile=myurl
  dbms=csv
  out=cars
  replace;
  getnames=yes;


proc print;
```

```
             Obs    car                            MPG         weight

               1    Buick Skylark                 28.4           2.67
               2    Dodge Omni                    30.9           2.23
               3    Mercury Zephyr                20.8           3.07
               4    Fiat Strada                   37.3           2.13
               5    Peugeot 694 SL                16.2           3.41
               6    VW Rabbit                     31.9          1.925
               7    Plymouth Horizon              34.2            2.2
               8    Mazda GLC                     34.1          1.975
               9    Buick Estate Wagon            16.9           4.36
              10    Audi 5000                     20.3           2.83
              11    Chevy Malibu Wagon            19.2          3.605
              12    Dodge Aspen                   18.6           3.62
              13    VW Dasher                     30.5           2.19
              14    Ford Mustang 4                26.5          2.585
              15    Dodge Colt                    35.1          1.915
              16    Datsun 810                      22          2.815
              17    VW Scirocco                   31.5           1.99
              18    Chevy Citation                28.8          2.595
              19    Olds Omega                    26.8            2.7
              20    Chrysler LeBaron W            18.5           3.94
              21    Datsun 510                    27.2            2.3
              22    AMC Concord D/L               18.1           3.41
              23    Buick Century Spec            20.6           3.38
              24    Saab 99 GLE                   21.6          2.795
              25    Datsun 210                    31.8           2.02
              26    Ford LTD                      17.6          3.725
              27    Volvo 240 GL                    17           3.14
              28    Dodge St Regis                18.2           3.83
              29    Toyota Corona                 27.5           2.56
              30    Chevette                        30          2.155
              31    Ford Mustang Ghia             21.9           2.91
              32    AMC Spirit                    27.4           2.67
              33    Ford Country Squir            15.5          4.054
              34    BMW 320i                      21.5            2.6
              35    Pontiac Phoenix               33.5          2.556
              36    Honda Accord LX               29.5          2.135
              37    Mercury Grand Marq            16.5          3.955
              38    Chevy Caprice Clas              17           3.84

             Obs       cylinders            hp    country

               1           4                90    U.S.
               2           4                75    U.S.
               3           6                85    U.S.
               4           4                69    Italy
               5           6               133    France
               6           4                71    Germany
               7           4                70    U.S.
               8           4                65    Japan
               9           8               155    U.S.
              10           5               103    Germany
              11           8               125    U.S.
              12           6               110    U.S.
              13           4                78    Germany
              14           4                88    U.S.
              15           4                80    Japan
              16           6                97    Japan
              17           4                71    Germany
              18           6               115    U.S.
              19           6               115    U.S.
              20           8               150    U.S.
              21           4                97    Japan
              22           6               120    U.S.
              23           6               105    U.S.
              24           4               115    Sweden
              25           4                65    Japan
              26           8               129    U.S.
              27           6               125    Sweden
              28           8               135    U.S.
              29           4                95    Japan
              30           4                68    U.S.
              31           6               109    U.S.
              32           4                80    U.S.
              33           8               142    U.S.
              34           4               110    Germany
              35           4                90    U.S.
              36           4                68    Japan
              37           8               138    U.S.
              38           8               130    U.S.
```

It wanted to display the data in two blocks, but it's all there.

On your Results window, you might have to scroll down a ways, but likewise it should be all there somewhere.

You might be wondering why we got all of `Chevy Malibu Wagon` but not all of `Ford Country Squire Wagon` (and some of the others). The reason appears to be this: SAS uses the first few rows of the data file (default 20) to figure out how long names are. The longest name it found in there is the Chevy wagon, so it thought that none of the names were longer than that, and when it got to the Ford wagon, it read as far as the length of the Chevy wagon and stopped there. So, you're thinking "why not update the length as you go, with the longest one found so far?" When SAS was designed, pieces of text were of a fixed length that you had to specify ahead of time, which is still part of the language specification, so what `proc import` does is to read the file part of the way down (the default is 20 lines) to find the maximum length of each text variable. Then it treats each text variable as being that long maximum, and goes back to the beginning and reads it "for real". The reason for looking only at a few lines is that if you have a big file, with millions of lines, say, you wouldn't want to scan the whole thing twice. One fix for this is to move the long pieces of text to the beginning of the data file (having read the data in once and found that a lot of the names are getting cut off). Another is to set the `guessingrows` option to something larger in `proc import`. There were just under 40 cars, so this ought to work:

```
proc import
  datafile=myurl
  dbms=csv
  out=cars
  replace;
  getnames=yes;
  guessingrows=40;

proc print;
  var car MPG;
```

```
                Obs    car                                 MPG

                  1    Buick Skylark                      28.4
                  2    Dodge Omni                         30.9
                  3    Mercury Zephyr                     20.8
                  4    Fiat Strada                        37.3
                  5    Peugeot 694 SL                     16.2
                  6    VW Rabbit                          31.9
                  7    Plymouth Horizon                   34.2
                  8    Mazda GLC                          34.1
                  9    Buick Estate Wagon                 16.9
                 10    Audi 5000                          20.3
                 11    Chevy Malibu Wagon                 19.2
                 12    Dodge Aspen                        18.6
                 13    VW Dasher                          30.5
                 14    Ford Mustang 4                     26.5
                 15    Dodge Colt                         35.1
                 16    Datsun 810                           22
                 17    VW Scirocco                        31.5
                 18    Chevy Citation                     28.8
                 19    Olds Omega                         26.8
                 20    Chrysler LeBaron Wagon             18.5
                 21    Datsun 510                         27.2
                 22    AMC Concord D/L                    18.1
                 23    Buick Century Special              20.6
                 24    Saab 99 GLE                        21.6
                 25    Datsun 210                         31.8
                 26    Ford LTD                           17.6
                 27    Volvo 240 GL                         17
                 28    Dodge St Regis                     18.2
                 29    Toyota Corona                      27.5
                 30    Chevette                             30
                 31    Ford Mustang Ghia                  21.9
                 32    AMC Spirit                         27.4
                 33    Ford Country Squire Wagon          15.5
                 34    BMW 320i                           21.5
                 35    Pontiac Phoenix                    33.5
                 36    Honda Accord LX                    29.5
                 37    Mercury Grand Marquis              16.5
                 38    Chevy Caprice Classic                17
```

and this time the Ford wagon's full name is read in (on line 33).

(b) Display only the car names and the countries they come from.

Solution:

```
data cars2;
  set cars;
  keep car country;



proc print;
```

```
                 Obs    car                            country

                   1    Buick Skylark                  U.S.
                   2    Dodge Omni                     U.S.
                   3    Mercury Zephyr                 U.S.
                   4    Fiat Strada                    Italy
                   5    Peugeot 694 SL                 France
                   6    VW Rabbit                      Germany
                   7    Plymouth Horizon               U.S.
                   8    Mazda GLC                      Japan
                   9    Buick Estate Wagon             U.S.
                  10    Audi 5000                      Germany
                  11    Chevy Malibu Wagon             U.S.
                  12    Dodge Aspen                    U.S.
                  13    VW Dasher                      Germany
                  14    Ford Mustang 4                 U.S.
                  15    Dodge Colt                     Japan
                  16    Datsun 810                     Japan
                  17    VW Scirocco                    Germany
                  18    Chevy Citation                 U.S.
                  19    Olds Omega                     U.S.
                  20    Chrysler LeBaron Wagon         U.S.
                  21    Datsun 510                     Japan
                  22    AMC Concord D/L                U.S.
                  23    Buick Century Special          U.S.
                  24    Saab 99 GLE                    Sweden
                  25    Datsun 210                     Japan
                  26    Ford LTD                       U.S.
                  27    Volvo 240 GL                   Sweden
                  28    Dodge St Regis                 U.S.
                  29    Toyota Corona                  Japan
                  30    Chevette                       U.S.
                  31    Ford Mustang Ghia              U.S.
                  32    AMC Spirit                     U.S.
                  33    Ford Country Squire Wagon      U.S.
                  34    BMW 320i                       Germany
                  35    Pontiac Phoenix                U.S.
                  36    Honda Accord LX                Japan
                  37    Mercury Grand Marquis          U.S.
                  38    Chevy Caprice Classic          U.S.
```

There is no `where` option here, because `where` is for doing something with certain *rows*, not columns.

(c) Display everything *except* horsepower:

Solution: Naming what you *don't* want is sometimes easier:

```
data cars3;
  set cars;
  drop hp;
```

```
proc print;
```

```
 Obs car                                 MPG       weight    cylinders country

   1 Buick Skylark                      28.4        2.67            4 U.S.
   2 Dodge Omni                         30.9        2.23            4 U.S.
   3 Mercury Zephyr                     20.8        3.07            6 U.S.
   4 Fiat Strada                        37.3        2.13            4 Italy
   5 Peugeot 694 SL                     16.2        3.41            6 France
   6 VW Rabbit                          31.9       1.925            4 Germany
   7 Plymouth Horizon                   34.2         2.2            4 U.S.
   8 Mazda GLC                          34.1       1.975            4 Japan
   9 Buick Estate Wagon                 16.9        4.36            8 U.S.
  10 Audi 5000                          20.3        2.83            5 Germany
  11 Chevy Malibu Wagon                 19.2       3.605            8 U.S.
  12 Dodge Aspen                        18.6        3.62            6 U.S.
  13 VW Dasher                          30.5        2.19            4 Germany
  14 Ford Mustang 4                     26.5       2.585            4 U.S.
  15 Dodge Colt                         35.1       1.915            4 Japan
  16 Datsun 810                           22       2.815            6 Japan
  17 VW Scirocco                        31.5        1.99            4 Germany
  18 Chevy Citation                     28.8       2.595            6 U.S.
  19 Olds Omega                         26.8         2.7            6 U.S.
  20 Chrysler LeBaron Wagon             18.5        3.94            8 U.S.
  21 Datsun 510                         27.2         2.3            4 Japan
  22 AMC Concord D/L                    18.1        3.41            6 U.S.
  23 Buick Century Special              20.6        3.38            6 U.S.
  24 Saab 99 GLE                        21.6       2.795            4 Sweden
  25 Datsun 210                         31.8        2.02            4 Japan
  26 Ford LTD                           17.6       3.725            8 U.S.
  27 Volvo 240 GL                         17        3.14            6 Sweden
  28 Dodge St Regis                     18.2        3.83            8 U.S.
  29 Toyota Corona                      27.5        2.56            4 Japan
  30 Chevette                           30         2.155            4 U.S.
  31 Ford Mustang Ghia                  21.9        2.91            6 U.S.
  32 AMC Spirit                         27.4        2.67            4 U.S.
  33 Ford Country Squire Wagon          15.5       4.054            8 U.S.
  34 BMW 320i                           21.5         2.6            4 Germany
  35 Pontiac Phoenix                    33.5       2.556            4 U.S.
  36 Honda Accord LX                    29.5       2.135            4 Japan
  37 Mercury Grand Marquis              16.5       3.955            8 U.S.
  38 Chevy Caprice Classic                17        3.84            8 U.S.
```

(d) Display only the cars that have 8-cylinder engines (but display all the variables for those cars).

Solution: This:

```
data cars4;
  set cars;
  if cylinders=8;



proc print;
```

```
        Obs    car                                MPG          weight

         1     Buick Estate Wagon                16.9            4.36
         2     Chevy Malibu Wagon                19.2           3.605
         3     Chrysler LeBaron Wagon            18.5            3.94
         4     Ford LTD                          17.6           3.725
         5     Dodge St Regis                    18.2            3.83
         6     Ford Country Squire Wagon         15.5           4.054
         7     Mercury Grand Marquis             16.5           3.955
         8     Chevy Caprice Classic               17            3.84

        Obs       cylinders             hp    country

         1                8            155    U.S.
         2                8            125    U.S.
         3                8            150    U.S.
         4                8            129    U.S.
         5                8            135    U.S.
         6                8            142    U.S.
         7                8            138    U.S.
         8                8            130    U.S.
```

8 of them, all from the US. Note that R and SAS have different ways of specifying a logical "equals": SAS uses one equals sign, while R uses two. (SAS doesn't use `=` for saving in variables, so it's free to do logical equals this way.)

This one, since it is selecting rows, has a `where` option. Put the `where` in the `proc`, which means you don't need to create a new data set (this is a common theme):

```
proc print;
  where cylinders=8;
```

```
        Obs    car                                  MPG         weight

          1    Buick Estate Wagon                  16.9           4.36
          2    Chevy Malibu Wagon                  19.2          3.605
          3    Chrysler LeBaron Wagon              18.5           3.94
          4    Ford LTD                            17.6          3.725
          5    Dodge St Regis                      18.2           3.83
          6    Ford Country Squire Wagon           15.5          4.054
          7    Mercury Grand Marquis               16.5          3.955
          8    Chevy Caprice Classic                 17           3.84

        Obs      cylinders           hp    country

          1          8              155    U.S.
          2          8              125    U.S.
          3          8              150    U.S.
          4          8              129    U.S.
          5          8              135    U.S.
          6          8              142    U.S.
          7          8              138    U.S.
          8          8              130    U.S.
```

If you use a `where` in a `proc`, put it *first*. Things seem to go better this way.

(e) Display the cylinders and horsepower for the cars that have horsepower 70 or less.

Solution: This one is selecting some observations and some variables:

```
data cars4;
  set cars;
  if hp<=70;
  keep cylinders hp;



proc print;
```

```
                Obs      cylinders           hp

                  1          4               69
                  2          4               70
                  3          4               65
                  4          4               65
                  5          4               68
                  6          4               68
```

There should be six cars, including one with horsepower exactly 70 (the Plymouth Horizon). You did do `<=` and not `<`, didn't you?

This one has a `where` option, but you need to be careful: first you need to create a new data set containing the right columns, and then you use `where` in the `proc print` to get the rows:

```
data cars5;
  set cars;
  keep cylinders hp;


proc print;
  where hp<=70;
```

```
                    Obs       cylinders             hp

                      4               4             69
                      7               4             70
                      8               4             65
                     25               4             65
                     30               4             68
                     36               4             68
```

As above, if you were choosing these two variables for only those cars where `mpg>30`, you wouldn't be able to use `where`; you'd have to select rows and columns in one `data` step with a `keep` and an `if`.

(f) Find the mean and SD of gas mileage of the cars with 4 cylinders.

Solution: Strategy:

- Create a data set with just the 4-cylinder cars.
- Run that through `proc means`.

So:

```
data cars5;
  set cars;
  if cylinders=4;




proc means;
  var mpg;
```

```
                            The MEANS Procedure

                          Analysis Variable : MPG

       N            Mean         Std Dev         Minimum         Maximum
      -------------------------------------------------------------------
      19      30.0210526       4.1824473      21.5000000      37.3000000
      -------------------------------------------------------------------
```

The `where` strategy is even easier, since there is no new data set at all:

```
proc means;
  where cylinders=4;
  var mpg;
```

```
                              The MEANS Procedure

                            Analysis Variable : MPG

        N           Mean         Std Dev         Minimum         Maximum
       -----------------------------------------------------------------
       19     30.0210526       4.1824473      21.5000000      37.3000000
       -----------------------------------------------------------------
```

Here is the best strategy to follow for SAS, since it is *only* `proc means`, with nothing else at all:

```
proc means data=cars;
  var mpg;
  class cylinders;
```

```
                              The MEANS Procedure

                            Analysis Variable : MPG

                  N
   cylinders    Obs   N           Mean        Std Dev        Minimum        Maximum
  ------------------------------------------------------------------------------
           4     19  19     30.0210526      4.1824473     21.5000000     37.3000000

           5      1   1     20.3000000              .     20.3000000     20.3000000

           6     10  10     21.0800000      4.0775265     16.2000000     28.8000000

           8      8   8     17.4250000      1.1925363     15.5000000     19.2000000
  ------------------------------------------------------------------------------
```

Results are the same as R. Now you see why one of the SDs was missing: there is only one 5-cylinder car, and you can't calculate an SD from just one number.

I figured I should do `data=cars`, since I have no idea what the current dataset is, and I wanted to make sure that `proc means` used the original one.

This is a very easy kind of question to set on an exam. Unless I am more specific, whatever way you can make it work is good. Just so you know.

10.5. The owner of a lawn mower repair shop is trying three different methods of sharpening lawn mower blades. He records the time (in minutes) it takes to sharpen the blades to an acceptable level using each method. He is concerned that the time may also depend on the type of lawn mower, so he selects mowers of three different types and tests each sharpening method once on each type of mower. There are therefore $3 \times 3 = 9$ observations.

The data are at `http://www.utsc.utoronto.ca/~butler/c32/mowing1.csv`.

(a) Read the data into SAS, and display the data you read in.

Solution: The usual, noting that it is a `.csv` file:

```
filename myurl url 'http://www.utsc.utoronto.ca/~butler/c32/mowing1.csv';

proc import datafile=myurl
  out=mowing
  dbms=csv
  replace;
  getnames=yes;

proc print;
```

```
        Obs    type                 M1            M2            M3

          1    Regular              5.4           4.9           5.2
          2    Mulching             5.2           5.4           5.3
          3    Riding               6.9           6.5           6.2
```

(b) Explain briefly how the data are not tidy.

Solution: The three columns `M1`, `M2`, `M3` all contain sharpening times, and there should be *one* column of these, with a second column labelling method. Or, there should be nine rows, one for each observation. Or there should be columns for each variable, mower type, method and sharpening time, and the methods are split over three columns (which are "levels" of the factor "method"). Anything like that.

(c) Obtain, using SAS, a tidy data set suitable for an analysis of variance. (I will not ask you to do this ANOVA.) Give your code and your output. For full credit, use a SAS array.

Solution: This is actually almost exactly like the one in the lecture notes (so I am giving you an easy time). The only challenge is to map the example in the lecture notes to the one here:

```
data mowing2;
  set mowing;
  array method_array [3] M1-M3;
  do i=1 to 3;
    sharp_time=method_array[i];
    method=vname(method_array[i]);
    output;
  end;
  keep type method sharp_time;

proc print;
```

Start by creating a new data set and bringing in everything from the data set you read in from the file.

The array needs to contain the three methods, and the loop loops over these three methods (not four like the example in class). You need to pick out the sharpening time from the array, and call it something reasonable (*not* `method`!) and you need to pick out the *name* of the method and call that something reasonable (which probably *should* be `method`). Then output that to the new data set, and keep only the three variables you need, `type`, `method` and sharpening time (that is, you need to get rid of `M1` through `M3`).

Did it work?

```
                          sharp_
        Obs    type         time     method

         1     Regular       5.4       M1
         2     Regular       4.9       M2
         3     Regular       5.2       M3
         4     Mulching      5.2       M1
         5     Mulching      5.4       M2
         6     Mulching      5.3       M3
         7     Riding        6.9       M1
         8     Riding        6.5       M2
         9     Riding        6.2       M3
```

That looks good.

If you prefer, you can also do it what I called “the tedious way” in the lecture notes, but expect to lose a mark since I asked you to use an array. It’s also very easy to make small mistakes in the “tedious way”, which, if you are not very careful, will happen to you.

I suppose I have to show you the tedious way as well (which I then have to get right myself!):

```
data mowing3;
  set mowing;
  sharp_time=M1;
  method='M1';
  output;
  sharp_time=M2;
  method='M2';
  output;
  sharp_time=M3;
  method='M3';
  output;
  keep type method sharp_time;

proc print;
```

Success?

```
                      sharp_
 Obs    type           time     method

  1     Regular         5.4       M1
  2     Regular         4.9       M2
  3     Regular         5.2       M3
  4     Mulching        5.2       M1
  5     Mulching        5.4       M2
  6     Mulching        5.3       M3
  7     Riding          6.9       M1
  8     Riding          6.5       M2
  9     Riding          6.2       M3
```

I think so.

The implication for both of these methods is a loop over each row of the (original) data set, so when you are thinking about what's going on, keep in mind that you are looking at a particular row of what I called `mowing`. In the row you are looking at, you pull out `M1, M2, M3` and save them one at a time in the new data set (the value and the name). Thus the array method is *actually* a loop within a loop.

Now, I said that I wasn't going to ask you do the ANOVA here. That's because it's a different flavour of ANOVA than we have seen: we are used to one grouping variable, and here we have two, mower type and sharpening method. When you have two grouping variables (explanatory factors), you're into the world of two-way analysis of variance, which you will explore more with me in D29, if you take that. You might have seen it in B27, if you took that course. I can't remember whether you do it there or not.

This one is a simple form called "randomized blocks", which I think is simple enough to talk about here. The "blocks" part of the experimental design is that the owner of the repair shop really cares about whether `method` makes a difference, and how. The type of lawn mower is not of primary interest, but it might make a difference to the sharpening time. A general principle of modelling is that you should include everything that makes a difference, whether you care about it or not, because then you will get a better test for what you *do* care about. The thing that you don't really care about, here `type`, is called a **block** or **blocking factor**, and the effect it is assumed to have is that it moves sharpening time up or down the same for all the methods, a so-called "additive" effect. That's about the only thing you can deal with when you have as small an amount of data as you have here: just one observation per treatment-block combination. (If one method is good for one type of lawn mower and another method is better for a different type, then we're dealing with **interaction**, and we need two or more observations per method-type combination to be able to deal with that. That's beyond us now, though.)

This is the same kind of idea as the issue of why it is good to do matched pairs when you can. If you think about our baseball-softball example, the distance thrown might vary because different balls are used (a ball effect), but it might also vary because different *students* were doing the throwing. In the guise of randomized blocks, we think of students as blocks, because we expect them to be different one from another, and we don't really care about that. The balls are the "treatments", and we want to know whether there is a ball effect after allowing for a possible student effect. Matched pairs has a handy get-out so that we don't have to think about things this way, though: we take the difference baseball minus softball for each student, so that we naturally have a measure of how much farther the *same* student throws the baseball. In other situations, such as the one in this question, we really do need to compute a sum of squares for blocks to "get it out of the way". See below.

The analysis here is `proc anova` and it looks a lot like what we've already done. This time, though, we have *two* categorical variables to go on the right side of the `model` line, two F-tests to consider, and possibly two Tukeys to look at, thus:

```
proc anova;
  class method type;
  model sharp_time=method type;
```

```
                          The ANOVA Procedure

                        Class Level Information

              Class         Levels    Values

              method             3    M1 M2 M3

              type               3    Mulching Regular Riding
```

```
                    Number of Observations Read          9
                    Number of Observations Used          9
                              The ANOVA Procedure

                       Dependent Variable: sharp_time

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

Model                        4         3.53333333     0.88333333     13.25   0.0141

Error                        4         0.26666667     0.06666667

Corrected Total              8         3.80000000
           R-Square     Coeff Var      Root MSE    sharp_time Mean

           0.929825      4.556451      0.258199           5.666667
Source                      DF          Anova SS    Mean Square   F Value   Pr > F

method                       2        0.12666667     0.06333333      0.95   0.4596
type                         2        3.40666667     1.70333333     25.55   0.0053
```

The action is at the bottom of this. The real interest is in whether the sharpening methods are different, and the answer is that they are not, even after allowing for possible differences due to type of mower. That's normally the end of it, because we weren't really interested in *testing* whether the types of mower made a difference: we were taking it for granted that they did. This is not really interesting.

If the methods showed a significant difference, we would follow up with Tukey. I wouldn't really do that here, but I can for illustration:

```
proc anova;
  class method type;
  model sharp_time=method type;
  means method / tukey;
  means type / tukey;
```

```
                              The ANOVA Procedure

              Tukey's Studentized Range (HSD) Test for sharp_time

                  Alpha                                   0.05
                  Error Degrees of Freedom                   4
                  Error Mean Square                   0.066667
                  Critical Value of Studentized Range  5.04016
                  Minimum Significant Difference        0.7513
            Means with the same letter are not significantly different.


            Tukey Grouping          Mean      N    method

                        A        5.8333      3    M1
                        A
                        A        5.6000      3    M2
                        A
                        A        5.5667      3    M3
                  Alpha                                   0.05
                  Error Degrees of Freedom                   4
                  Error Mean Square                   0.066667
                  Critical Value of Studentized Range  5.04016
                  Minimum Significant Difference        0.7513
            Means with the same letter are not significantly different.


           Tukey Grouping          Mean      N    type

                       A        6.5333      3    Riding

                       B        5.3000      3    Mulching
                       B
                       B        5.1667      3    Regular
```

The Tukey for `methods` shows no significant differences, not a surprise (we really shouldn't even be looking at it). For the mower types, the Riding type takes longer to sharpen than the other two types, which do not differ significantly from each other.[38]

A final note is that by having the same number of observations per `method-type` combination, we can fairly compare the methods with each other by looking at their means (since all the methods were tested on all the types), and we can fairly compare the types with each other by looking at *their* means, since they are compared over all the methods. This principle of having the same number of observations per combination is an important one in experimental design, and makes the analysis a lot easier to cope with.[39]

10.6. The data in `http://www.utsc.utoronto.ca/~butler/c32/expgrow.txt` are for two variables, mysteriously called `xx` and `yy`, in that order, with `yy` being the response variable.

(a) Read the data into SAS and list out the values (there are only a few of them).

Solution: This kind of thing. The variable names are indeed on the first line of the file:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/expgrow.txt";

proc import
  datafile=myurl
  dbms=dlm
  out=xy
  replace;
  delimiter=' ';
  getnames=yes;

proc print;
```

```
                    Obs          xx          yy

                      1           5         0.5
                      2           6           1
                      3           7         2.1
                      4           8         4.2
                      5           9         8.3
                      6          10        16.7
```

(b) What do you notice about the `yy` values (at least approximately)? (The `xx` values go up in steps of 1.)

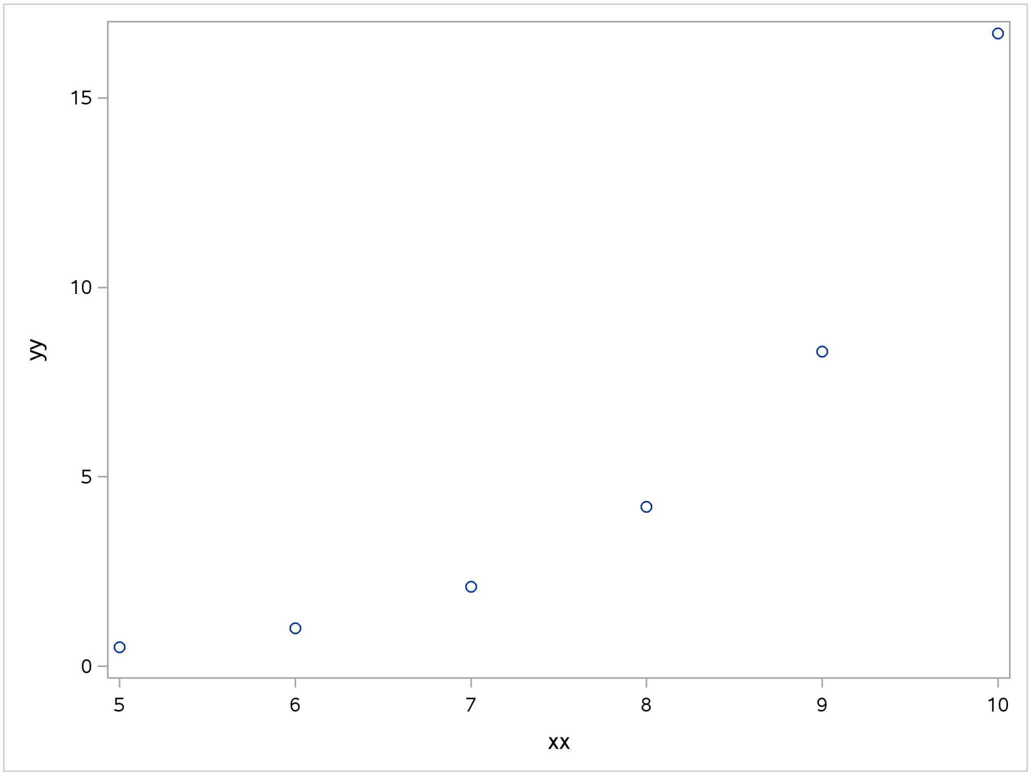
Solution: Each one is almost exactly double the one before it (the one before it times two).[40]

(c) Make a scatterplot for predicting `yy` from `xx`.

Solution: The usual way:

```
proc sgplot;
  scatter x=xx y=yy;
```

giving



I originally called my variables `x` and `y`, which made the `scatter` line of code very confusing!

(d) Do you think a straight line would fit well here? Explain briefly.

Solution: I don't think so. The rate of increase seems to get bigger as we move further to the right. In fact, it looks like exponential growth.

The mathematicians among you will have clued in already that this is in fact the case.

(e) Fit a linear regression and obtain a residual plot (the residual plot will be part of the output you obtain with the regression). On the evidence of this output, why do you think a linear regression would be a bad idea?

Solution:

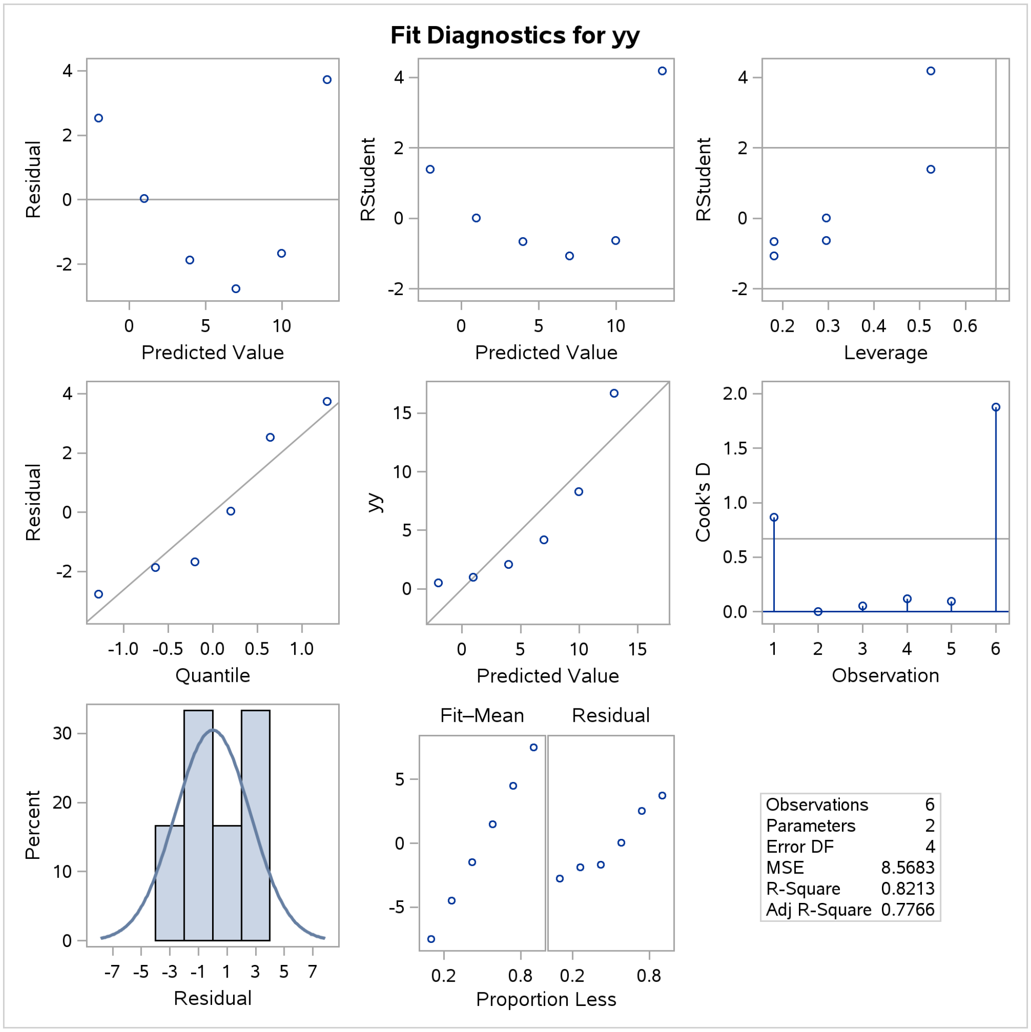
```
proc reg;
  model yy=xx;
```

```
                    The REG Procedure
                      Model: MODEL1
                  Dependent Variable: yy

          Number of Observations Read          6
          Number of Observations Used          6
```

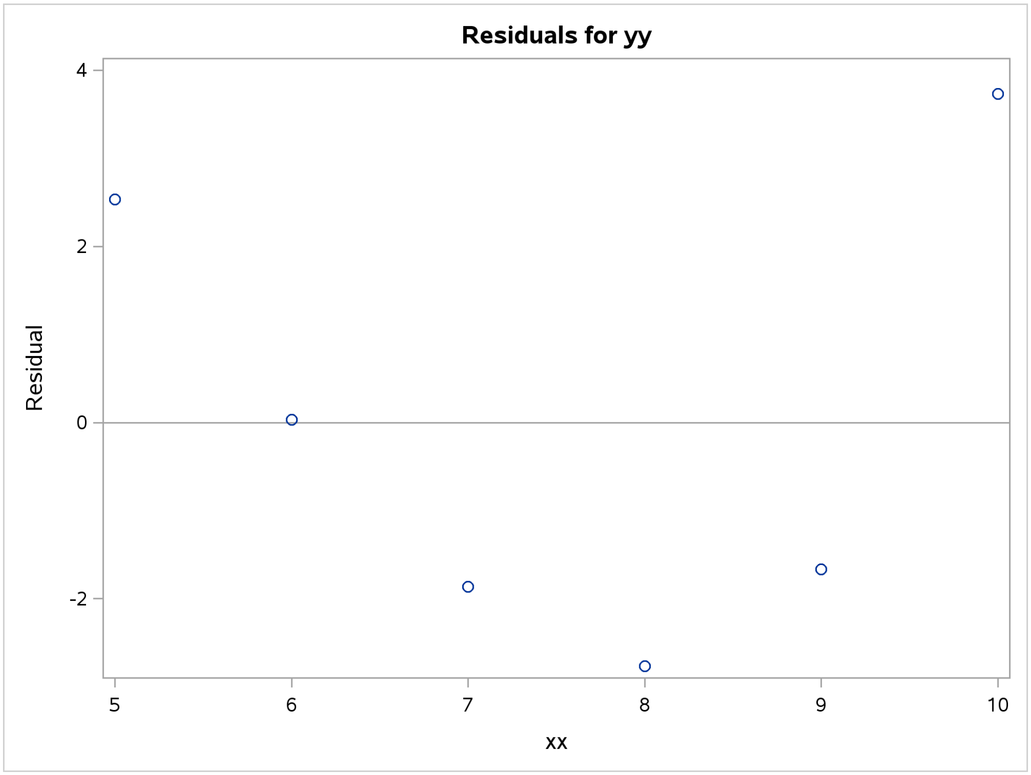
```
                            Analysis of Variance

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

Model                       1      157.50000      157.50000      18.38    0.0128
Error                       4       34.27333        8.56833
Corrected Total             5      191.77333
             Root MSE              2.92717    R-Square     0.8213
             Dependent Mean        5.46667    Adj R-Sq     0.7766
             Coeff Var            53.54582
                            Parameter Estimates

                             Parameter       Standard
     Variable     DF          Estimate          Error    t Value    Pr > |t|

     Intercept     1         -17.03333        5.38230      -3.16      0.0340
     xx            1           3.00000        0.69973       4.29      0.0128
```

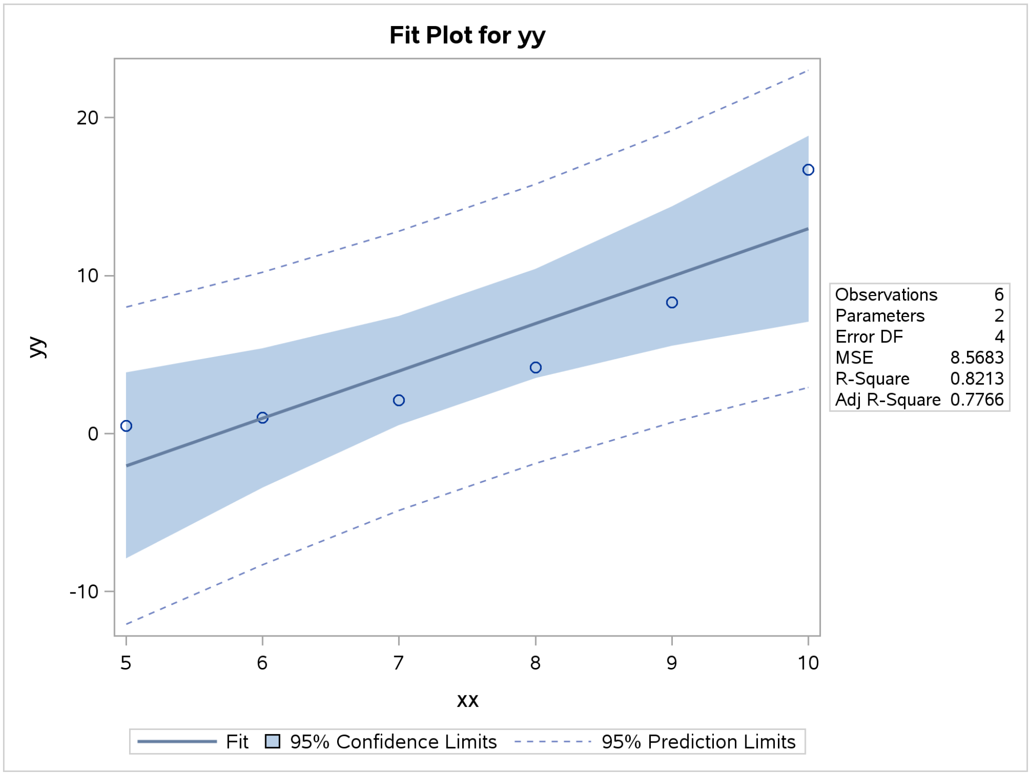
R-squared at 0.82 is pretty high, and the slope is significantly nonzero and positive. This *is* an upward trend, after all. But is a straight line best, or can we do better?

Here are the graphs that come out of the regression:



```
Observations        6
Parameters          2
Error DF            4
MSE            8.5683
R-Square       0.8213
Adj R-Square   0.7766
```





Observations	6
Parameters	2
Error DF	4
MSE	8.5683
R-Square	0.8213
Adj R-Square	0.7766

This residual plot (top left of the 3×3 array of plots) shows a clear curved pattern, indicating (as we thought before) that a curved relationship was happening here.

We don't need to look at the other plots, having found a problem already, but: the second plot in the first row is a different form of residuals (that sometimes shows things more clearly, but gives the same story here), and the third picture plots the leverages (how unusual the x is) against the residuals.

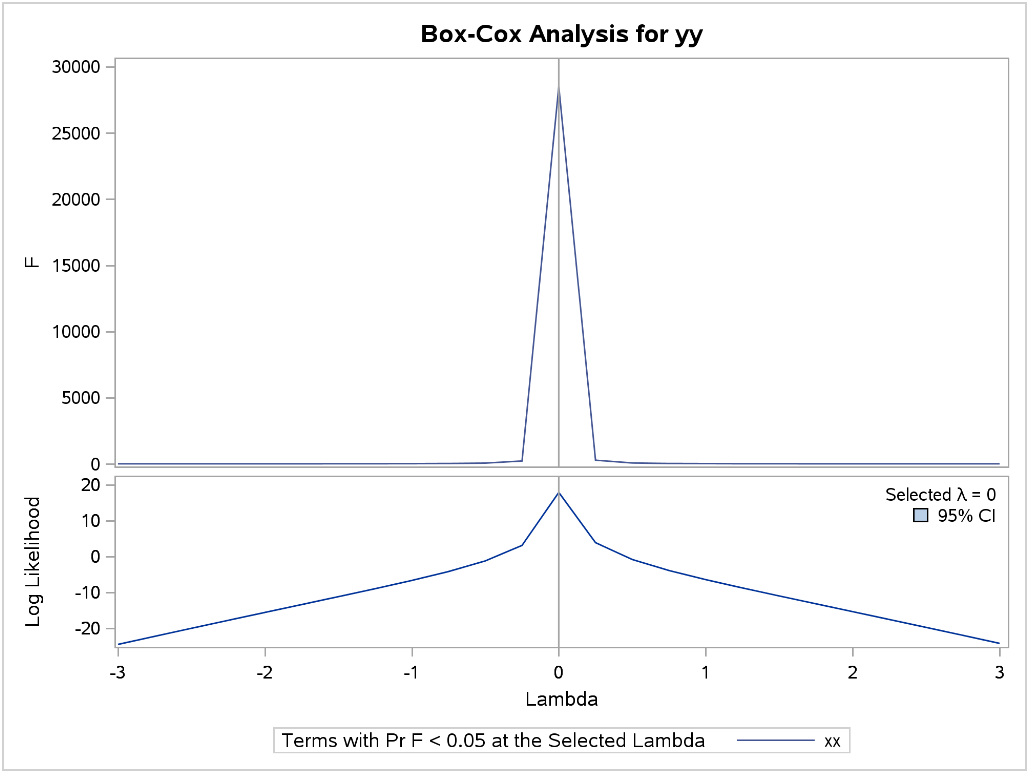
On the second row, the most important thing is the first picture, a normal quantile plot of the residuals. The residuals should have a normal distribution (approximately), and here, even with all the other problems, they do. Below that is a histogram with superimposed normal curve of the residuals; given that there are only six of them, they look shoulder-shruggingly normal.

But, that curve on the residual plot, indicating a curved relationship, is what we need to address.

(f) Use Box-Cox to determine an appropriate transformation for `y`. What do you get?

Solution: This calls for `proc transreg`:

```
proc transreg;
  model boxcox(yy)=identity(xx);
```



The bottom graph shows a graph with a clear peak at $\lambda = 0$. So, replacing `y` with the log of `y` would be expected to do a great job of straightening out the relationship.

Mathematically, this is in fact not surprising at all. Exponential growth would have a formula like $y = ae^{bx}$ where a and b are coefficients to estimate. If we take logs of both sides, we get $\ln y = \ln(ae^{bx})$ or $\ln y = \ln a + bx$, which says that $\ln y$ has a linear relationship with x. That is, if y and x have an exponential growth[41] relationship, the log of y will have a linear relationship with x.

(g) Create a new data set with the transformed `y` in it, and plot the appropriately-transformed `y` against `x`. Is it straight now?

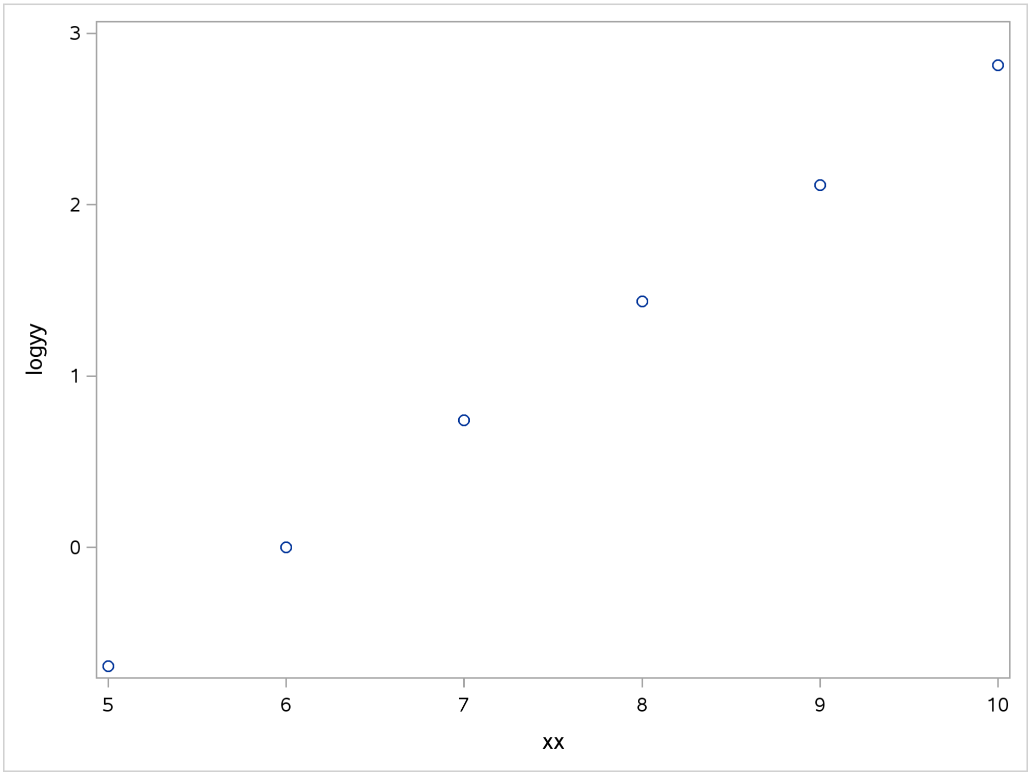
Solution: Create the new data set first, which will then become the "default" one, so that you don't have to refer to it by name:

```
data xynew;
  set xy;
  logyy=log(yy);
```

First the new data set (you can check that those are base e logs in the last column), and then the graph:

```
proc print;

proc sgplot;
  scatter x=xx y=logyy;
```

```
                Obs              xx              yy       logyy

                 1                5             0.5    -0.69315
                 2                6               1     0.00000
                 3                7             2.1     0.74194
                 4                8             4.2     1.43508
                 5                9             8.3     2.11626
                 6               10            16.7     2.81541
```



That's straight all right.

(h) Confirm that the transformation has straightened out the relationship by fitting a regression and by obtaining a residual plot.

Solution: Use the transformed variable as the response:

```
proc reg;
  model logyy=xx;
```

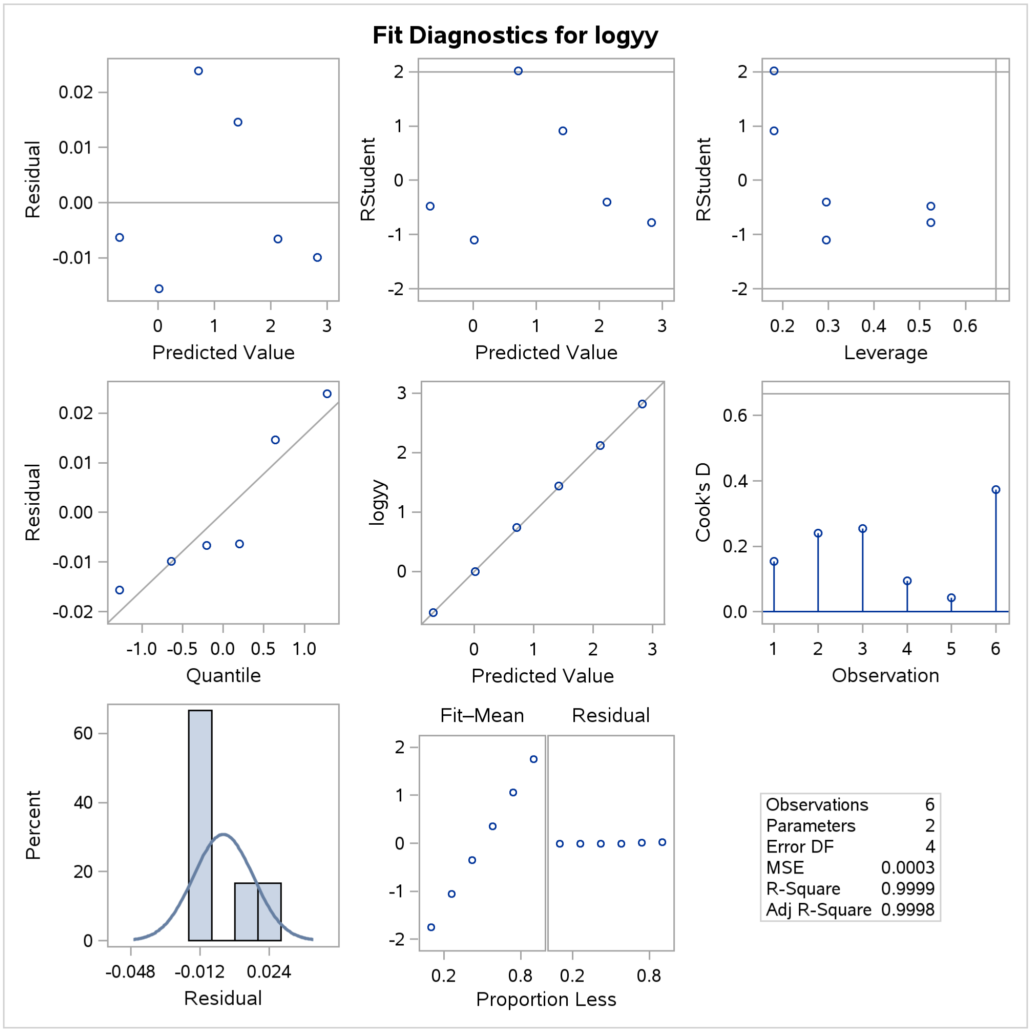
My regression output is

```
                                 The REG Procedure
                                  Model: MODEL1
                            Dependent Variable: logyy

                       Number of Observations Read           6
                       Number of Observations Used           6
                                Analysis of Variance

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

Model                      1           8.63439        8.63439    28521.0    <.0001
Error                      4           0.00121     0.00030274
Corrected Total            5           8.63560
              Root MSE                 0.01740    R-Square     0.9999
              Dependent Mean           1.06926    Adj R-Sq     0.9998
              Coeff Var                1.62724
```

```
                          Parameter Estimates

                         Parameter       Standard
    Variable     DF       Estimate          Error    t Value    Pr > |t|

    Intercept     1       -4.19889        0.03199    -131.24      <.0001
    xx            1        0.70242        0.00416     168.88      <.0001
```

The R-squared is 0.9998, as high as you could ever wish for. This is basically a perfect fit.

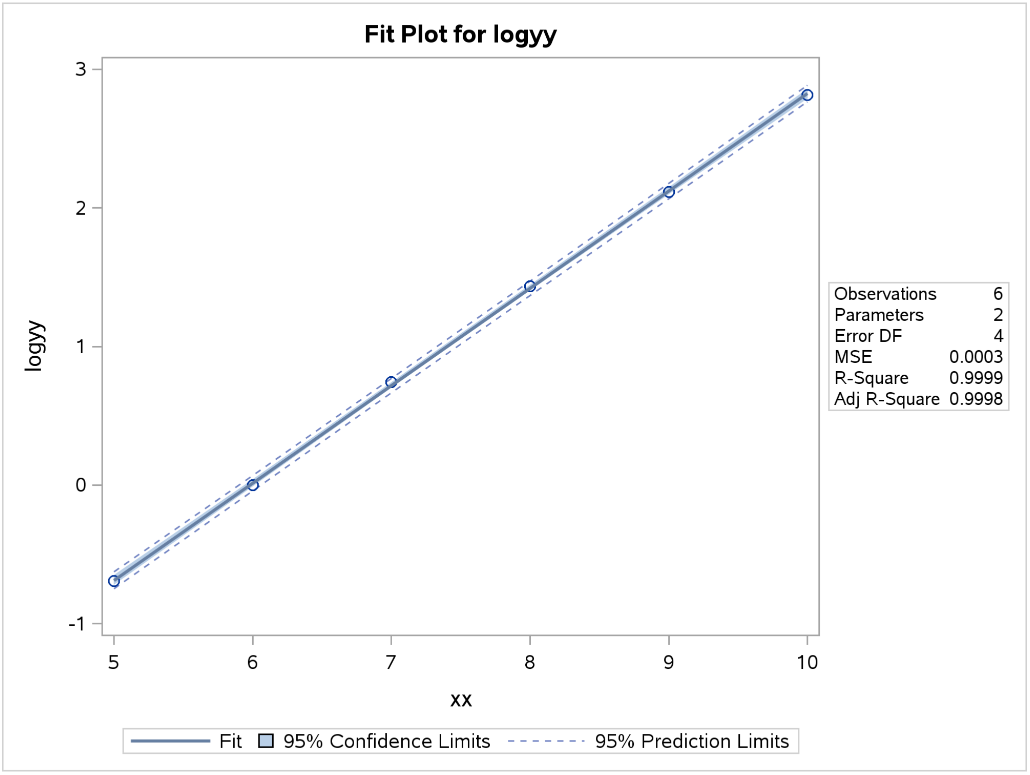
My residual plot is at the top left of



Observations	6
Parameters	2
Error DF	4
MSE	0.0003
R-Square	0.9999
Adj R-Square	0.9998

The residual plot still appears to have something of a pattern to it, but look at the vertical scale. The actual values of `y` go up to 275, but the size of the residuals[42] on the log scale is *tiny*: no more than 0.03. So the predictions are alarmingly accurate, and I don't think we need to worry about any shape in those tiny residuals.[43]

You also get a "fit plot" at the end: a scatterplot with the fitted line on it. Here, you see how alarmingly good the fit is:



(i) Use a calculator or a spreadsheet (or R) to predict the value of `y` when `x` is 5. How close are you to the true value?

Solution: One of the students noticed that the intercept and slope I was using here were not the same as the ones that SAS got. This, right here, is the danger of copying and pasting, which is what I did (and didn't notice that I had the wrong values). So, let's use R as a calculator, but first let's read in the data again and run the regression again, to be sure we're doing the same thing:

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/expgrow.txt"
xy=read_delim(my_url, " ")

## Parsed with column specification:
## cols(
##   xx = col_double(),
##   yy = col_double()
## )

xy

## # A tibble: 6 x 2
##       xx    yy
##    <dbl> <dbl>
## 1      5   0.5
## 2      6   1
## 3      7   2.1
## 4      8   4.2
## 5      9   8.3
## 6     10  16.7
```

Now, we run the regression predicting log of `yy` from `xx` and didisplay the results. You can do it this way rather than creating a new column with the log-values in it (though there's no harm in doing it that way if you prefer):

```
xy.1=lm(log(yy)~xx, data=xy)
summary(xy.1)

##
## Call:
## lm(formula = log(yy) ~ xx, data = xy)
##
## Residuals:
##         1         2         3         4         5         6
## -0.006354 -0.015627  0.023891  0.014618 -0.006631 -0.009897
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.198892   0.031993  -131.2 2.02e-08 ***
## xx           0.702420   0.004159   168.9 7.37e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0174 on 4 degrees of freedom
## Multiple R-squared:  0.9999,Adjusted R-squared:  0.9998
## F-statistic: 2.852e+04 on 1 and 4 DF,  p-value: 7.374e-09
```

Checking carefully, that's the same thing that SAS got. So now we can pull out the intercept and slope *without* retyping them:

```
b=coefficients(xy.1)
b

## (Intercept)          xx
##  -4.1988921   0.7024198
```

Now let's predict yy when xx is 5:[44]

```
pred=b[1]+5*b[2]
pred

## (Intercept)
##   -0.686793
```

This is the predicted log-y, so we have to anti-log it. The function that does this is usually called `exp` (on a scientific calculator or in a spreadsheet or in R):

```
exp(pred)

## (Intercept)
##   0.5031872
```

Going back to the original data set, the true value of `y` when `x=5` is 0.5. Our prediction of 0.503 is really very close, off by this percent:

```
(0.503-0.500)/0.500*100

## [1] 0.6
```

less than one percent. (I copied those numbers, which I shouldn't have done, because if I change the data, these will change again.)

This is where I should probably come clean and say that I made up these data to illustrate how things work. Sometimes it's nice to use data where you can see[45] what's going on, to convince yourself that the result is what you would expect.[46] I should probably also some clean and say that I appear to have two data sets like this, and I used the wrong one here the first time.

11 Regression

11.1. A management consultancy obtained data on salaries and other work information of 100 company executives (from different companies). Their aim was to predict salary from some or all of the other variables (and to determine which of those other variables are important determinants of salary). The data are in `http://www.utsc.utoronto.ca/~butler/c32/execsal.xlsx`, as an Excel spreadsheet, with the columns being (respectively):

- Row number (ignore)
- Log of annual salary
- experience (years)
- education (years)
- gender (1=male, 0=female)
- number of employees supervised
- corporate assets (millions of dollars)
- board member (1=yes, 0=no)
- age (years)
- company profits (past 12 months, millions of dollars)

- has international responsibility (1=yes, 0=no)
- company's total sales (past 12 months, millions of dollars)

The consultancy used log of salary because the relationship with other variables (in previous studies) seemed to be straighter. (A consequence of using logs is that a one-unit increase in any of the other variables is associated with a certain *percentage* increase in annual salary, which often makes sense.) Note that the data set already contains a variable `logsal`, which is the log-salary, so you don't need to create one.

(a) Read the data into SAS, bearing in mind the format of the data. You'll need to know the name of the sheet you want to read in. Also, reading an Excel file only works "locally": that is, you'll need to grab your own copy of the spreadsheet and upload it to SAS Studio.

Solution: First, open the spreadsheet and take a look at it. The sheet you want is called `execsal2`. Save it somewhere on your computer and then upload it to SAS Studio.

Then, find a previous `proc import` with `dbms=xlsx`, and adapt it to what you need, replacing my username with yours:

```
proc import
  datafile='/home/ken/execsal.xlsx'
  dbms=xlsx
  out=salaries
  replace;
  sheet=execsal2;
  getnames=yes;
```

Or remember DODRG and this time note that you need an extra S for "sheet".

I ran that through `proc print` to check that I had the right thing, and I did. Or you can summarize:

```
proc means;
```

```
                              The MEANS Procedure

 Variable  Label      N          Mean       Std Dev       Minimum       Maximum
 ------------------------------------------------------------------------------
 row       row      100    50.5000000    29.0114920     1.0000000   100.0000000
 logsal    logsal   100    11.4550180     0.2598109    10.6643000    12.0634000
 exp       exp      100    13.0800000     7.3425287     1.0000000    26.0000000
 educ      educ     100    16.0200000     2.3049354    12.0000000    20.0000000
 gender    gender   100     0.6600000     0.4760952             0     1.0000000
 sup       sup      100   340.1000000   167.1779733    60.0000000   600.0000000
 cass      cass     100   175.1000000    15.4066102   150.0000000   200.0000000
 board     board    100     0.4900000     0.5024184             0     1.0000000
 age       age      100    42.8400000     9.0729034    23.0000000    64.0000000
 profits   profits  100     7.7000000     1.5537508     5.0000000    10.0000000
 int       int      100     0.1800000     0.3861229             0     1.0000000
 sales     sales    100    24.8300000     2.7415803    20.0000000    30.0000000
 ------------------------------------------------------------------------------
```

As you see, there are 100 rows of data, which would be a lot for someone else to look at. The names (you can check) match up with what I said the variables were.

(b) Run a regression predicting log-salary from everything else, except row number. Show the text output (here and below).

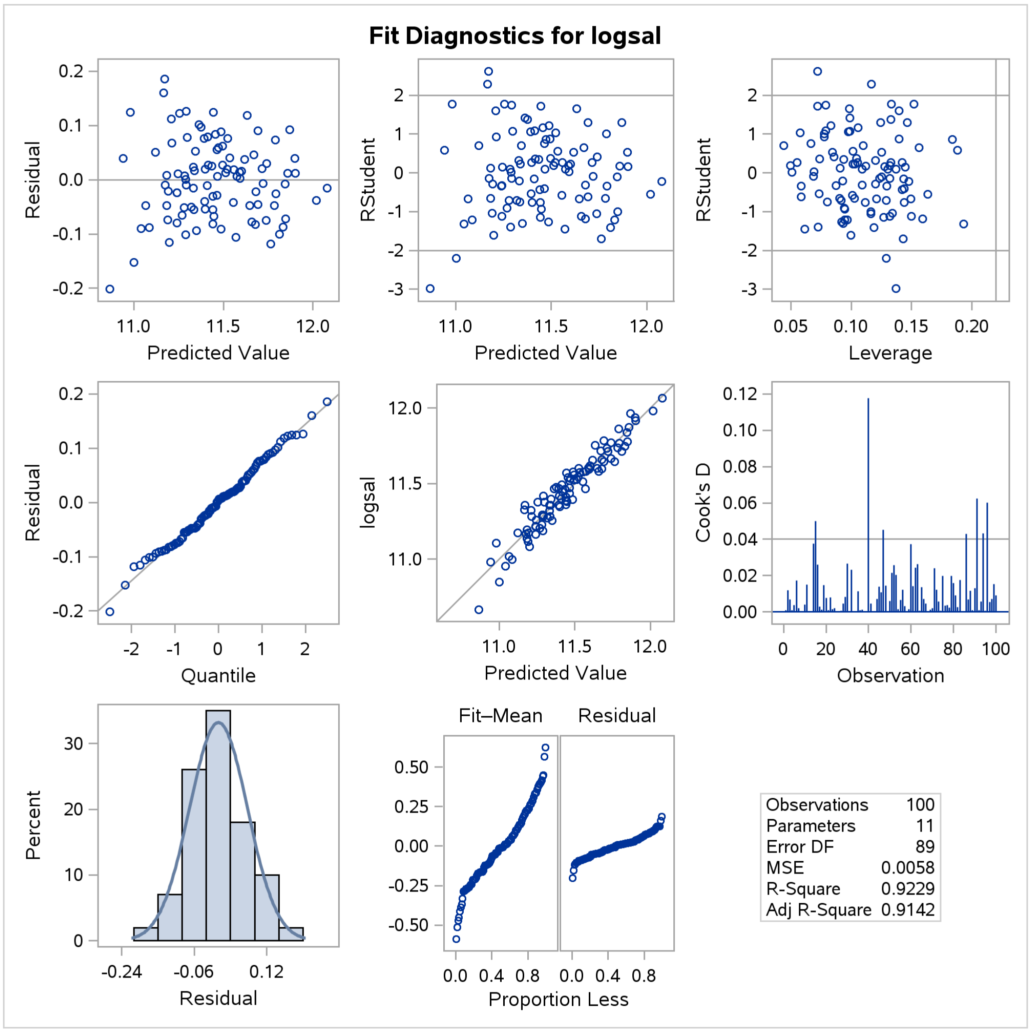
Solution:

```
proc reg;
  model logsal=exp educ gender sup cass board age profits int sales;
```

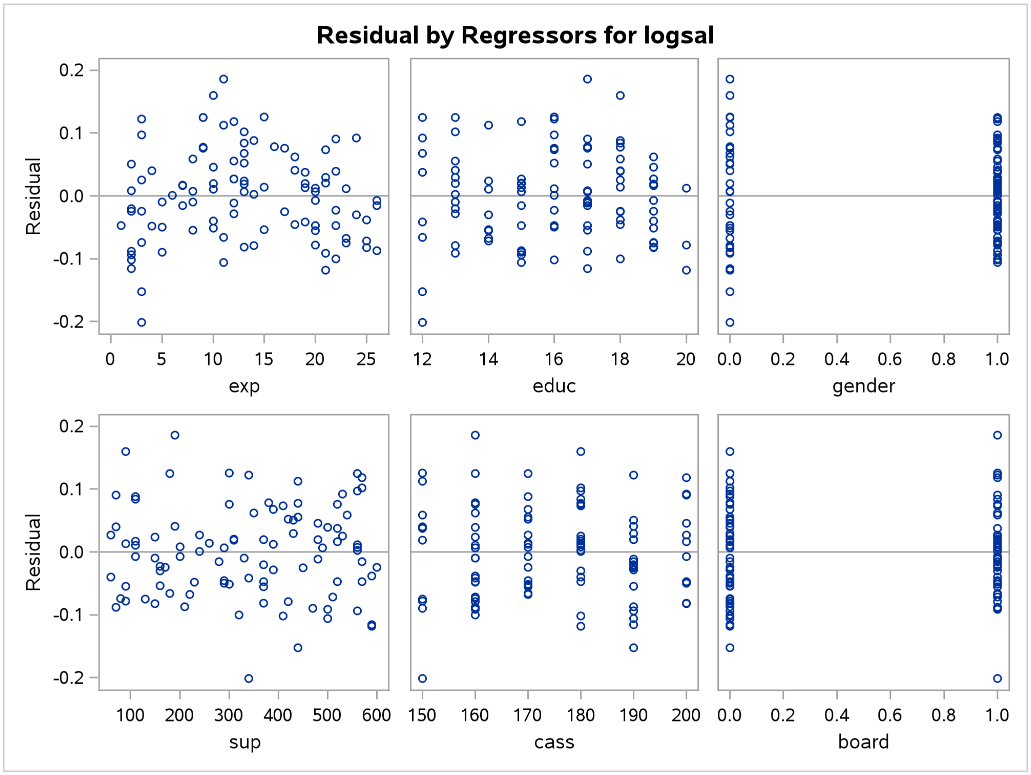
with output

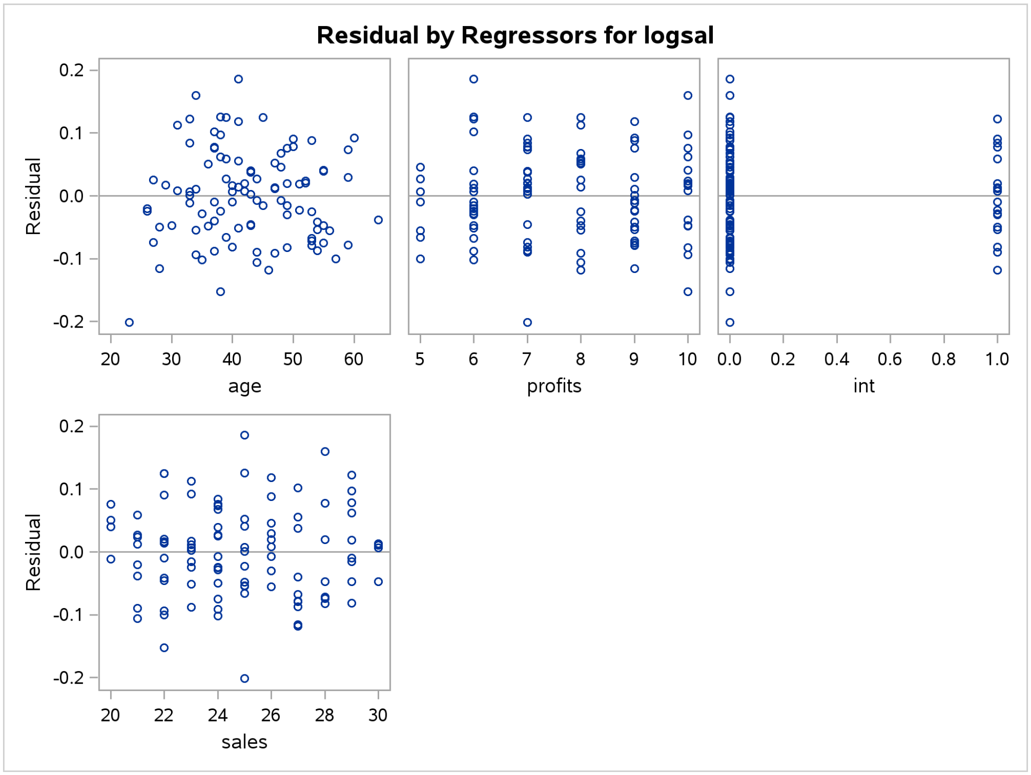
```
                              The REG Procedure
                                Model: MODEL1
                       Dependent Variable: logsal logsal

                    Number of Observations Read         100
                    Number of Observations Used         100
                              Analysis of Variance

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

Model                    10          6.16747        0.61675     106.54    <.0001
Error                    89          0.51520        0.00579
Corrected Total          99          6.68267
             Root MSE                0.07608    R-Square     0.9229
             Dependent Mean         11.45502    Adj R-Sq     0.9142
             Coeff Var               0.66420
                              Parameter Estimates

                                  Parameter      Standard
 Variable    Label        DF       Estimate         Error   t Value   Pr > |t|

 Intercept   Intercept     1       10.02192       0.14805     67.69     <.0001
 exp         exp           1        0.02792       0.00177     15.75     <.0001
 educ        educ          1        0.02903       0.00343      8.48     <.0001
 gender      gender        1        0.22434       0.01708     13.13     <.0001
 sup         sup           1     0.00051397    0.00004922     10.44     <.0001
 cass        cass          1        0.00205    0.00052499      3.90     0.0002
 board       board         1       -0.01538       0.01686     -0.91     0.3641
 age         age           1    -0.00050971       0.00144     -0.35     0.7238
 profits     profits       1       -0.00263       0.00513     -0.51     0.6089
 int         int           1       -0.02656       0.02037     -1.30     0.1956
 sales       sales         1    -0.00097742       0.00296     -0.33     0.7420
```

and the graphs



Observations	100
Parameters	11
Error DF	89
MSE	0.0058
R-Square	0.9229
Adj R-Square	0.9142


Residual by Regressors for logsal
Residual
exp
educ
gender
sup
cass
board


Residual by Regressors for logsal
Residual
age
profits
int
sales

I didn't ask you to look at the plots, because I wanted you to do the variable-elimination (coming up). Normally, you would check that things are at least approximately OK, here and at the end. So I'll do it here, starting with the array of nine graphs of which I look at the usual two:

- residuals vs. fitted values, top left: a tiny bit of evidence of fanning-in, since the four residuals farthest from zero are all on the left. I'd really want more evidence of fanning-in than this, though.
- normal quantile plot of residuals: as straight as you could wish for.
- There are a lot of explanatory variables, and we get a plot of residuals against each one. These look pretty random and trend-free, so I don't think we need to be concerned. Note that some of the variables take only a few possible values (they are rather discrete), so you get stacks of points one above another, eg. for profits. Some of the explanatory variables are either 0 or 1 (these are "indicators" for categorical variables with two categories). For these, you want both categories to have average residual around zero with equal spread. For `gender` 1, the males, the residuals appear less spread out. I think we will have to live with that. (Another way would be to do the regression twice, for males and females separately.)

(c) Which explanatory variable is least significant? Run a regression without it.

Solution: This question is going to involve a great deal of copying and pasting. `sales` comes out first:

```
proc reg;
  model logsal=exp educ gender sup cass board age profits int;
```

with output

```
                              The REG Procedure
                                Model: MODEL1
                         Dependent Variable: logsal logsal

                      Number of Observations Read         100
                      Number of Observations Used         100
                              Analysis of Variance

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

Model                        9         6.16683        0.68520     119.55    <.0001
Error                       90         0.51583        0.00573
Corrected Total             99         6.68267
             Root MSE                  0.07571    R-Square     0.9228
             Dependent Mean           11.45502    Adj R-Sq     0.9151
             Coeff Var                 0.66090
```

```
                              Parameter Estimates

                                    Parameter      Standard
 Variable    Label       DF          Estimate         Error    t Value    Pr > |t|

 Intercept   Intercept    1           9.99498       0.12293      81.30      <.0001
 exp         exp          1           0.02776       0.00170      16.33      <.0001
 educ        educ         1           0.02904       0.00341       8.52      <.0001
 gender      gender       1           0.22525       0.01677      13.43      <.0001
 sup         sup          1        0.00051529    0.00004881      10.56      <.0001
 cass        cass         1           0.00204    0.00052207       3.91      0.0002
 board       board        1          -0.01472       0.01666      -0.88      0.3791
 age         age          1       -0.00041412       0.00140      -0.30      0.7683
 profits     profits      1          -0.00260       0.00510      -0.51      0.6118
 int         int          1          -0.02666       0.02027      -1.32      0.1916
```

(d) Continue removing the least significant variable until you need to stop, and explain briefly why you stopped.

Solution: You might guess that `board`, `age`, `profits` and `int` will need to come out, but take them one at a time, `age` first:

```
proc reg;
  model logsal=exp educ gender sup cass board profits int;
```

giving

```
                             The REG Procedure
                               Model: MODEL1
                     Dependent Variable: logsal logsal

                   Number of Observations Read         100
                   Number of Observations Used         100
                             Analysis of Variance

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

Model                     8         6.16633        0.77079     135.85    <.0001
Error                    91         0.51633        0.00567
Corrected Total          99         6.68267
             Root MSE               0.07533    R-Square     0.9227
             Dependent Mean        11.45502    Adj R-Sq     0.9159
             Coeff Var              0.65758
                              Parameter Estimates

                                    Parameter      Standard
 Variable    Label       DF          Estimate         Error    t Value    Pr > |t|

 Intercept   Intercept    1           9.97787       0.10791      92.47      <.0001
 exp         exp          1           0.02737       0.00103      26.47      <.0001
 educ        educ         1           0.02915       0.00337       8.65      <.0001
 gender      gender       1           0.22451       0.01650      13.61      <.0001
 sup         sup          1        0.00051473    0.00004853      10.61      <.0001
 cass        cass         1           0.00206    0.00051702       3.98      0.0001
 board       board        1          -0.01336       0.01593      -0.84      0.4037
 profits     profits      1          -0.00259       0.00508      -0.51      0.6117
 int         int          1          -0.02617       0.02010      -1.30      0.1961
```

Then profits:

```
proc reg;
  model logsal=exp educ gender sup cass board int;
```

giving

```
                                The REG Procedure
                                  Model: MODEL1
                        Dependent Variable: logsal logsal

                     Number of Observations Read         100
                     Number of Observations Used         100
                               Analysis of Variance

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

Model                     7            6.16486        0.88069     156.48    <.0001
Error                    92            0.51781        0.00563
Corrected Total          99            6.68267
             Root MSE              0.07502    R-Square     0.9225
             Dependent Mean       11.45502    Adj R-Sq     0.9166
             Coeff Var             0.65493
                               Parameter Estimates

                                     Parameter      Standard
 Variable    Label       DF         Estimate         Error    t Value    Pr > |t|

 Intercept   Intercept    1          9.96623       0.10503      94.88      <.0001
 exp         exp          1          0.02736       0.00103      26.58      <.0001
 educ        educ         1          0.02908       0.00335       8.67      <.0001
 gender      gender       1          0.22430       0.01643      13.65      <.0001
 sup         sup          1       0.00051576    0.00004829      10.68      <.0001
 cass        cass         1          0.00201    0.00050694       3.97      0.0001
 board       board        1         -0.01202       0.01564      -0.77      0.4444
 int         int          1         -0.02491       0.01986      -1.25      0.2130
```

Then board:

```
proc reg;
  model logsal=exp educ gender sup cass int;
```

```
                         The REG Procedure
                           Model: MODEL1
                 Dependent Variable: logsal logsal

              Number of Observations Read         100
              Number of Observations Used         100
                        Analysis of Variance

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

Model                    6       6.16154        1.02692     183.26    <.0001
Error                   93       0.52113        0.00560
Corrected Total         99       6.68267
             Root MSE              0.07486    R-Square     0.9220
             Dependent Mean       11.45502    Adj R-Sq     0.9170
             Coeff Var             0.65348
                        Parameter Estimates

                                 Parameter      Standard
 Variable    Label       DF       Estimate         Error   t Value   Pr > |t|

 Intercept   Intercept    1        9.94602       0.10146     98.03     <.0001
 exp         exp          1        0.02733       0.00103     26.62     <.0001
 educ        educ         1        0.02933       0.00333      8.81     <.0001
 gender      gender       1        0.22322       0.01633     13.67     <.0001
 sup         sup          1     0.00052305    0.00004724     11.07     <.0001
 cass        cass         1        0.00206    0.00050144      4.11     <.0001
 int         int          1       -0.02549       0.01980     -1.29     0.2011
```

Finally (we hope) `int`:

```
proc reg;
  model logsal=exp educ gender sup cass;
```

```
                           The REG Procedure
                             Model: MODEL1
                    Dependent Variable: logsal logsal

                 Number of Observations Read         100
                 Number of Observations Used         100
                          Analysis of Variance

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

Model                     5        6.15225        1.23045     218.06    <.0001
Error                    94        0.53041        0.00564
Corrected Total          99        6.68267
             Root MSE              0.07512    R-Square     0.9206
             Dependent Mean       11.45502    Adj R-Sq     0.9164
             Coeff Var             0.65576
                          Parameter Estimates

                                  Parameter      Standard
 Variable    Label       DF       Estimate          Error   t Value   Pr > |t|

 Intercept   Intercept    1        9.96193        0.10106     98.58     <.0001
 exp         exp          1        0.02728        0.00103     26.50     <.0001
 educ        educ         1        0.02909        0.00334      8.72     <.0001
 gender      gender       1        0.22469        0.01635     13.74     <.0001
 sup         sup          1     0.00052442     0.00004740     11.06     <.0001
 cass        cass         1        0.00196     0.00049718      3.95     0.0002
```

Yep, that's the end. Everything else is strongly significant and has to stay in the model.

Also note that R-squared began and also ended around 92%: taking out those variables has had only a tiny effect on the fit of the model.

(e) Which explanatory variables are in your final model? Name them in full. That is, don't just list the names of the variables.

Solution: These ones:

- Years of experience
- Years of education
- Gender
- Number of employees supervised
- Corporate assets

(f) Look at each of your slope coefficients. Are they positive or negative? Does that make sense in the context of this problem?

Solution: Mine are all positive. That is, someone who has more years of experience, more education, supervises more employees or works in a company with more corporate assets would be expected to receive a higher salary. We'd expect all of these variables to have this kind of effect.

The one I didn't talk about was `gender`. This is also positive. Since males were 1 and females 0, according to the question, this means that males are expected to make more than females, *all else being equal*. This may not make you happy, but it's what the data are saying. (And note the strength of the conclusion: it's *after adjusting for any other differences between males and females*.)

The right thing to do next is to look at residual plots for your final model. The *right* thing to do is to split your data into a "training set" with which you build your model, and a separate "test set" on which you see how well it works. But that's farther than we go now.

SAS also contains a procedure called `glmselect`, which automates this process. Here's how it looks for this dataset:

```
proc glmselect;
  model logsal=exp educ gender sup cass board age profits int sales
    / selection=backward;
```

with output

```
                      The GLMSELECT Procedure

             Data Set                      WORK.SALARIES
             Dependent Variable                   logsal
             Selection Method                   Backward
             Select Criterion                        SBC
             Stop Criterion                          SBC
             Effect Hierarchy Enforced              None
              Number of Observations Read         100
              Number of Observations Used         100
                             Dimensions

                     Number of Effects       11
                     Number of Parameters    11
                      The GLMSELECT Procedure

                     Backward Selection Summary

                    Effect        Number
            Step    Removed    Effects In           SBC

               0                       11     -476.1799
           ----------------------------------------------
               1    sales              10     -480.6625
               2    age                 9     -485.1707
               3    profits             8     -489.4911
               4    board               7     -493.4571
               5    int                 6     -496.2957*

                     * Optimal Value of Criterion
```

```
         Selection stopped at a local minimum of the SBC criterion.
                               Stop Details

            Candidate               Candidate         Compare
            For          Effect           SBC         SBC

            Removal      cass       -485.5667    >    -496.2957
                         The GLMSELECT Procedure
                              Selected Model

         The selected model is the model at the last step (Step 5).


                Effects: Intercept exp educ gender sup cass
                            Analysis of Variance

                                          Sum of           Mean
     Source                 DF          Squares          Square    F Value

     Model                   5          6.15225         1.23045     218.06
     Error                  94          0.53041         0.00564
     Corrected Total        99          6.68267
                       Root MSE                0.07512
                       Dependent Mean         11.45502
                       R-Square                 0.9206
                       Adj R-Sq                 0.9164
                       AIC                  -409.92676
                       AICC                 -408.70937
                       SBC                  -496.29574
                            Parameter Estimates

                                                   Standard
         Parameter    DF        Estimate              Error    t Value

         Intercept     1        9.961935           0.101057      98.58
         exp           1        0.027276           0.001029      26.50
         educ          1        0.029092           0.003337       8.72
         gender        1        0.224693           0.016350      13.74
         sup           1        0.000524        0.000047398      11.06
         cass          1        0.001962           0.000497       3.95
```

You can read through the output to see which variables were removed at each step, and which ones were left at the end: the same five as we found, since the procedure is supposed to be identical.

`proc glmselect` can also produce plots. See the baseball example at `https://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_glmselect_sect030.htm` for illustrations.

Yet another way to go is to remove a bunch of xs from the first regression (the nonsignificant ones) and test whether that was a good idea. That goes in the `proc reg` like this:

```
proc reg;
  model logsal=exp educ gender sup cass board age profits int sales;
  test board, age, profits, int;
```

The `test` line is testing the null hypothesis that all four of these variables (the nonsignificant ones in the original regression) have slope 0: that is, that they contribute nothing to the original regression. You get all the regression output again plus this:

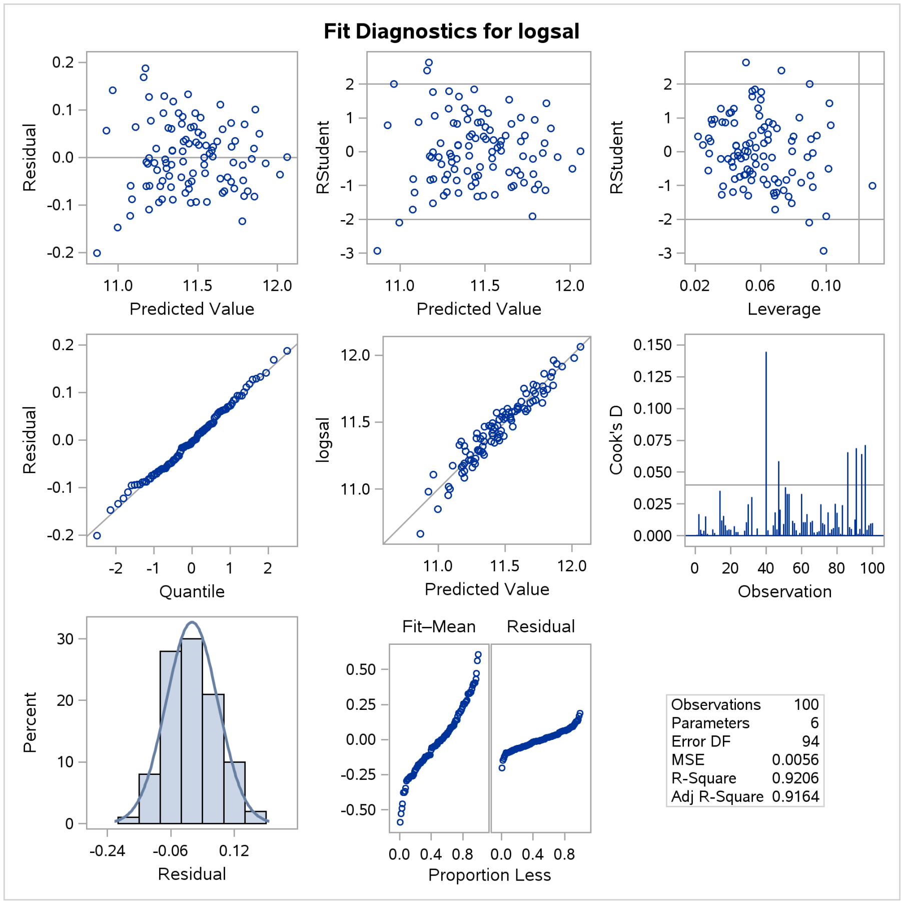
```
                          The REG Procedure
                           Model: MODEL1

               Test 1 Results for Dependent Variable logsal

                                         Mean
          Source             DF        Square    F Value    Pr > F

          Numerator           4       0.00371       0.64    0.6343
          Denominator        89       0.00579
```

This is like `anova` in R: it's testing that the small model (the one without those four xs) fits equally well compared to the big one (with everything in it). Since this null hypothesis is not rejected, we would prefer the smaller model because it is simpler. (If the null had been rejected here, we would have preferred the big model because it would fit significantly better.)

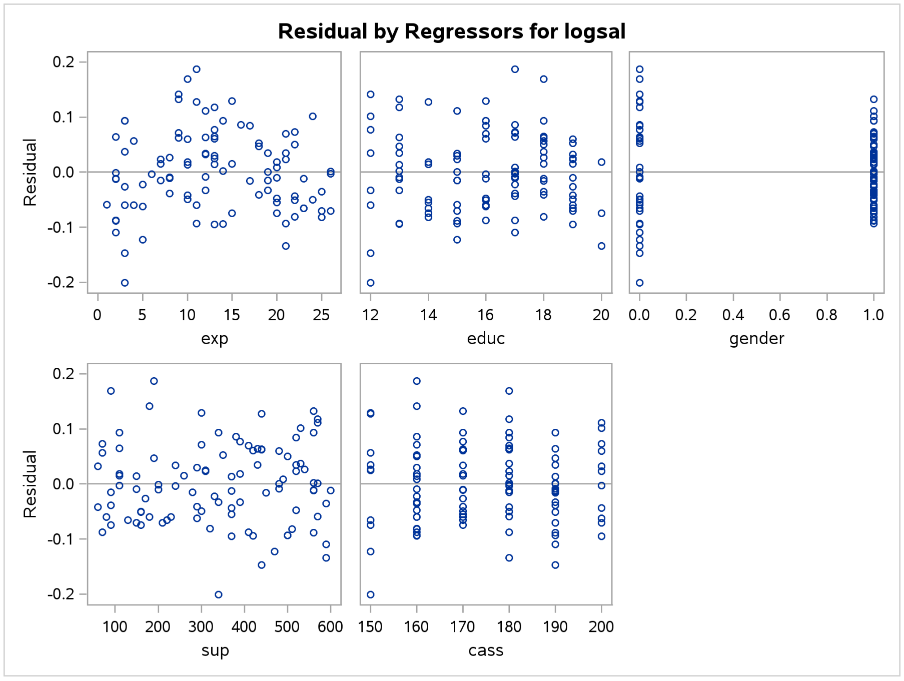
Remember that the t-tests in regression strictly only refer to one x-variable at a time, so if you take out more than one, you need to test at the end that what you did was OK. (What could have happened was that taking one of these xs out made one of the others significant, which can happen if the xs are correlated with each other. In this case, though, we were all good.)

`test` in SAS will test any "linear hypothesis" about slopes, so that for example you can also test that a slope is 3, or that two of the slopes add up to 6, or combinations of things like this. There is some theory that says how to do this using matrices and an F-test, which you probably see in C67. If you do, the formulas at `https://support.sas.com/documentation/cdl/en/statug/63962/HTML/default/viewer.htm#statug_reg_sect022.htm` will look familiar.

We should probably look at our residual plots (from our last regression) just to make sure that all is OK:



```
Observations      100
Parameters          6
Error DF           94
MSE            0.0056
R-Square       0.9206
Adj R-Square   0.9164
```



The few issues we have are the same as before, which we decided to live with.

11.2. The United States is divided into a large number of counties: areas larger than a city but much smaller than a state. This question will work with a data set of the 440 largest counties, which can be found in `http://www.utsc.utoronto.ca/~butler/c32/smsa.txt`.

The variables in the data set are:

- an ID number of the county
- the name of the county (text)
- the state in which the county is located (text)
- land area of the county (square miles)
- total population
- Percent of population aged 18–34
- Percent of population aged 65 or older
- Number of active physicians
- Number of hospital beds
- Total number of serious crimes
- Percent high school graduates (percent of all adults aged 25 or older that completed grade 12)
- Percent of population with bachelor's degrees (B. Sc. or BA)
- Percent of population below poverty level

- Percent of labour force that is unemployed (labour force includes those who could be employed, and excludes university/college students, those serving in military, those who cannot work for health reasons).
- Per capita (mean) income of entire population
- Total personal income of entire population (millions of dollars)
- Region of the US (1=northeast, 2=north central, 3=south, 4=west)

Our aim in this question is to understand the factors affecting the number of active physicians (family doctors) in a county.

(a) Read the data into SAS. Display the values for yourself, but not to hand in (if you were handing this in).

Solution:

This kind of thing:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/smsa.txt";
proc import
  datafile=myurl
  dbms=dlm
  out=county
  replace;
  delimiter=' ';
  getnames=yes;
```

You ought to run this with `proc print` on the end, until you are happy that you have it right, *but if you were to hand in 440 lines of* `proc print` *output, you would deserve to lose as many marks as the grader decides to deduct.* Or more.

All the variables that are percentages had names starting with `pct`. This makes it easier to find them below.

(b) List the first 10 observations of your data set, and check that the columns that should be percentages actually look like percentages. Hint: to display a certain number of rows, *specify a data set name with* `data=` and put `obs=` and a number *in brackets* on the end of the line.

Solution: The hint suggests this (I was trying not to give it away completely). You have to specify a name for your data set; it doesn't work otherwise:

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

```
Obs          id    name            state          area           pop

  1           1    Los_Angeles      CA            4060       8863164
  2           2    Cook             IL             946       5105067
  3           3    Harris           TX            1729       2818199
  4           4    San_Diego        CA            4205       2498016
  5           5    Orange           CA             790       2410556
  6           6    Kings            NY              71       2300664
  7           7    Maricopa         AZ            9204       2122101
  8           8    Wayne            MI             614       2111687
  9           9    Dade             FL            1945       1937094
 10          10    Dallas           TX             880       1852810

Obs     pct1834         pct65    physicians          beds        crimes

  1        32.1           9.7         23677         27700        688936
  2        29.2          12.4         15153         21550        436936
  3        31.3           7.1          7553         12449        253526
  4        33.5          10.9          5905          6179        173821
  5        32.6           9.2          6062          6369        144524
  6        28.3          12.4          4861          8942        680966
  7        29.2          12.5          4320          6104        177593
  8        27.4          12.5          3823          9490        193978
  9        27.1          13.9          6274          8840        244725
 10        32.6           8.2          4718          6934        214258

Obs   pcthighsch   pctbachelor     pctpverty      pctunemp       meaninc

  1           70          22.3          11.6             8         20786
  2         73.4          22.8          11.1           7.2         21729
  3         74.9          25.4          12.5           5.7         19517
  4         81.9          25.3           8.1           6.1         19588
  5         81.2          27.8           5.2           4.8         24400
  6         63.7          16.6          19.5           9.5         16803
  7         81.5          22.1           8.8           4.9         18042
  8           70          13.7          16.9            10         17461
  9           65          18.8          14.2           8.7         17823
 10         77.1          26.3          10.4           6.1         21001

Obs     totalinc        region

  1       184230             4
  2       110928             2
  3        55003             3
  4        48931             4
  5        58818             4
  6        38658             1
  7        38287             4
  8        36872             2
  9        34525             3
 10        38911             3
```

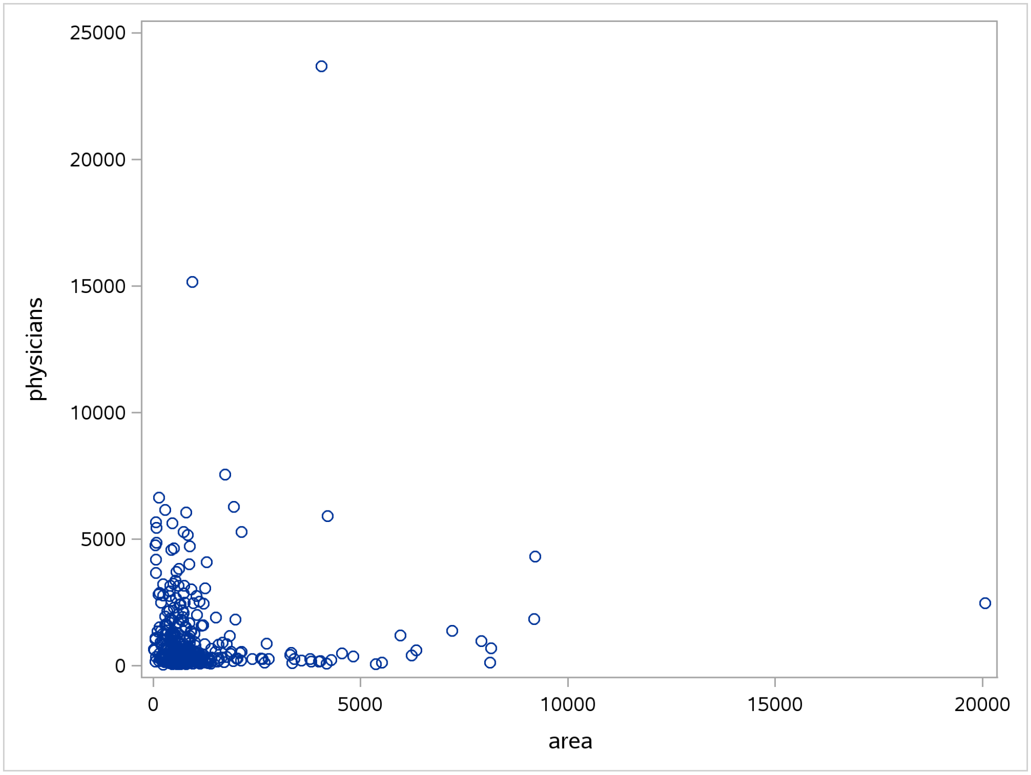
This displays all the many variables for the first 10 observations. Now, because I named the "percent" variables beginning with `pct`, I can easily check that these percentages of people: aged 18–34, aged over 65, completing high school, with a bachelor's degree, in poverty, and unemployed look like percentages, and these are the only ones that do. (You should be checking six variables altogether.)

I thought you could do this by using a `where` line with `_N_` in it to pick out the rows you want, but it doesn't work. You can use `_N_` when creating a new data set with `data` and `set`, but not otherwise. That seems like way too much trouble here.

(c) We are going to predict the number of physicians in a county from some of the other variables. Start by obtaining a scatterplot of the number of physicians against the land area. What do you see, and what potential problems might this cause for a regression?

Solution:

```
proc sgplot;
  scatter x=area y=physicians;
```



Almost all the data points are at the bottom left of the picture, with only a few elsewhere. There are a few counties with very big land area (one especially big), and a few counties with a lot of physicians (not always the ones with large land area). As a result, the relationship is not at all clear.

One of the problems with regression is “influential points”, observations that are very different from the others. We seem to have a few of them here. The problem with influential points is that they can (as their name implies) influence where the regression goes, even though there are only a few of them.

This is more discussion than you need, but I want you to observe two things: (i) that the majority of the points are bottom left (or that only a few are elsewhere), to answer “what do you see”, and (ii) the points far away from the others can have a big influence over where the regression line goes, to answer “potential problems”.

I guess this plot also shows a non-linear relationship, but that’s not the best answer because the evidence for non-linearity is in those (relatively few) points off by themselves, not in the big mass of points bottom left, for which it’s very unclear what kind of trend there is.

(d) One way to solve the problems unearthed in the previous part is to transform the variables that can be very large. Create a new data set with log-transformed number of physicians, land area and population.

Solution: This is `data` and `set`:

```
data county2;
  set county;
  logphys=log(physicians);
  logpop=log(pop);
  logarea=log(area);
```

If you like, print out the first few lines to check that the new values look sensible. Or you can summarize, eg. like this:

```
proc means;
  var physicians logphys pop logpop area logarea;
```

```
                              The MEANS Procedure

 Variable       N           Mean        Std Dev        Minimum        Maximum
 ----------------------------------------------------------------------------
 physicians   440    987.9977273        1789.75     39.0000000       23677.00
 logphys      440      6.1517531      1.1440522      3.6635616     10.0722594
 pop          440      393010.92      601987.02      100043.00     8863164.00
 logpop       440     12.4759757      0.7903838     11.5133554     15.9974144
 area         440        1041.41        1549.92     15.0000000       20062.00
 logarea      440      6.5174458      0.8717066      2.7080502      9.9065828
 ----------------------------------------------------------------------------
```

The minimum and maximum of the logged variables should be the (natural) logs of the original values:

```
log(39)

## [1] 3.663562

log(23677)

## [1] 10.07226

log(100043)

## [1] 11.51336

log(8863164)

## [1] 15.99741

log(15)

## [1] 2.70805

log(20062)

## [1] 9.906583
```

That appears to check out.

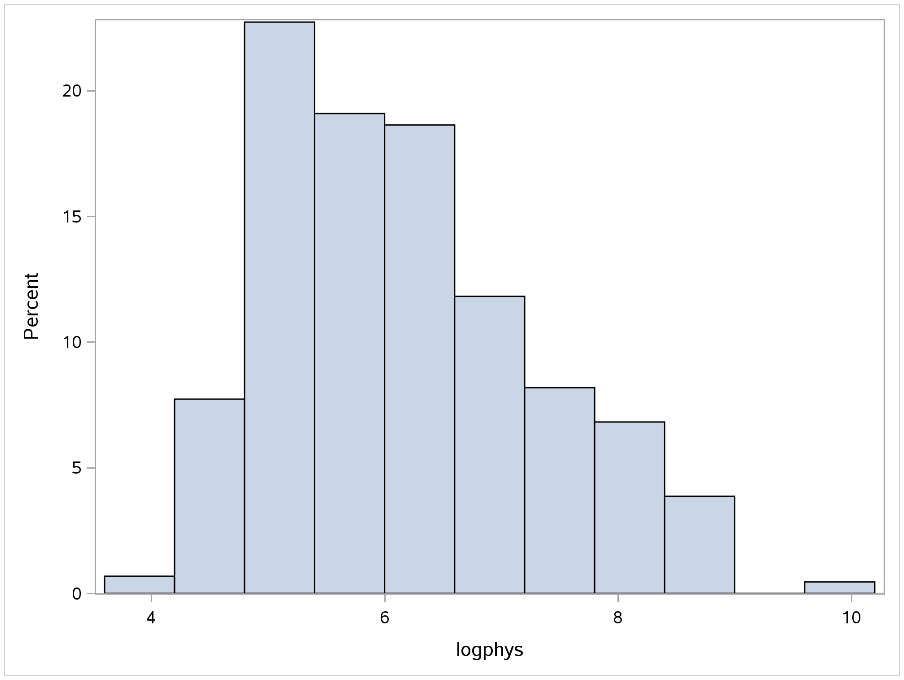
Note that taking logs has made the very big values not so very big. There is a county with over 8 million people in it! But the log of that is only about 16.

The log of the mean (population, say) is not the same as the mean of the log-population. You might like to think about why not.

(e) Draw histograms of your three new variables. Do they have something like normal distributions?

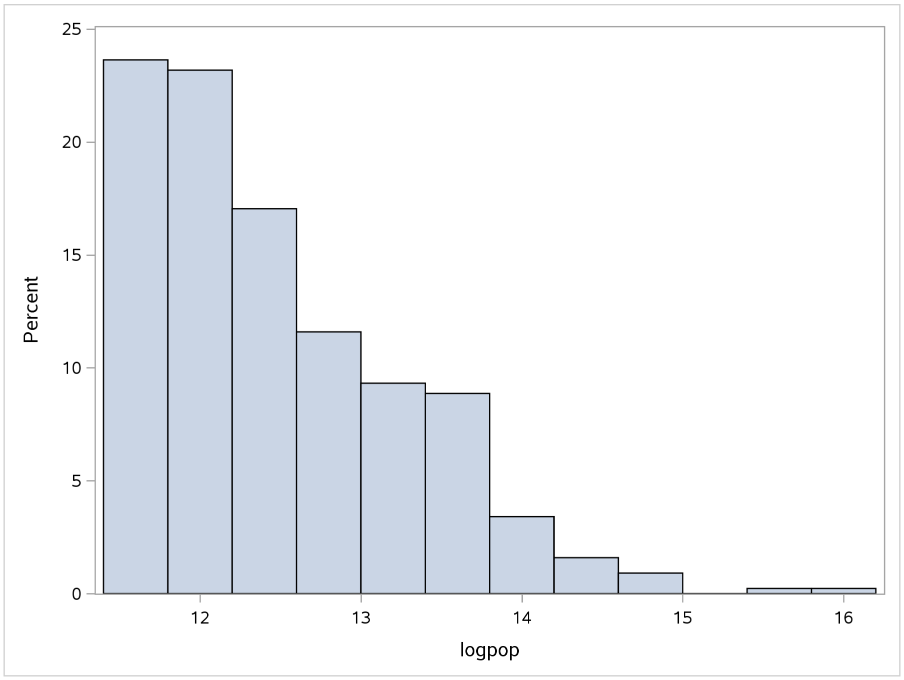
Solution: The obvious thing is to draw the histograms one at a time, copying and pasting your code. Log-physicians:

```
proc sgplot;
  histogram logphys;
```


Percent
20
15
10
5
0
4
6
8
10
logphys

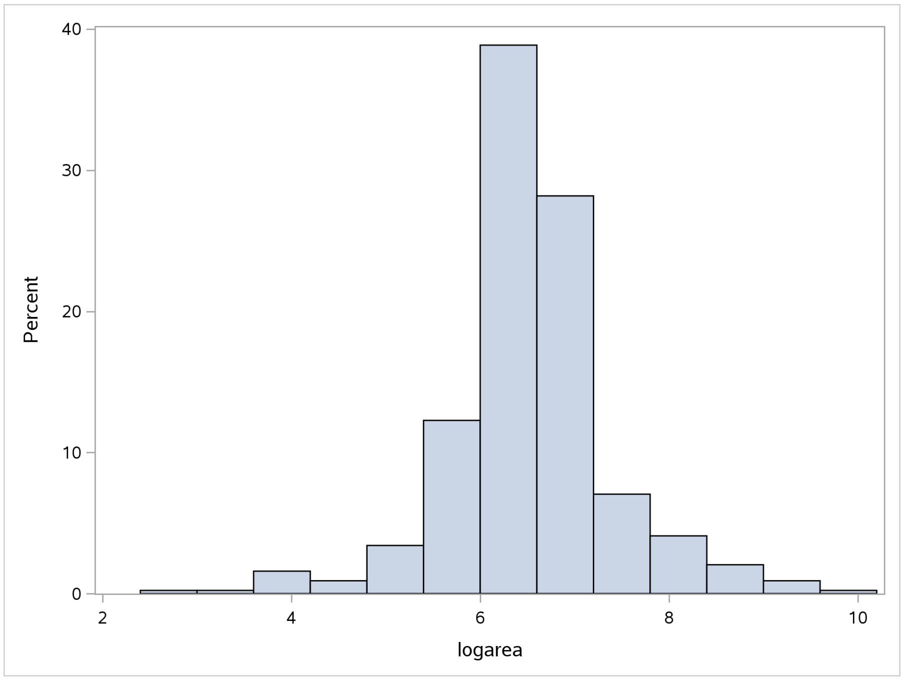
Log-population:

```
proc sgplot;
  histogram logpop;
```



Log-area:

```
proc sgplot;
  histogram logarea;
```



Log-area is nice and symmetric. Log-population is still a bit skewed, and log-physicians is a bit skewed with an outlier. But if you compare the histograms of the original variables, things are a lot better than they were.

I wanted to say something about normal distributions and regression at this point, since that often seems misunderstood. What you actually *need* is for the "errors" to be normally distributed, and since you never actually observe the errors themselves, you look at the residuals, and if they are approximately normal, with no patterns in relation to anything else, you are good. There is *no* need for the y values or the x values to be normally distributed; in fact, the theory of regression says only that the x's are *given* (not random at all), or, if you prefer, you work *conditional* on the x's you observed.

A little bit of the theory, for those who care: you assume that the model (one x) is $y_i = \alpha + \beta x_i + e_i$, where the errors e_i are the only random thing, and they have independent normal distributions with mean 0 and variance σ^2. The x_i are fixed, and the intercept α and slope β are constant parameters to be estimated (which is done by maximum likelihood or least squares). Another way to look at this, because of properties of the normal distribution, is that the y_i have independent normal distributions with mean $\alpha + \beta x_i$ and constant variance σ^2. I actually like this way better, because it transfers over to generalized linear models, which you might see later. Generalized linear models don't have "errors" in the same sense; they have a distribution for the response, with a mean that depends on the x and a variance that might depend on the distribution. For example, logistic regression says that y_i has a binomial distribution with a success probability p_i that depends on the x_i. In a typical application, x might be the dose of some poison and y might be whether an animal lives or dies. In a binomial distribution, the mean and the variance both depend on p, so once you know the mean, you also know the variance.

As I said, you never actually observe the e_i; the best you can do is *estimate* them, using the residuals. The independence of the errors plays out in the need for randomness in any graphs involving residuals; the normality of the errors plays out in wanting the normal quantile plot of the residuals to be straight, and the constant variance plays out in wanting no fanning-out.

Having said all of that, if the distribution of your x's has outliers, so (probably) will the distribution of your y's, and then you will be dealing with influential points. It is not *necessary* for the distribution of your x's to be even approximately normal, but it generally makes your life easier if it is.

So that's why I had you do the transformations and look at the histograms afterwards.

(f) Do a regression predicting the log-number of physicians from the log-population and log-area. Display and comment on the results (the printed part, not the graphs, yet).

Solution: Nothing terribly surprising in the code. I forgot that you separate explanatory variables in SAS by a space, not a plus, so I had to do it twice:

```
proc reg;
  model logphys=logpop logarea;
```

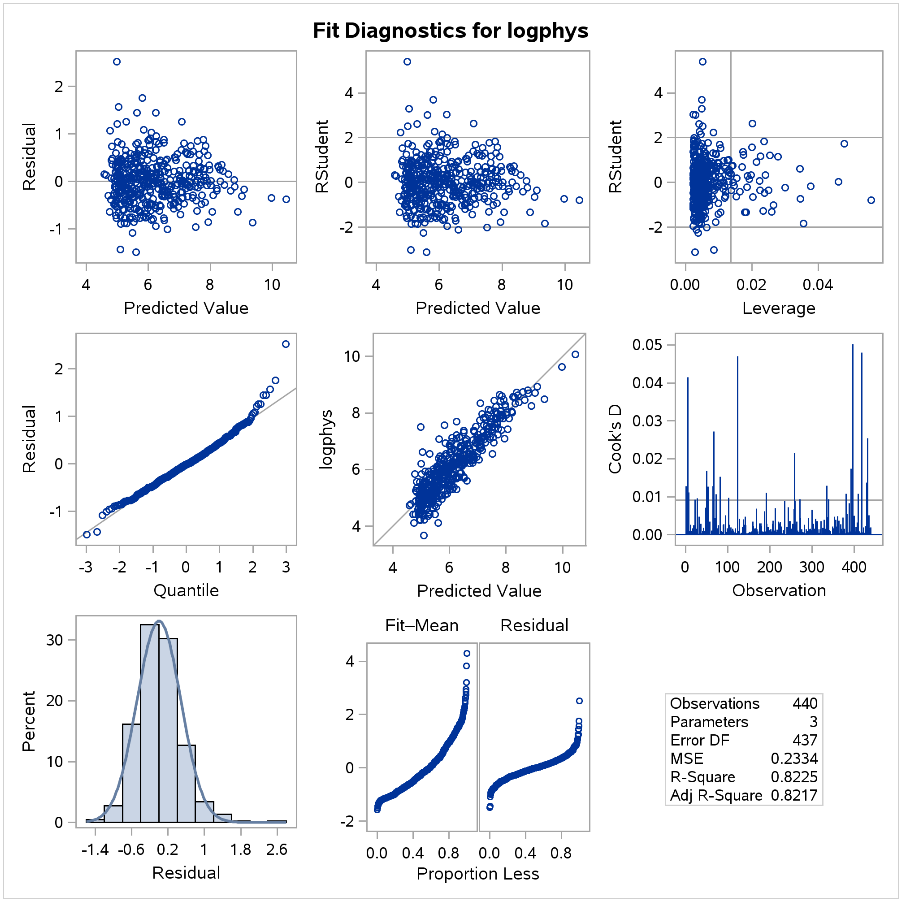
```
                             The REG Procedure
                               Model: MODEL1
                        Dependent Variable: logphys

                 Number of Observations Read         440
                 Number of Observations Used         440
```

```
                           Analysis of Variance

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

Model                     2      472.58876      236.29438    1012.37    <.0001
Error                   437      101.99878        0.23341
Corrected Total         439      574.58754
             Root MSE              0.48312    R-Square     0.8225
             Dependent Mean        6.15175    Adj R-Sq     0.8217
             Coeff Var             7.85340
                          Parameter Estimates

                        Parameter       Standard
     Variable     DF     Estimate          Error    t Value    Pr > |t|

     Intercept     1     -9.04884        0.39932     -22.66      <.0001
     logpop        1      1.30489        0.02918      44.71      <.0001
     logarea       1     -0.16557        0.02646      -6.26      <.0001
```

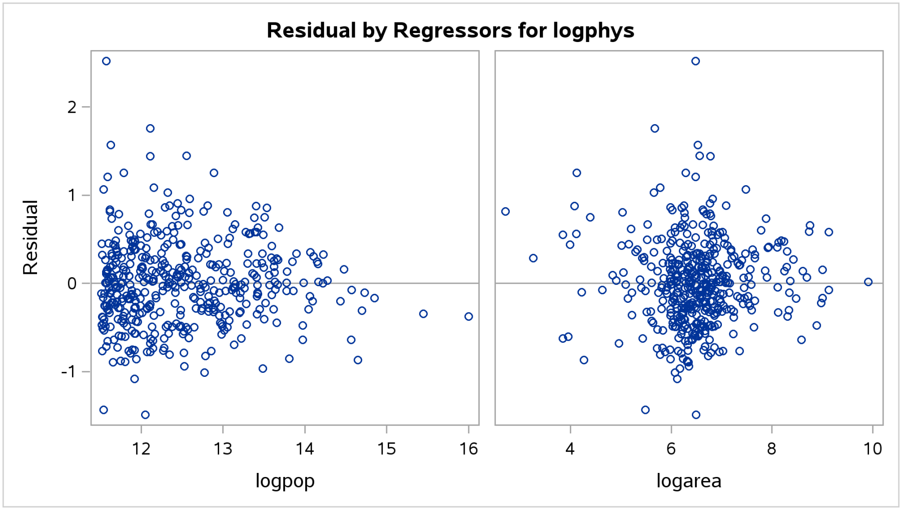
A nice high R-squared (for this kind of thing) of 82.25%. Both explanatory variables are strongly significant. Log-population has a positive slope and log-area has a negative one. That means that counties with a higher population have more physicians (no surprise there!). Counties with a larger area have fewer physicians, even after accounting for population. That is to say, you can't just say that larger counties are likely to be more sparsely populated and *that's* the reason they have fewer doctors. I think you have to say something along the lines of cities having to be big enough to support having a physician, and a county with large area might have a decent-sized population but not very many cities of any size and therefore not many places where it is profitable for a doctor to be. Something like that.

(g) Check the residual plots for the regression you just did. Do you see anything unacceptable?

Solution: Here is that array of graphs:



Observations	440
Parameters	3
Error DF	437
MSE	0.2334
R-Square	0.8225
Adj R-Square	0.8217



Residuals vs. fitted values top left looks pretty much like a random cloud (a couple of outliers). You might see some fanning in, but on the other hand this might be driven by the few counties that happen to have a small predicted value and a large-in-size residual. When you have a lot of data, it's important to beware of problems that are really only caused by a few points. The normal quantile plot looks pretty straight, with maybe a few outliers at the top; residuals against log-area (on the right) a nice random scatter; residuals vs. log-population has some fanning in. This might be because even the distribution of log-population was still skewed and we ought to have gone further in our transformation (something like reciprocal, maybe, instead of log).

I might think about some other way of transforming population, but overall I think this is not too bad.

(h) The regression you just did predicts log-physicians from log-population and log-area. Do a little algebra to get a relationship predicting the actual number of physicians from (functions of) the other variables. Simplify your result as far as you can.

Solution: I can't remember whether I promised "no math" or "very little math" at the start of the course, but anyway. Let's define some symbols to make our lives easier: let d be the number of physicians ("d" for "doctor"), p be the population and a the area of a county. Then our regression says (rounding things off a bit):

$$\log d = -9.05 + 1.30 \log p - 0.17 \log a$$

Take e-to-the-power-of both sides, which I'll write exp:

$$d = \exp(-9.05 + 1.30 \log p - 0.17 \log a)$$

Adding things inside exp means multiplying the separate exp's:

$$d = \exp(-9.05) \exp(1.30 \log p) \exp(-0.17 \log a)$$

A piece of math: $\exp(a \log x) = \{\exp(\log x)\}^a = x^a$:

$$d = e^{-9.05} p^{1.30} a^{-0.17}$$

and you can work out the first exp if you like (it's a very small number).

This is a multiplicative model: the contribution of increasing population is to *multiply* predicted number of physicians by something. You can even work out what: if you multiply the population by 2, leaving everything else fixed, you get this:

$$\begin{aligned} \frac{d(2p)}{d(p)} &= \frac{e^{-9.05}(2p)^{1.30}a^{-0.17}}{e^{-9.05}p^{1.30}a^{-0.17}} \\ &= 2^{1.3} = 2.46 \end{aligned}$$

since almost everything cancels: that is, doubling the population slightly more than doubles the number of physicians, if the area of a county is held constant. Doubling the area while holding the population constant, by the same logic, changes the number of physicians by a factor of $2^{-0.17} = 0.89$; that is, making it about 90% of what it was before.

Clearly there is an effect of population density at work here.

(i) To satisfy the curiosity that you probably have, find the ten largest counties by population and then list them. Where do you think the second-largest county is?

Solution: This is actually the same strategy that you would use in R, but implemented differently: make a new data set that is the old one sorted (in descending order) by population, and then display its first ten lines. SAS has a `proc sort` that does this:

```
proc sort;
  by descending pop;

proc print data=county2 (obs=10);
  var name state area pop;
```

```
        Obs    name              state             area            pop

          1    Los_Angeles        CA               4060        8863164
          2    Cook               IL                946        5105067
          3    Harris             TX               1729        2818199
          4    San_Diego          CA               4205        2498016
          5    Orange             CA                790        2410556
          6    Kings              NY                 71        2300664
          7    Maricopa           AZ               9204        2122101
          8    Wayne              MI                614        2111687
          9    Dade               FL               1945        1937094
         10    Dallas             TX                880        1852810
```

What `proc sort` does is to sort the data set by the variable(s) requested and *save it back* in a data set of the same name. That's why my `proc print` worked. (If you don't like that, you put an `out=` on the `proc sort` line with a new data set name.)

All of Los Angeles is in one county, which makes it the biggest one in the entire country. You might not know where Cook County is, but it's in Illinois, and the biggest city in Illinois is Chicago, so you might guess that it includes Chicago. If you look it up on Google Maps, you'll see that you were exactly right.[47]

The other ones that you may not know:

- Harris County is Houston, Texas
- Kings County is Brooklyn, New York
- Maricopa County is Phoenix, Arizona
- Wayne County is Detroit, Michigan
- Dade County is Miami, Florida (and part of the Everglades).

There are 3144 counties in the US altogether, but the top 146 of them contain half the country's population. The smallest county is in Hawaii, and has a population of 88. It's here: `https://www.google.ca/maps/place/Kalawao+County,+HI,+USA/@21.2021259,-156.9809354,13z/data=!3m1!4b1!4m5!3m4!1s0x7eaac65579ddc6f9:0xe77e605e9cb41ca2!8m2!3d21.2273942!4d-156.9749731`. The second smallest is in Texas, here: `https://www.google.ca/maps/place/Loving+County,+TX,+USA/@31.8257709,-103.7953638,11z/data=!3m1!4b1!4m5!3m4!1s0x86e4ce71ef41234d:0xdeecafd0cb46ec56!8m2!3d31.8883434!4d-103.6362715`.

12 Regression with categorical variables

12.1. This is a reorganization of the crickets problem that you may have seen before (minus the data tidying). We have previously done the equivalent of this in R, and we have seen these data (and a lot of these ideas) in class.

Male tree crickets produce "mating songs" by rubbing their wings together to produce a chirping sound. It is hypothesized that female tree crickets identify males of the correct species by how fast (in chirps per second) the male's mating song is. This is called the "pulse rate". Some data for two species of crickets are in `http://www.utsc.utoronto.ca/~butler/c32/crickets2.csv` as a CSV file. The columns are species (text), temperature, and pulse rate (numbers). This is the tidied version of the data set that the previous version of this question had you create.

The research question is whether males of the different species have different average pulse rates. It is also of interest to see whether temperature has an effect, and if so, what.

(a) First, read in and display the data (into SAS).

Solution: Same old same old:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/crickets2.csv";

proc import
  datafile=myurl
  out=crickets
  dbms=csv
  replace;
  getnames=yes;

proc print;
```

```
              Obs    species          temperature      pulse_rate

                1    exclamationis           20.8            67.9
                2    exclamationis           20.8            65.1
                3    exclamationis             24            77.3
                4    exclamationis             24            78.7
                5    exclamationis             24            79.4
                6    exclamationis             24            80.4
                7    exclamationis           26.2            85.8
                8    exclamationis           26.2            86.6
                9    exclamationis           26.2            87.5
               10    exclamationis           26.2            89.1
               11    exclamationis           28.4            98.6
               12    exclamationis             29           100.8
               13    exclamationis           30.4            99.3
               14    exclamationis           30.4           101.7
               15    niveus                  17.2            44.3
               16    niveus                  18.3            47.2
               17    niveus                  18.3            47.6
               18    niveus                  18.3            49.6
               19    niveus                  18.9            50.3
               20    niveus                  18.9            51.8
               21    niveus                  20.4              60
               22    niveus                    21            58.5
               23    niveus                    21            58.9
               24    niveus                  22.1            60.7
               25    niveus                  23.5            69.8
               26    niveus                  24.2            70.9
               27    niveus                  25.9            76.2
               28    niveus                  26.5            76.1
               29    niveus                  26.5              77
               30    niveus                  26.5            77.7
               31    niveus                  28.6            84.7
```

31 crickets, of two different species. Check.

(b) Carry out a two-sample t-test to compare mean pulse rates in the two different species. For reasons we see in a moment, look at the pooled test.

Solution: This is actually a piece of cake (if you remember how to do it):

```
proc ttest;
  var pulse_rate;
  class species;
```

```
species             N        Mean     Std Dev     Std Err     Minimum     Maximum

exclamationis      14     85.5857     11.6993      3.1268     65.1000       101.7
niveus             17     62.4294     12.9568      3.1425     44.3000     84.7000
Diff (1-2)                23.1563     12.4089      4.4784
```

```
species          Method               Mean       95% CL Mean        Std Dev

exclamationis                      85.5857     78.8307  92.3407     11.6993
niveus                             62.4294     55.7676  69.0912     12.9568
Diff (1-2)       Pooled            23.1563     13.9969  32.3157     12.4089
Diff (1-2)       Satterthwaite     23.1563     14.0858  32.2268

            species          Method             95% CL Std Dev

            exclamationis                       8.4815  18.8481
            niveus                              9.6499  19.7194
            Diff (1-2)       Pooled             9.8825  16.6815
            Diff (1-2)       Satterthwaite
        Method           Variances        DF    t Value    Pr > |t|

        Pooled           Equal            29       5.17      <.0001
        Satterthwaite    Unequal      28.719       5.22      <.0001
                           Equality of Variances

             Method      Num DF    Den DF    F Value    Pr > F

             Folded F        16        13       1.23    0.7188
```

Remembering to look at the pooled test, the *t*-statistic and its P-value are the same as we got from R. SAS gives us (at the bottom) a test that the pulse rate variances are the same for each species, and this is not rejected, so using the pooled test is sound.

(c) Reproduce the two-sample *t*-test using a regression predicting pulse rate from species.

Solution: Think carefully here: it's a regression, but the explanatory variable is categorical, so we have to use `proc glm` rather than `proc reg`:

```
proc glm;
  class species;
  model pulse_rate=species / solution;
```

```
                                   The GLM Procedure

                                Class Level Information

                    Class         Levels    Values

                    species            2    exclamationis niveus
                       Number of Observations Read          31
                       Number of Observations Used          31
                                   The GLM Procedure

                          Dependent Variable: pulse_rate

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

 Model                        1     4116.742402     4116.742402      26.74    <.0001

 Error                       29     4465.432437      153.980429

 Corrected Total             30     8582.174839
```

```
          R-Square     Coeff Var      Root MSE    pulse_rate Mean

          0.479685      17.02480      12.40889           72.88710
Source                     DF      Type I SS    Mean Square   F Value   Pr > F

species                     1    4116.742402    4116.742402     26.74   <.0001
Source                     DF    Type III SS    Mean Square   F Value   Pr > F

species                     1    4116.742402    4116.742402     26.74   <.0001
                                                 Standard
Parameter                        Estimate           Error   t Value   Pr > |t|

Intercept                   62.42941176 B     3.00959670     20.74     <.0001
species   exclamationis     23.15630252 B     4.47842320      5.17     <.0001
species   niveus             0.00000000 B      .              .         .
NOTE: The X'X matrix has been found to be singular, and a generalized inverse
      was used to solve the normal equations.  Terms whose estimates are
      followed by the letter 'B' are not uniquely estimable.
```

Look along the line for the `species` that is not the baseline (*exclamationis*): the *t*-statistic is the same 5.17 as the *t*-test gave. Also, if you check the type III sums of squares table above, you'll find that the F-value of 26.74 is the *square* of the *t*-value, and its P-value is the same. This is in some ways an accident, since the regression *assumes* each observation has the same variance, and so the regression will always correspond to the pooled test in this situation, regardless of whether the pooled test is actually the right thing to do.

If we had had more than two species, this would have been like the pigs example in class, where the appropriate test would have been an analysis of variance. The F-statistic from that would have been the same as the F-statistic in the type III sums of squares table. (Again, for regular ANOVA, not for something like Welch ANOVA, for the same reasons as above.)

(d) Fit a regression predicting pulse rate from species and temperature as well. Compare your answers with R's. (Display the graphical output as well.)

Solution: Same idea again: one of the explanatory variables is categorical, so we need to use `proc glm`:

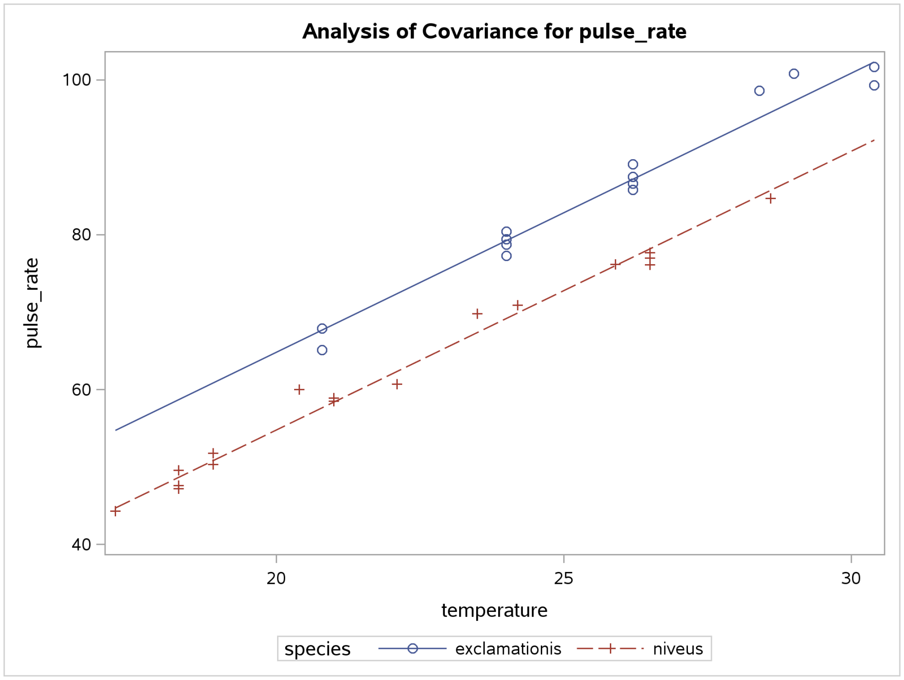
```
proc glm;
  class species;
  model pulse_rate=species temperature / solution;
```

```
                           The GLM Procedure

                        Class Level Information

                Class         Levels    Values

                species            2    exclamationis niveus

                   Number of Observations Read          31
                   Number of Observations Used          31
                           The GLM Procedure

                    Dependent Variable: pulse_rate

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

Model                        2     8492.824970    4246.412485   1330.72   <.0001

Error                       28       89.349869       3.191067

Corrected Total             30     8582.174839
```

```
            R-Square     Coeff Var      Root MSE    pulse_rate Mean

            0.989589      2.450853      1.786356           72.88710
 Source                      DF      Type I SS    Mean Square   F Value   Pr > F

 species                      1    4116.742402    4116.742402   1290.08   <.0001
 temperature                  1    4376.082568    4376.082568   1371.35   <.0001
 Source                      DF    Type III SS    Mean Square   F Value   Pr > F

 species                      1     598.003953     598.003953    187.40   <.0001
 temperature                  1    4376.082568    4376.082568   1371.35   <.0001
                                                          Standard
Parameter                         Estimate               Error    t Value   Pr > |t|

Intercept                     -17.27619743 B      2.19552853      -7.87     <.0001
species     exclamationis      10.06529123 B      0.73526224      13.69     <.0001
species     niveus              0.00000000 B       .                .         .
temperature                     3.60275287        0.09728809      37.03     <.0001

NOTE: The X'X matrix has been found to be singular, and a generalized inverse
      was used to solve the normal equations.  Terms whose estimates are
      followed by the letter 'B' are not uniquely estimable.
```

In the bottom table, the estimates are the same as R's, at least allowing for the fact that the other species was used as the baseline (so the sign got switched). Everything else is consistent.

Here's the graph that comes out:

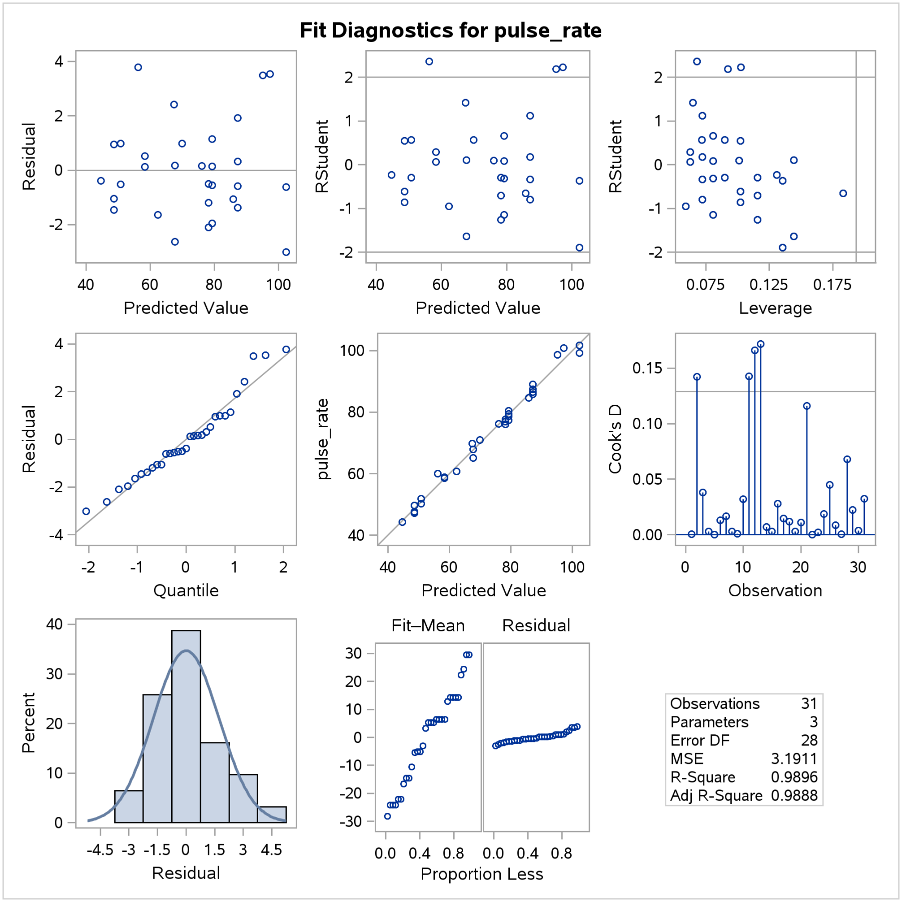

Analysis of Covariance for pulse_rate
pulse_rate
100
80
60
40
20
25
30
temperature
species
exclamationis
niveus

This is like the graph we drew in R, except that here the two lines are *constrained* to come out with the same slope, whether the data support that or not. The fact that it looks the same as R's graph suggests that identical slopes *is* supported by the data.

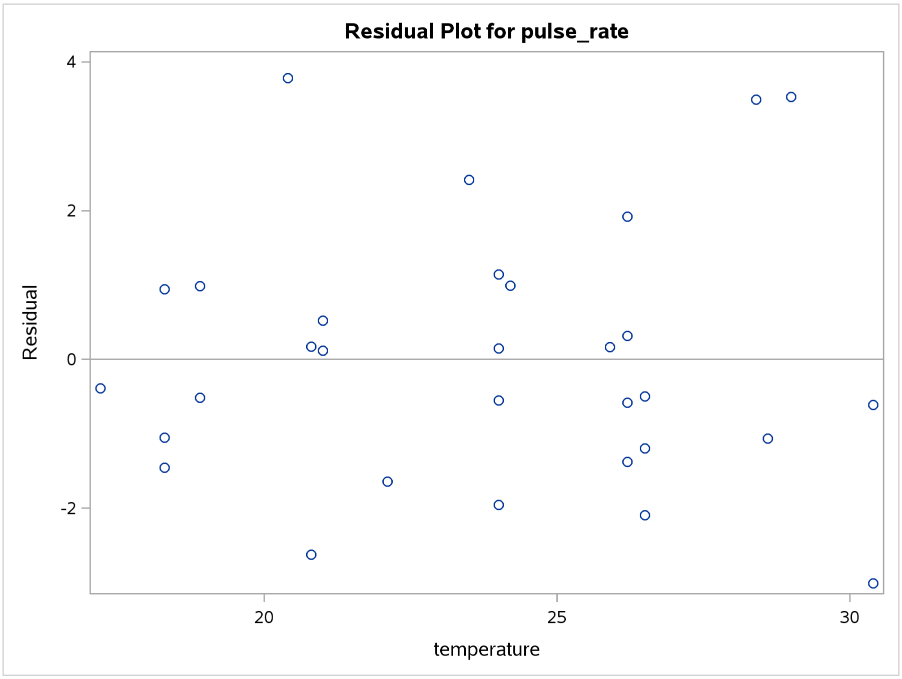
As far as I currently understand (and I am typing this while heading home on the bus, so I can't check just yet), `proc glm` doesn't naturally produce residual plots, because it takes an ANOVA-like approach to the analysis rather than a regression-like one.

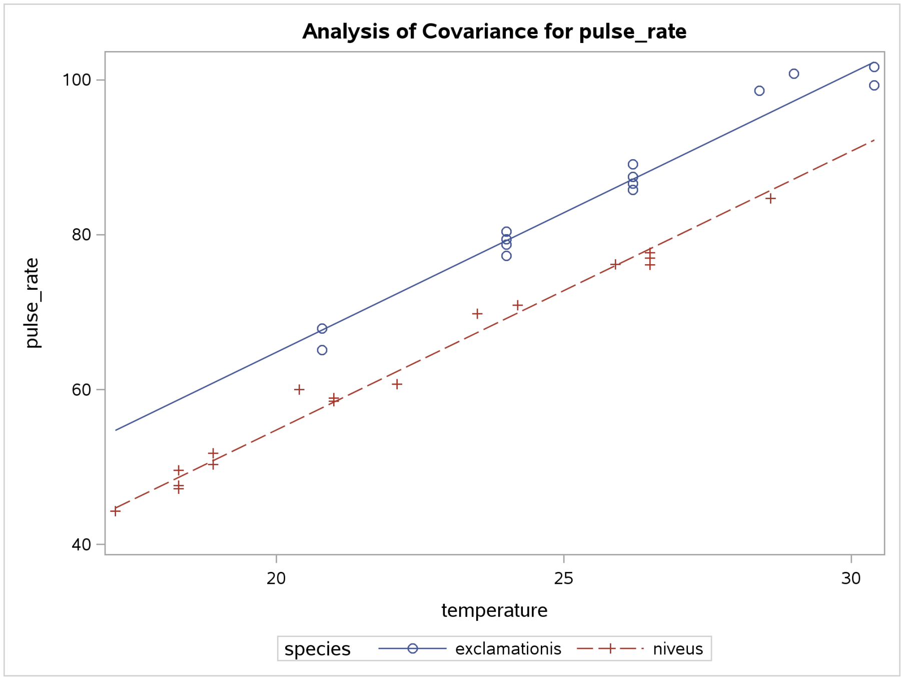
I'm home now. This is how it's done:

```
proc glm plots=(diagnostics residuals);
  class species;
  model pulse_rate=species temperature / solution;
```



Observations	31
Parameters	3
Error DF	28
MSE	3.1911
R-Square	0.9896
Adj R-Square	0.9888




Analysis of Covariance for pulse_rate
pulse_rate
100
80
60
40
20
25
30
temperature
species
exclamationis
niveus

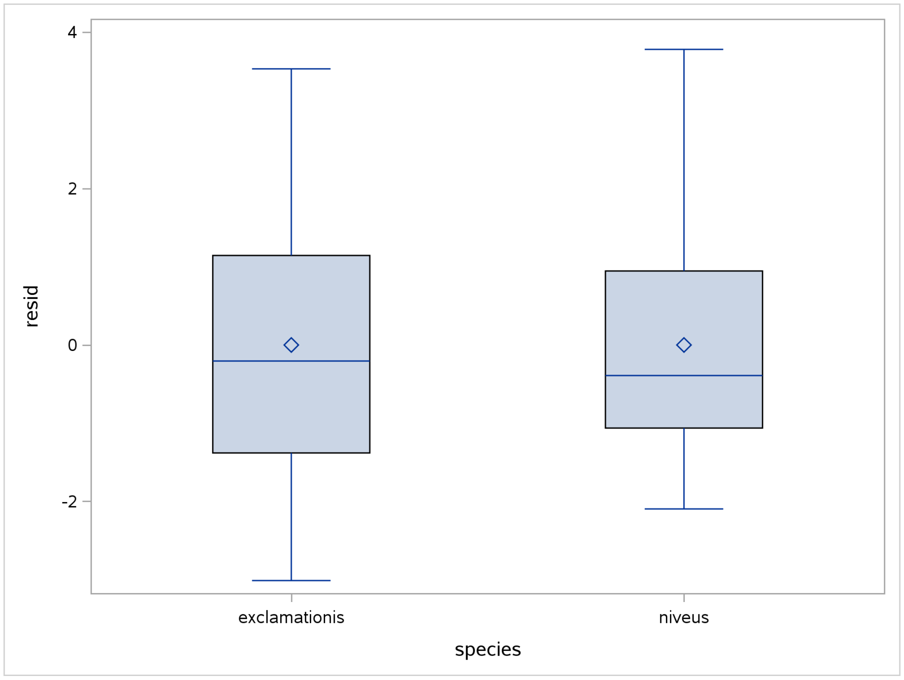
This looks like a regression output. In the first set of plots, we see that the residuals against fitted (top left) look random and the residuals are close to normal (2nd row, 1st plot). Below that, the residuals against temperature also look random. But we don't get a plot of residuals against species (that we made with a boxplot before). To get *that*, I need to do this:

```
proc glm;
  class species;
  model pulse_rate=species temperature / solution;
  output out=res p=fitted r=resid;
```

That makes us a dataset (which becomes the current one) containing all the data plus the fitted values and residuals from this model. This is the same idea as `augment` (from `broom`) in R.

Then we use `proc sgplot` to plot whatever we want to plot, namely this:

```
proc sgplot;
  vbox resid / category=species;
```



A remark: now that I've gotten the output data set with residuals and stuff in it, I could have used that to make all of my residual plots, and dispensed with the `plots` on my `proc glm` earlier. It's normally more convenient to take the plots SAS gives you, but there is no problem in obtaining an output data set and using that to make your plots. If it works, it's good.[48]

12.2. Have you gone outside in the summer and been bitten by mosquitoes? *Consumer Reports* magazine tested 14 products that all claim to be an effective mosquito repellent. Each product was classified as either a lotion/cream or an aerosol/spray. The cost per use of each product was calculated by dividing the total cost by the amount of product that was needed to cover exposed areas of the skin. Human testers applied each product and the hours of protection provided by the product was measured by exposing the arms to 200 mosquitoes. The data from the report are in `https://www.utsc.utoronto.ca/~butler/c32/repellent.txt`.

(a) Read the data into SAS and display what you have.

Solution: As I originally got the data, the values were aligned in columns, which `proc import` won't read in. So I re-saved them delimited by single spaces. You ought to check by looking at the file that this is what you have, so that you read it in correctly:

```
filename myurl url "https://www.utsc.utoronto.ca/~butler/c32/repellent.txt";

proc import
  datafile=myurl
  out=repellent
  dbms=dlm
  replace;
  getnames=yes;
  delimiter=' ';

proc print;
```

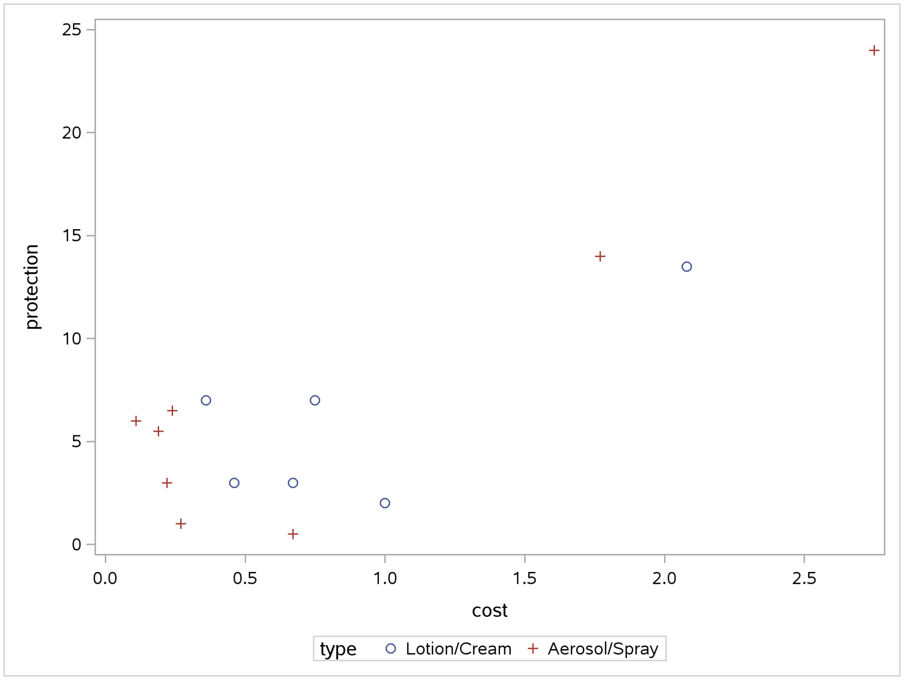
```
        Obs    type                    cost      protection

          1    Lotion/Cream            2.08            13.5
          2    Aerosol/Spray           0.67             0.5
          3    Lotion/Cream               1               2
          4    Lotion/Cream            0.75               7
          5    Lotion/Cream            0.46               3
          6    Aerosol/Spray           0.11               6
          7    Aerosol/Spray           0.22               3
          8    Aerosol/Spray           0.19             5.5
          9    Aerosol/Spray           0.24             6.5
         10    Aerosol/Spray           0.27               1
         11    Aerosol/Spray           1.77              14
         12    Lotion/Cream            0.67               3
         13    Lotion/Cream            0.36               7
         14    Aerosol/Spray           2.75              24
```

There are 14 rows and the right columns with apparently the right values, so I call this good.

(b) Our aim is to predict hours of protection from the other variables. Draw a graph that will show how the variables are related.

Solution: There are three variables, two quantitative and one categorical, so a scatterplot with the categorical variable distinguished eg. by colour is the way to go. This is the same idea as our example plot of weight vs. height by gender (for the Australian athletes data), so you can mimic the idea there:

```
proc sgplot;
  scatter x=cost y=protection / group=type;
```



(c) Would you say that there are typically more hours of protection if the cost per application of an insect repellent is larger? Explain briefly.

Solution: You could reasonably say "yes", because the scatterplot shows an upward trend. That will do fine for an answer.

Another direction you could take is to say that the three very expensive repellents do give a longer protection time, but there is pretty much no relationship for the other 11 repellents at the bottom left. (That would be saying that there is a distinction between "cheap" and "expensive" but nothing more nuanced than that.)

(d) Would you say that there is a distinction between the two types of repellent in terms of protection, allowing for differences in cost? Explain briefly.

Solution: I would say that there is not, because if you compare repellents of similar cost but of different types have about the same hours of protection. (You need to get at the idea of whether type says anything about protection *over and above* what cost says, and I think the answer to that is no.)

If you can make a different case convincingly, then go for it.

(e) Fit a regression predicting hours of protection from cost (that is, not including the type of repellent), and find the piece of the output showing the intercept and slope and their t-tests, and the piece showing R-squared.

Solution: This is the output from a suitable `proc reg`, but you have to pick out what you want.

```
proc reg;
  model protection=cost;
```

You can include all the text output (not the graphs):

```
                          The REG Procedure
                            Model: MODEL1
                    Dependent Variable: protection

                 Number of Observations Read          14
                 Number of Observations Used          14
                          Analysis of Variance

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

Model                      1       386.13694      386.13694      31.19    0.0001
Error                     12       148.57735       12.38145
Corrected Total           13       534.71429
             Root MSE              3.51873    R-Square     0.7221
             Dependent Mean        6.85714    Adj R-Sq     0.6990
             Coeff Var            51.31478
                          Parameter Estimates

                          Parameter       Standard
    Variable      DF       Estimate          Error    t Value    Pr > |t|

    Intercept      1        1.31386        1.36736       0.96      0.3556
    cost           1        6.72495        1.20421       5.58      0.0001
```

or just the bit from Root MSE down:

```
             Root MSE              3.51873    R-Square     0.7221
             Dependent Mean        6.85714    Adj R-Sq     0.6990
             Coeff Var            51.31478
                          Parameter Estimates

                          Parameter       Standard
    Variable      DF       Estimate          Error    t Value    Pr > |t|

    Intercept      1        1.31386        1.36736       0.96      0.3556
    cost           1        6.72495        1.20421       5.58      0.0001
```

Your choice.

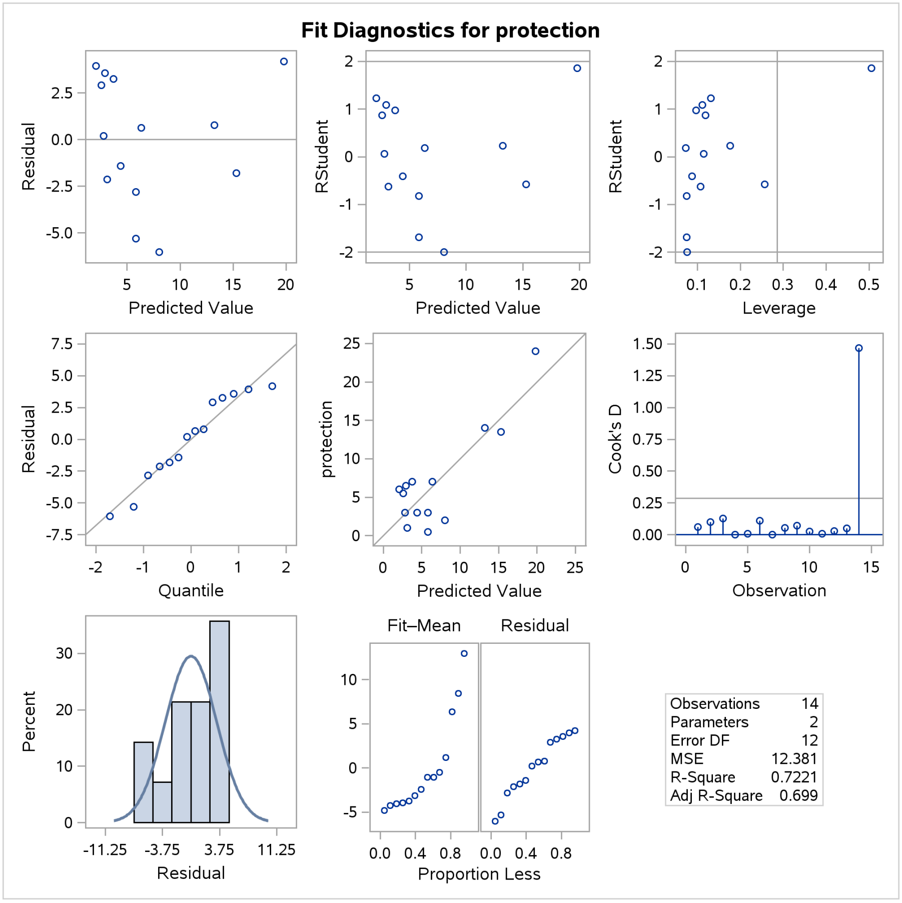
(f) Does `cost` contribute significantly to the prediction of hours of protection? Does that make sense, given what you saw on your graph? Explain briefly.

Solution: The P-value of `cost` in the regression is a very small 0.0001, so it *does* contribute significantly to the prediction (that is, `protection` *does* depend on `cost`). This is completely consistent with the upward trend on the scatterplot.

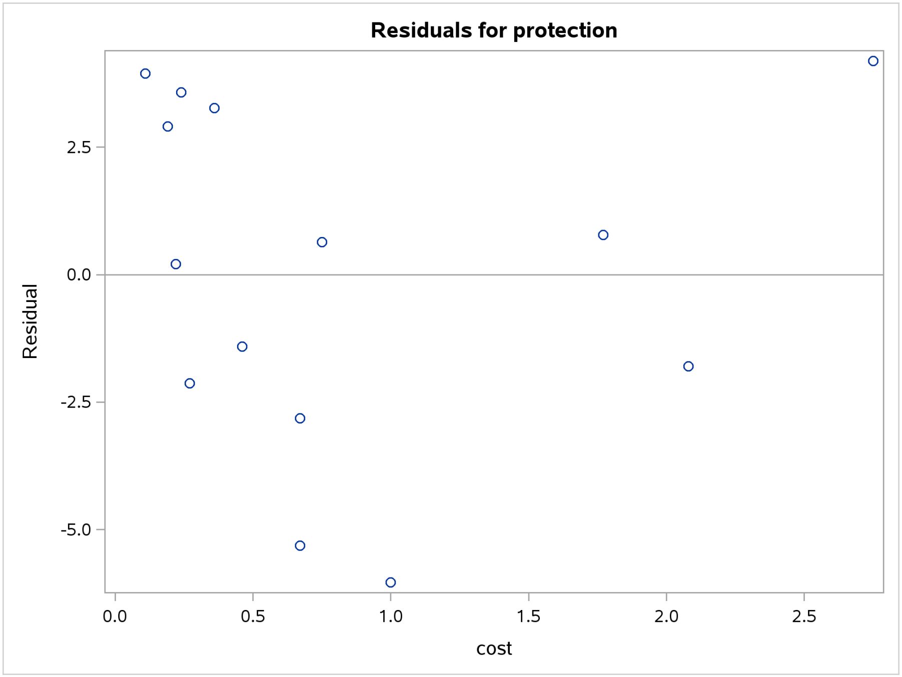
If you thought that it wasn't really a linear trend on the scatterplot, just a distinction between the cheap repellents and the expensive ones, then you can argue as above, or you can say that a linear regression is not the right thing to fit, and therefore we shouldn't trust the P-value here. I'd go with either of these two approaches.

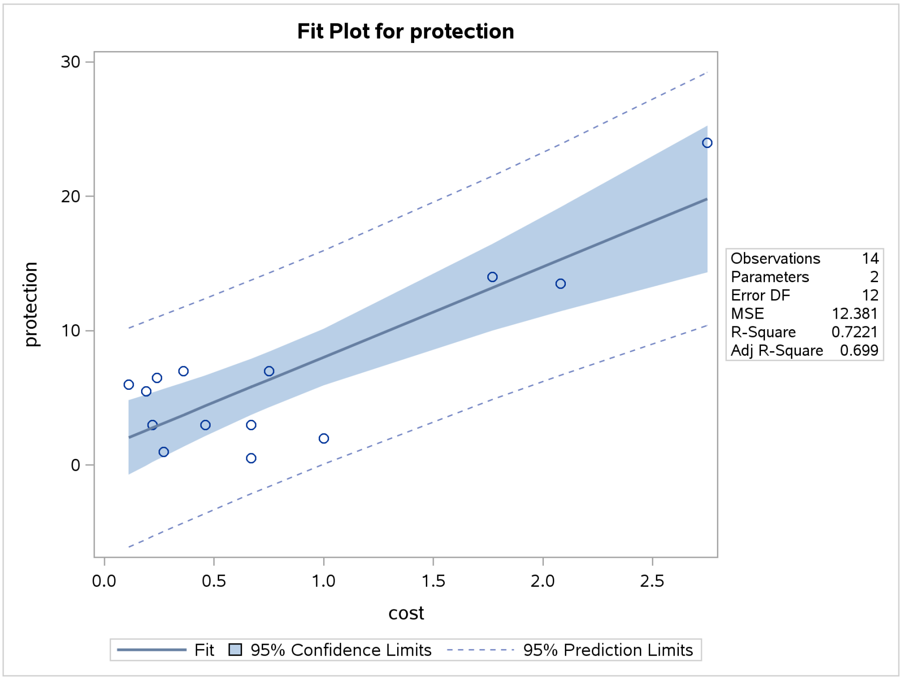
Extra: there are several other issues here that would be worth exploring, but which make the question a bit long for an assignment.

The first of those is whether we trust a linear regression at all. Another way to look at that is to examine the residual plots from the regression (the residuals vs. fitted values probably being the most interesting):



```
Observations      14
Parameters         2
Error DF          12
MSE           12.381
R-Square      0.7221
Adj R-Square   0.699
```





```
Observations      14
Parameters         2
Error DF          12
MSE           12.381
R-Square      0.7221
Adj R-Square   0.699
```

You might make the call that this top left plot shows a down-and-up curve. This would be consistent with the idea that for the “cheap” repellents, the more expensive ones of those don’t offer any more protection than the cheaper of the cheaper ones. (If a linear model is appropriate, any increase in cost ought to go with an increase in protection, at least on average, but if you compare the repellents that cost about \$0.75 per use vs. the ones that cost less than \$0.50, there really isn’t much difference in protection on the scatterplot.)

The second thing, even given a linear relationship with cost, is whether `type` makes any difference to protection. We suspected from looking at the scatterplot that it doesn’t. To model this, though, we need to borrow an idea from the crickets question: `type` is categorical, so we have to use `proc glm` rather than `proc reg`:

```
proc glm;
  class type;
  model protection=cost type;
```

This is the text part of the output:

```
                          The GLM Procedure

                       Class Level Information

             Class         Levels    Values

             type               2    Aerosol/Spray Lotion/Cream
                   Number of Observations Read          14
                   Number of Observations Used          14
                          The GLM Procedure

                     Dependent Variable: protection

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

 Model                        2    405.6506906    202.8253453     17.29   0.0004

 Error                       11    129.0635951     11.7330541

 Corrected Total             13    534.7142857
            R-Square     Coeff Var      Root MSE    protection Mean

            0.758631      49.95309      3.425355           6.857143
 Source                      DF      Type I SS    Mean Square   F Value   Pr > F

 cost                         1    386.1369387    386.1369387     32.91   0.0001
 type                         1     19.5137520     19.5137520      1.66   0.2236
 Source                      DF    Type III SS    Mean Square   F Value   Pr > F

 cost                         1    396.3634883    396.3634883     33.78   0.0001
 type                         1     19.5137520     19.5137520      1.66   0.2236
```

The ANOVA table with Model in it says that *something* predicts hours of protection, and the bottom table, the one of Type III sums of squares, says that it's `cost` but *not* `type`: the type of repellent has no effect on the hours of protection once you allow for the effect of cost. This is why I had you make a careful assessment on the scatterplot of the effect of `type`; the way an effect of `type` would show up there is that *if you hold cost constant* (that is, comparing repellents with similar cost), does one type consistently offer more protection than the other? The answer to that is clearly "no": both from the scatterplot and from this regression (which is strictly a "general linear model").

The last thing I wanted to investigate was my thought that the effect of `cost` was not a linear one, but a distinction into "cheap" and "expensive" repellents. That means creating a new cost variable that is a categorical version of the old one, dividing let's say at $1.50 since there's a big gap there. In SAS, that means creating a new data set. We'll call our new variable `cost_new` (note the lack of imagination!), and we'll have to use `if-then-else` SAS-style, as compared to defining a new variable that is just true or false by setting it equal to a logical condition. (The way below is the SAS version of `ifelse` in a `mutate`.)

```
data repellent2;
  set repellent;
  if (cost>1.50) then cost_new="expensive";
  else cost_new="cheap";

proc print;
```

Did it work?

```
 Obs    type                      cost      protection    cost_new

   1    Lotion/Cream              2.08            13.5    expensive
   2    Aerosol/Spray             0.67             0.5    cheap
   3    Lotion/Cream                 1               2    cheap
   4    Lotion/Cream              0.75               7    cheap
   5    Lotion/Cream              0.46               3    cheap
   6    Aerosol/Spray             0.11               6    cheap
   7    Aerosol/Spray             0.22               3    cheap
   8    Aerosol/Spray             0.19             5.5    cheap
   9    Aerosol/Spray             0.24             6.5    cheap
  10    Aerosol/Spray             0.27               1    cheap
  11    Aerosol/Spray             1.77              14    expensive
  12    Lotion/Cream              0.67               3    cheap
  13    Lotion/Cream              0.36               7    cheap
  14    Aerosol/Spray             2.75              24    expensive
```

It did.

Next, we put both categorical variables `type` and `cost_new` in a `proc glm`:

```
proc glm;
  class type;
  class cost_new;
  model protection=type cost_new / solution;
```

```
                               The GLM Procedure

                            Class Level Information

               Class         Levels    Values

               type               2    Aerosol/Spray Lotion/Cream

               cost_new           2    cheap expensive
                     Number of Observations Read          14
                     Number of Observations Used          14
                               The GLM Procedure

                         Dependent Variable: protection

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

 Model                       2     406.8772321    203.4386161     17.51   0.0004

 Error                      11     127.8370536     11.6215503

 Corrected Total            13     534.7142857
           R-Square     Coeff Var      Root MSE    protection Mean

           0.760925      49.71516      3.409040           6.857143
 Source                     DF       Type I SS    Mean Square   F Value   Pr > F

 type                        1       9.2872024      9.2872024      0.80   0.3905
 cost_new                    1     397.5900298    397.5900298     34.21   0.0001
 Source                     DF     Type III SS    Mean Square   F Value   Pr > F

 type                        1       1.0568858      1.0568858      0.09   0.7686
 cost_new                    1     397.5900298    397.5900298     34.21   0.0001
                                                   Standard
 Parameter                      Estimate             Error   t Value   Pr > |t|

 Intercept                   16.79464286 B     2.32286887      7.23     <.0001
 type      Aerosol/Spray      0.55803571 B     1.85046124      0.30     0.7686
 type      Lotion/Cream       0.00000000 B      .               .        .
 cost_new  cheap            -13.05357143 B     2.23174022     -5.85     0.0001
 cost_new  expensive          0.00000000 B      .               .        .
NOTE: The X'X matrix has been found to be singular, and a generalized inverse
      was used to solve the normal equations.  Terms whose estimates are
      followed by the letter 'B' are not uniquely estimable.
```

This shows that more expensive repellents do differ in protection time than cheap ones (from the Type III SS table), but that again the type of repellent has no effect on the cost.

The bottom table of parameter estimates tells you *what* effect the categorical variables have on the protection type. The last two lines say that the cheap repellents offer about 13 hours *less* protection time on average than the expensive ones, and that this difference is (strongly) significant. This test is not relying on there being a linear relationship between cost and protection time, merely on the division of the repellents into “cheap” and “expensive” ones.

The message for you the consumer seems to be that you get a longer relief from mosquitoes from the repellents that cost more per use, and then you have to decide whether that extra cost is worthwhile. But at least now you have some kind of basis for making your decision.

13 Dates and times and other miscellanea

13.3. In Denali National Park, Alaska, the size of the wolf population depends on the size of the caribou population (since wolves hunt and kill caribou). This is a large national park, so caribou are found in very large herds, so big, in fact, that the well-being of the entire herd is not threatened by wolf attacks.[49] Can the size of the caribou population be used to predict the size of the wolf population?

The data can be found at `http://www.utsc.utoronto.ca/~butler/c32/caribou.txt`. The columns are: the date of the survey,[50] the name of the park employee in charge of the survey, the caribou population (in hundreds) and the wolf population (actual count).[51]

We are going to use SAS for this question, but this format of data file is one we don’t know how to read into SAS, so we are going to use R to help us first.

(a) Take a look at the data file. How would you describe its format? Read it into R, and check that you got something sensible.

Solution: This looks at first sight as if it’s separated by spaces, but most of the data values are separated by *more than one* space. If you look further, you’ll see that the values are *lined up in columns*, with the variable names aligned at the top. This is exactly the kind of thing that `read_table` will read. We start with the usual `library(tidyverse)`:

```
library(tidyverse)
my_url="http://www.utsc.utoronto.ca/~butler/c32/caribou.txt"
denali=read_table(my_url)
```

```
## Parsed with column specification:
## cols(
##   date = col_character(),
##   name = col_character(),
##   caribou = col_double(),
##   wolf = col_double()
## )
```

```
denali
```

```
## # A tibble: 7 x 4
##   date       name           caribou  wolf
##   <chr>      <chr>            <dbl> <dbl>
## 1 09/01/1995 David S.            30    66
## 2 09/24/1996 Youngjin K.         34    79
## 3 10/03/1997 Srinivasan M.       27    70
## 4 09/15/1998 Lee Anne J.         25    60
## 5 09/08/1999 Stephanie T.        17    48
## 6 09/03/2000 Angus Mc D.         23    55
## 7 10/06/2001 David S.            20    60
```

That worked: four columns with the right names, and the counts of caribou and wolf are numbers. The only (small) weirdness is that the dates are text rather than having been converted into dates. This is because they are not year-month-day, which is the only format that gets automatically converted into dates when read in. (You could use `mdy` from `lubridate` to make them dates.)

(b) Save your R data frame as a `.csv` file. This goes using `write_csv`, which is the exact opposite of `read_csv`. It takes two things: a data frame to save as a `.csv`, and the name of a file to save it in.

Solution: You probably haven't seen this before, so I hope I gave you some clues:

```
write_csv(denali,"denali.csv")
```

This saves the data frame as a `.csv` into your project folder on `rstudio.cloud`. You can go to the Files pane and click on it to open it. Select View File: this will open the actual file in a new tab, so you can see what it looks like, namely this:

```
date,name,caribou,wolf
09/01/1995,David S.,30,66
09/24/1996,Youngjin K.,34,79
10/03/1997,Srinivasan M.,27,70
09/15/1998,Lee Anne J.,25,60
09/08/1999,Stephanie T.,17,48
09/03/2000,Angus Mc D.,23,55
10/06/2001,David S.,20,60
```

The columns don't line up any more, but there are no extra spaces, so that reading this into SAS as a `.csv` should go smoothly.

Download the file from `rstudio.cloud` to your computer. To do this, go back to the Files pane, and click the checkbox to the left of `denali.csv` (or whatever you called it). Then click on More (above the Files pane) and Export, then click Download. It will go to your Downloads folder, or wherever downloaded things go.

If you are running R Studio on your computer, it will be in the folder associated with the project you're in now, and you can find it there.

Then upload the `.csv` file to SAS Studio.

(c) Read your `.csv` file into SAS, and list the values.

Solution: This is the usual `proc import`. First, though, make sure you have uploaded the `.csv` from wherever it is now to your account on SAS Studio, and then, with your username rather than mine:

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

and then

```
proc print;
```

```
       Obs          date    name                  caribou          wolf

         1    09/01/1995    David S.                   30            66
         2    09/24/1996    Youngjin K.                34            79
         3    10/03/1997    Srinivasan M.              27            70
         4    09/15/1998    Lee Anne J.                25            60
         5    09/08/1999    Stephanie T.               17            48
         6    09/03/2000    Angus Mc D.                23            55
         7    10/06/2001    David S.                   20            60
```

Go searching in the log tab for `proc import`, and below it you'll see some lines with `format` on them. This tells you how the variables were read. Mine is:

```
1596              informat date mmddyy10. ;
1597              informat name $13. ;
1598              informat caribou best32. ;
1599              informat wolf best32. ;
1600              format date mmddyy10. ;
1601              format name $13. ;
1602              format caribou best12. ;
1603              format wolf best12. ;
```

The dates were correctly deduced to be month-day-year, the names as text, and the caribou and wolf counts as numbers (that's what the `best` followed by a number is).

These appear to be duplicated because the `informat` lines are how the values are read in, and the `format` lines are how they are listed out (by default). These are not necessarily the same.

(d) Display the data set with the dates in Canadian/British format, day before month.

Solution: Add a `format` to the `proc print`. The right one is `ddmmyy10.`, to get the day before the month; a width of 10 characters gives room for the slashes and 4-digit years.

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

```
 Obs          date    name                 caribou          wolf

   1    01/09/1995    David S.                  30            66
   2    24/09/1996    Youngjin K.               34            79
   3    03/10/1997    Srinivasan M.             27            70
   4    15/09/1998    Lee Anne J.               25            60
   5    08/09/1999    Stephanie T.              17            48
   6    03/09/2000    Angus Mc D.               23            55
   7    06/10/2001    David S.                  20            60
```

Compare this one:

```
proc print;
  format date ddmmyy8.;
```

```
 Obs        date    name                 caribou          wolf

   1    01/09/95    David S.                  30            66
   2    24/09/96    Youngjin K.               34            79
   3    03/10/97    Srinivasan M.             27            70
   4    15/09/98    Lee Anne J.               25            60
   5    08/09/99    Stephanie T.              17            48
   6    03/09/00    Angus Mc D.               23            55
   7    06/10/01    David S.                  20            60
```

This one has only two-digit years, leaving us prone to the "Y2K problem",[52] where it is not clear which century each year belongs to.

(e) Display the data set in such a way that you see the days of the week and the month names for the dates.

Solution: I left this open to you to make the precise choice of format, but one possibility is this one:

```
proc print;
  format date weekdate20.;
```

```
Obs                  date    name                 caribou          wolf

  1     Fri, Sep 1, 1995    David S.                  30            66
  2    Tue, Sep 24, 1996    Youngjin K.               34            79
  3     Fri, Oct 3, 1997    Srinivasan M.             27            70
  4    Tue, Sep 15, 1998    Lee Anne J.               25            60
  5     Wed, Sep 8, 1999    Stephanie T.              17            48
  6     Sun, Sep 3, 2000    Angus Mc D.               23            55
  7     Sat, Oct 6, 2001    David S.                  20            60
```

The number on the end of `weekdate` is how many characters SAS uses to display the date. It makes the best of the space you give it:

```
proc print;
  format date weekdate5.;
```

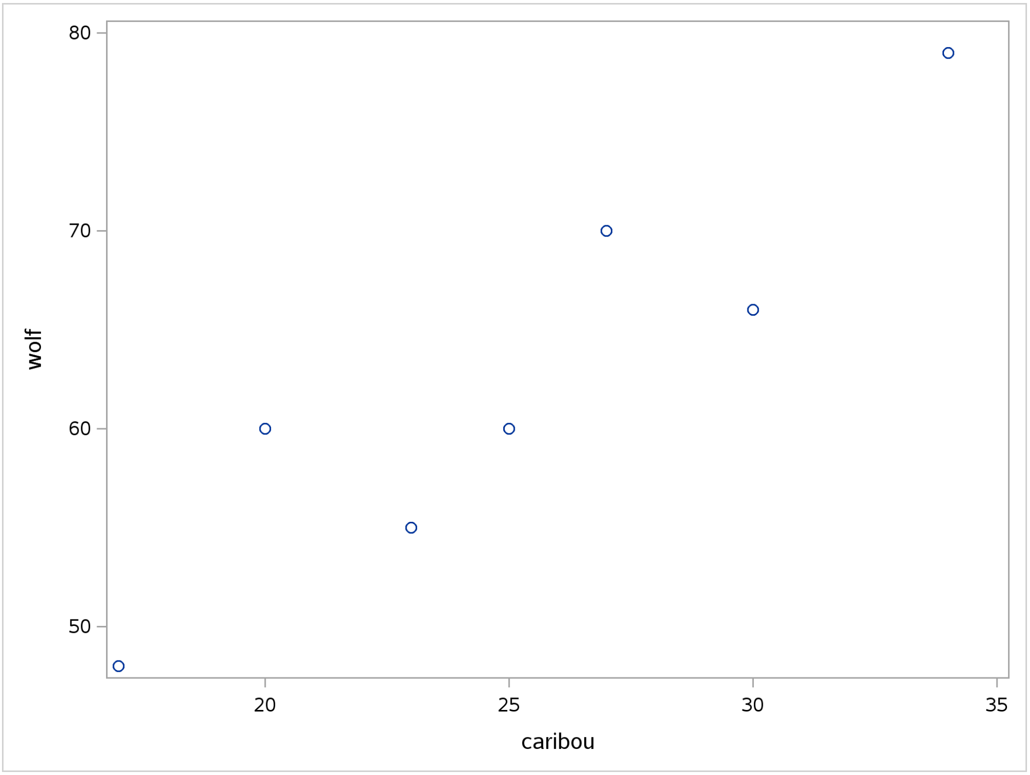
```
         Obs     date     name                 caribou          wolf

           1      Fri     David S.                  30            66
           2      Tue     Youngjin K.               34            79
           3      Fri     Srinivasan M.             27            70
           4      Tue     Lee Anne J.               25            60
           5      Wed     Stephanie T.              17            48
           6      Sun     Angus Mc D.               23            55
           7      Sat     David S.                  20            60
```

The best it can do is to show you the day of the week.

(f) Enough playing around with dates. Make a scatterplot of caribou population (explanatory) against wolf population (response). Do you see any relationship?

Solution: The usual with `proc sgplot`:

```
proc sgplot;
  scatter x=caribou y=wolf;
```



That looks like an upward trend: when the caribou population is large, the wolf population is large too.[53]

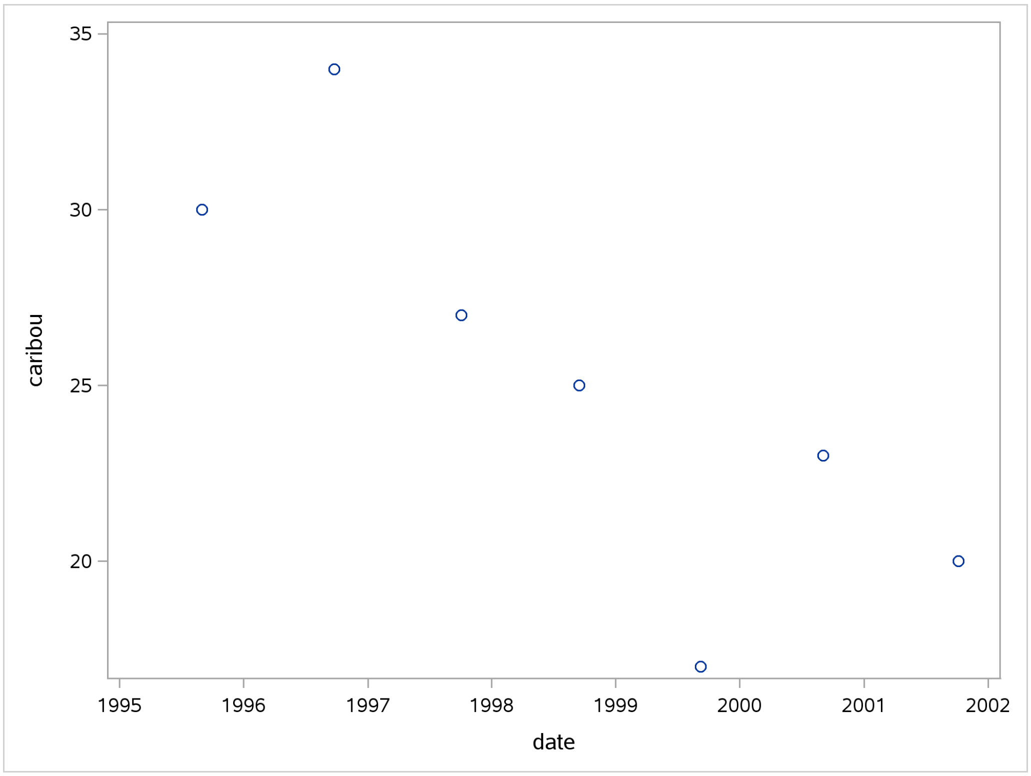
I should point out that the wolf and caribou surveys were taken at different times of the year. The dates in the data file were actually of the caribou surveys (in the fall, as I said). The wolf surveys were taken in the "late winter" (of the following year). This makes sense if you think of the wolf population as varying as a response to the caribou population; you need to allow some time for this "response" to happen. It might even be that the response happens over a longer time frame than this, if you think of the time required for wolves to have and raise pups,[54] which might be a period of years. In the grand scheme of things, there might be a multi-year cyclic variation in caribou and wolf populations; they go up and down together, but there might also be a time lag.

According to `http://www.pbs.org/wnet/nature/river-of-no-return-gray-wolf-fact-sheet/7659/`, wolves in the wild typically live 6–8 years, but many die earlier, often of starvation (so the size of the population of the wolves' prey animals matters a lot).

(g) Make a plot of caribou population against time (this is done the obvious way). What seems to be happening to the caribou population over time?

Solution: Make a scatterplot, with the survey date as explanatory variable, and caribou population as response (since time always goes on the x-axis):

```
proc sgplot;
  scatter x=date y=caribou;
```



A coding note here: I didn't need a `format` on my `proc sgplot`, because the dates are already formatted (from `proc import`). If they had been dates that we constructed ourselves, eg. from year, month and day as numbers, they would not have come with a format, and we would have had to supply one when we printed or plotted them.

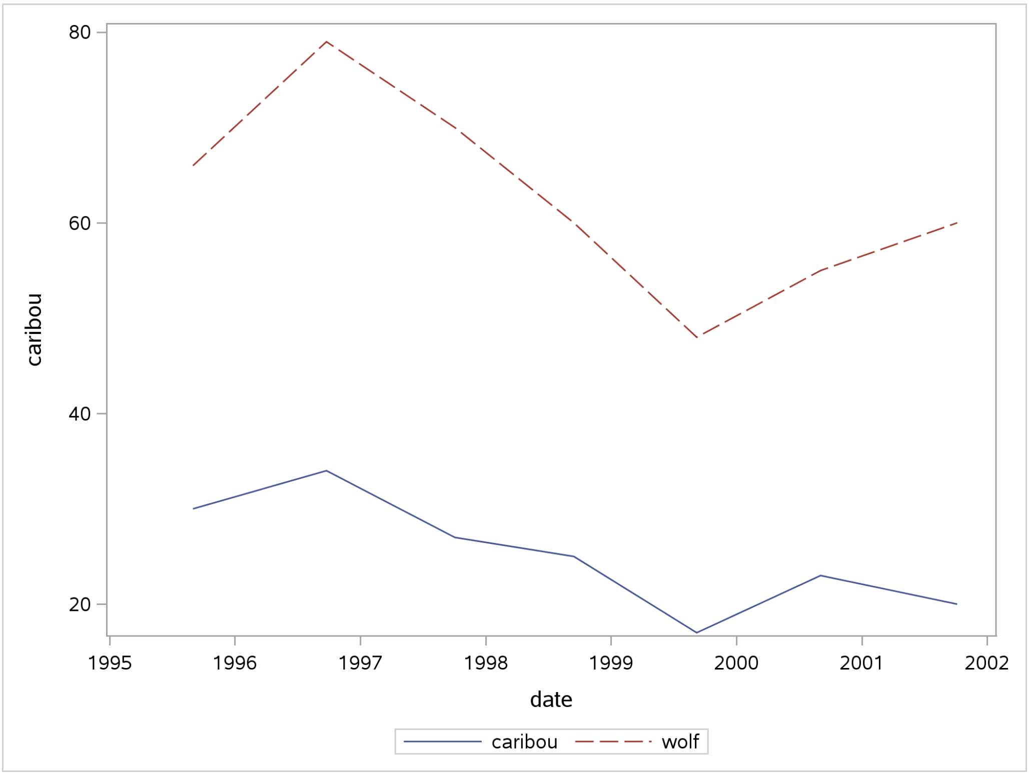
The caribou population is declining over time. We only have seven years' data, though, so it's not clear whether this is to do with climate change or some multi-year cycle in which wolf and caribou populations go up and down together, and we just happen to have hit the "down" part of the cycle.

Where the tick marks are on the x-axis mark the *start* of the year in question, so that the surveys come correctly about $\frac{3}{4}$ of the way through the year. SAS displayed just the years on the x-axis, since that was the scale of the data. If our dates happened to be all one year or all one month, you would have seen more of the format.

(h) The caribou and wolf populations over time are really "time series", so they can be plotted against time by `series` instead of `scatter`. Make a plot of *both* the caribou and wolf populations against time. (That is, use one `sgplot` with two `series` lines, where the `series` lines look just as `scatter` lines would.) How is `series` different from `scatter`?

Solution: I tried to give you enough hints:

```
proc sgplot;
  series x=date y=caribou;
  series x=date y=wolf;
```



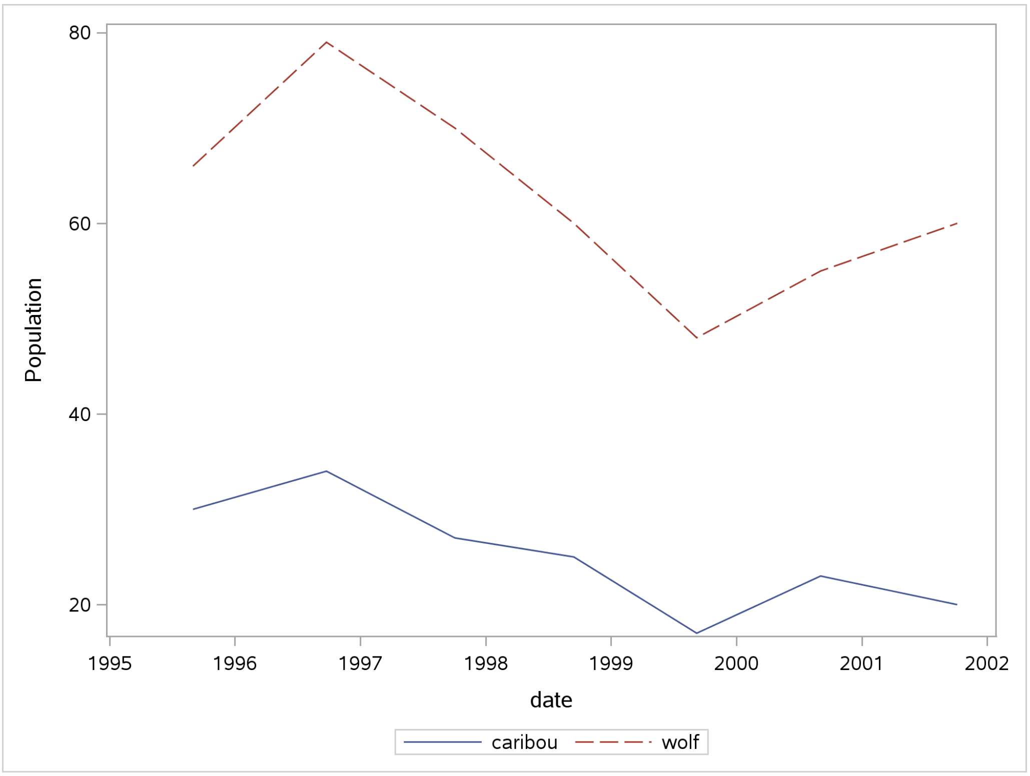
The big difference is that the points are joined by lines, each one to the next one in time order, so that the time nature of the data is more apparent. (Without the lines, it could be much less obvious which data value belongs to which series.)

Because we plotted two series, we also get a legend, and the two series are distinguished by colour and line type.

You should probably recall the scales: the caribou population was measured in hundreds, so the caribou numbers are a lot bigger than the wolf numbers. Evidently they measured caribou population in hundreds to get comparable numbers with the wolf population. In fact, I expect that park officials produced a graph very like this one. Probably in Excel, though.[55]

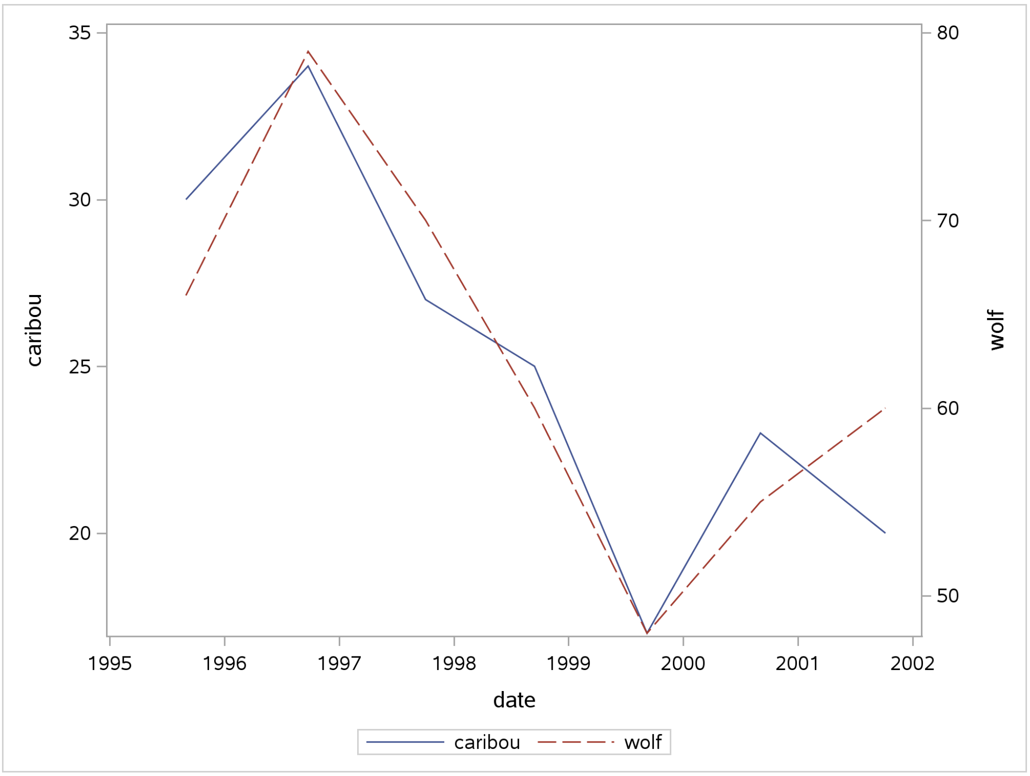
I should have labelled the y-axis "population". That can be done:

```
proc sgplot;
  series x=date y=caribou;
  series x=date y=wolf;
  yaxis label='Population';
```



This is, in fact, one of those rare cases where we can justify having a second y-axis: left one for caribou, right one for wolf. Be aware, though, that you can scale the second y-axis how you like, which means that you can obtain a variety of apparent relationships between the two variables. This is the right way to do it (letting SAS choose the scale):

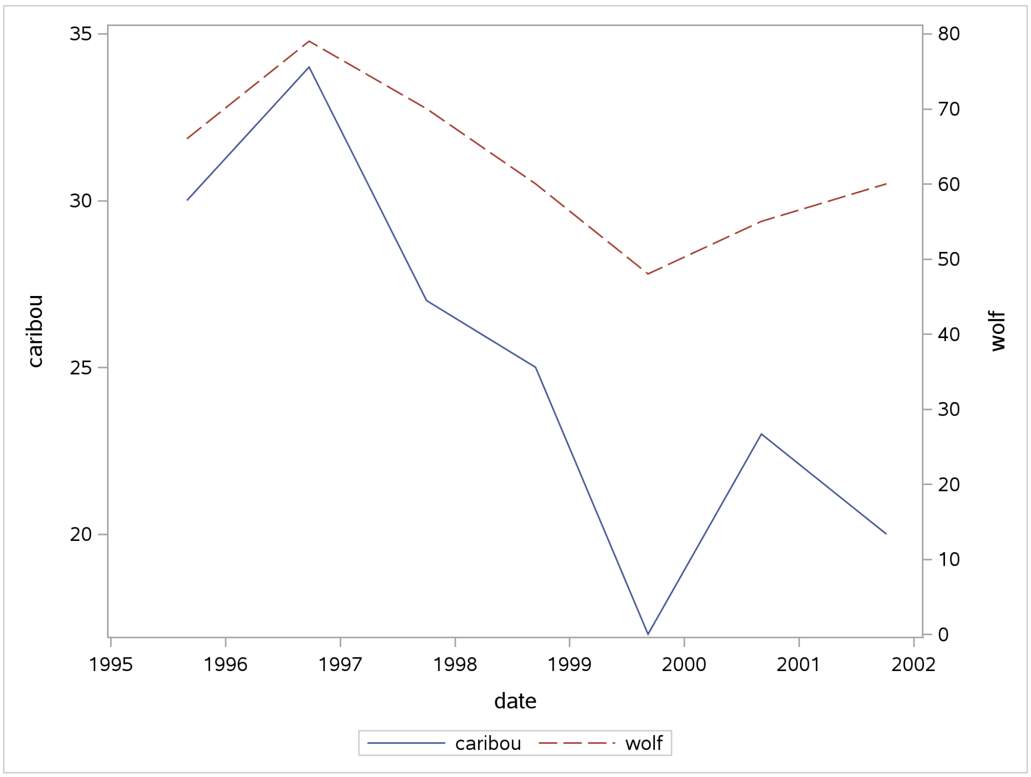
```
proc sgplot;
  series x=date y=caribou;
  series x=date y=wolf / y2axis;
```



This makes it even clearer that caribou and wolf populations rise and fall together.

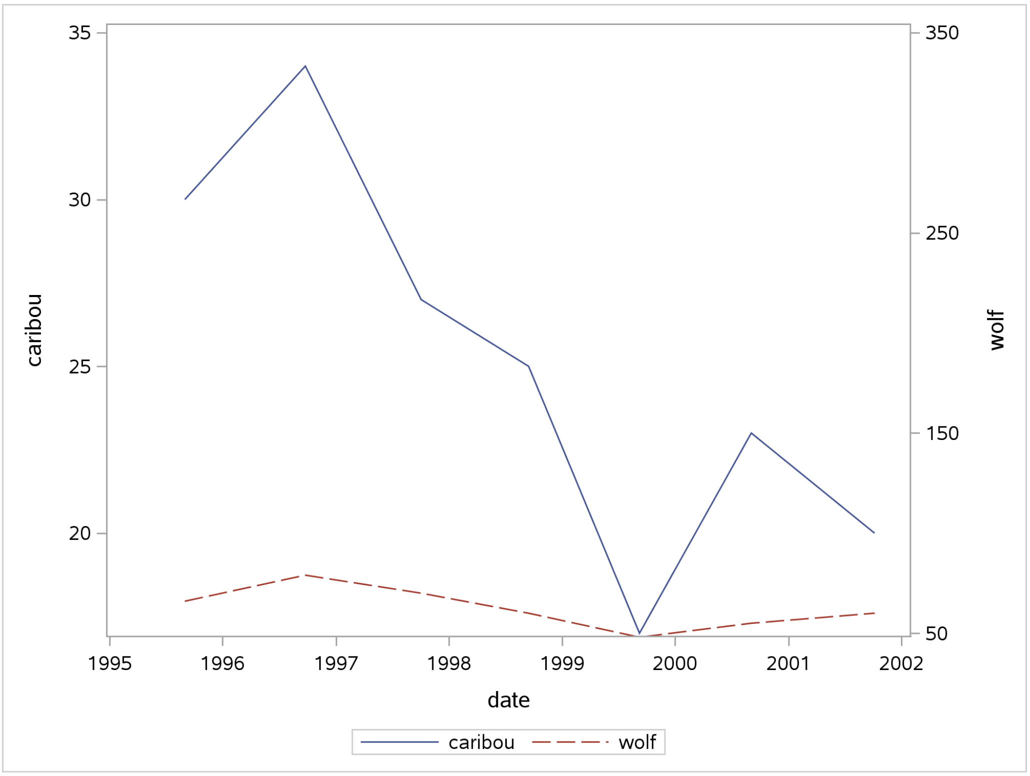
This is about the only double y-axis setup that I like, because you can choose the scale of the second y-axis however you like, to make it look as if the wolf population is very big:

```
proc sgplot;
  y2axis values=(0 to 80 by 10);
  series x=date y=caribou;
  series x=date y=wolf / y2axis;
```



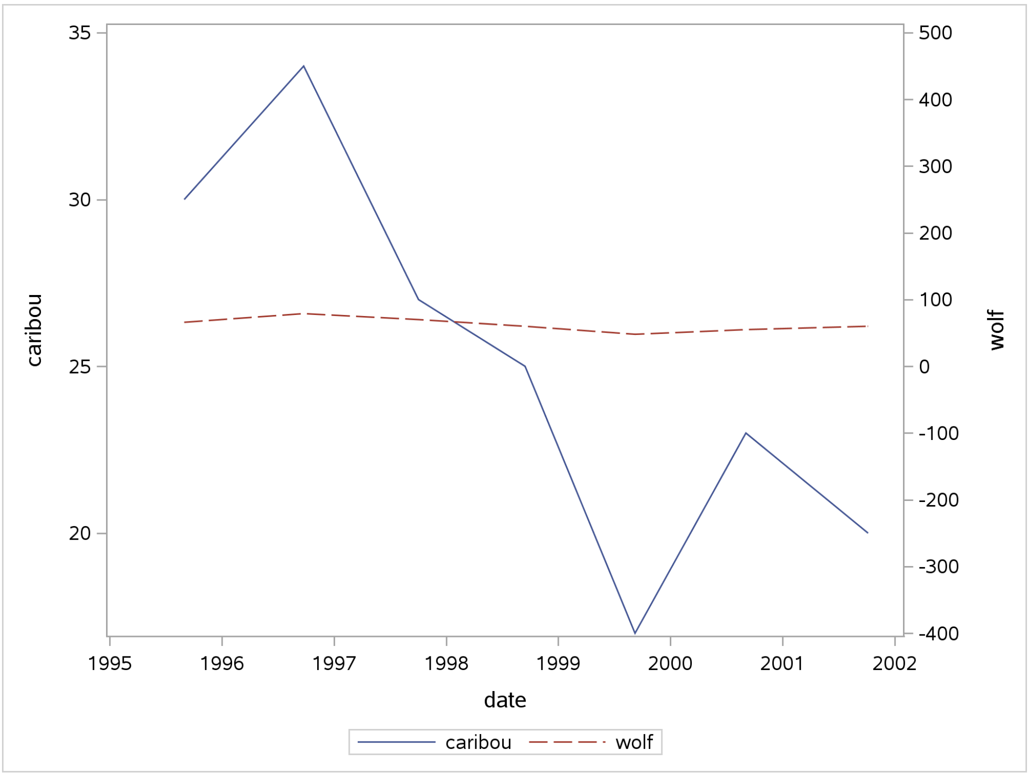
or very small:

```
proc sgplot;
  y2axis values=(50 to 400 by 100);
  series x=date y=caribou;
  series x=date y=wolf / y2axis;
```



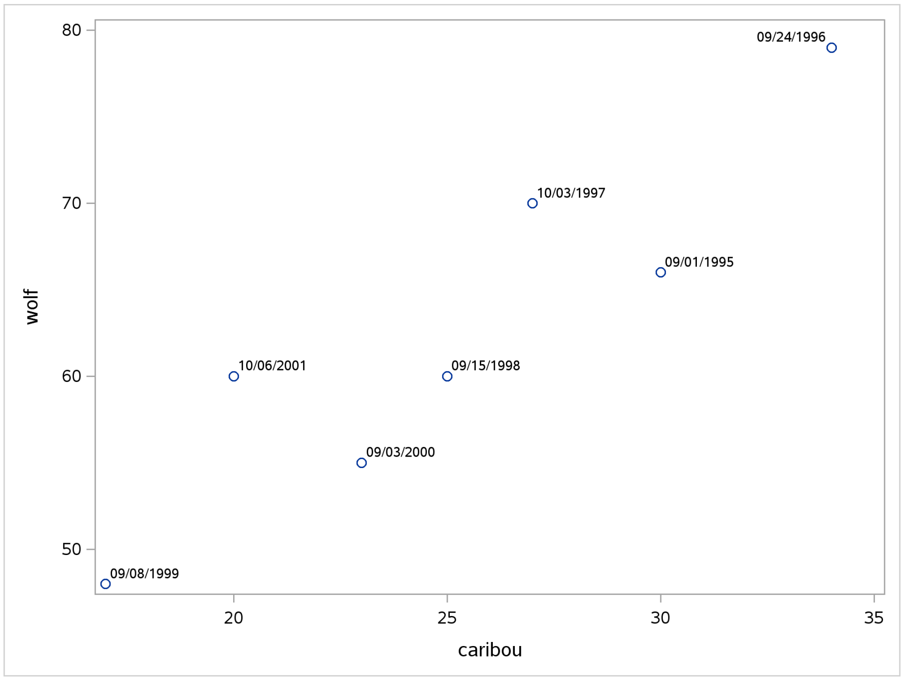
or hardly varies at all:

```
proc sgplot;
  y2axis values=(-400 to 500 by 100);
  series x=date y=caribou;
  series x=date y=wolf / y2axis;
```



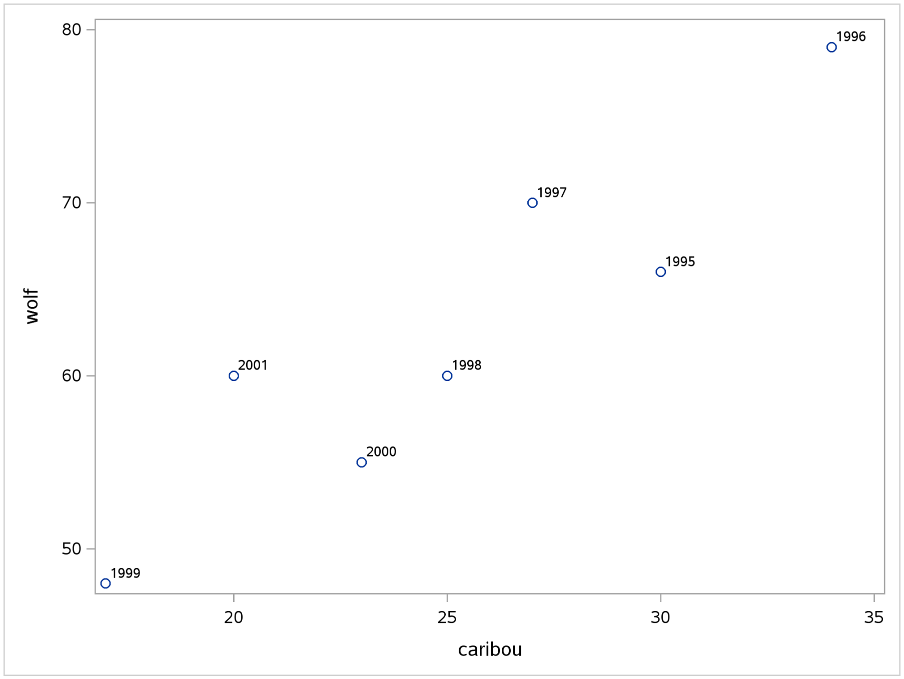
In my opinion, too many people just plot series against time, possibly with a second y-axis. Variables that vary together, like the wolf and caribou populations here, ought to be plotted *against each other* on a scatterplot, possibly with the time points labelled:

```
proc sgplot;
  scatter x=caribou y=wolf / datalabel=date;
```



or, better, with just the years, which we obtain first:

```
data denali2;
  set denali;
  year=year(date);

proc sgplot;
  scatter x=caribou y=wolf / datalabel=year;
```



1996 was the high year for the populations, followed by a precipitous decline, and by 2000 or 2001 we would guess that the populations were beginning to increase again.

The ambitious among you may like to join the points by arrows, in time order. The further ambitious may like to compare the graphs here with other predator-prey relationships.

(i) Back in part (f), you drew a scatterplot of wolf population against caribou population. How does any trend there show up in the time plot you just drew?

Solution: Back in (f), we found an upward trend with the two populations: they tended to be large or small together. That should show up here as this: in a year where caribou population is large, wolf population is also large. In the fall 1996 survey, both populations were at their biggest, and in the fall 1999 survey, both populations were smallest. Also, the shapes of the time trends are very similar. So the plot of (f) and this one are telling the same story, with this plot giving the additional information that both populations (over this time span) are decreasing over time.

13.4. The Worcester Heart Attack Study is an ongoing study of heart attacks in the Worcester, MA area. The main purpose of the study is to investigate changes over time in incidence and death rates, and also the use of different treatment approaches. We will be mainly using this data set to investigate data handling and dealing with dates. The data can be found at `http://www.utsc.utoronto.ca/~butler/c32/whas500.txt`.

(a) Read the data into SAS. Display the first five rows of your dataset as read in, with the dates shown as year-month-day.

Solution:

Much the usual kind of thing:

```
filename myurl url 'http://www.utsc.utoronto.ca/~butler/c32/whas500.txt';

proc import
  datafile=myurl
  out=whas
  dbms=dlm
  replace;
  getnames=yes;
  delimiter=" ";

proc print data=whas(obs=5);
  format admitdate--fdate yymmdd10.;
```

This is actually my second attempt. My first attempt skipped the `format` while I looked at what variables I had. I saw that the columns `admitdate` through `fdate` were dates, and they were consecutive columns, so that the `format` as shown would work.[56]

I said only 5 rows because there are a lot of variables:

```
Obs          id           age        gender            hr         sysbp

  1           1            83             0            89           152
  2           2            49             0            84           120
  3           3            70             1            83           147
  4           4            70             0            65           123
  5           5            70             0            63           135

Obs      diasbp           bmi           cvd           afb           sho

  1          78      25.54051             1             1             0
  2          60      24.02398             1             0             0
  3          88       22.1429             0             0             0
  4          76      26.63187             1             0             0
  5          85      24.41255             1             0             0

Obs         chf           av3        miord       mitype         year  admitdate

  1           0             0            1            0            1 1997-01-13
  2           0             0            0            1            1 1997-01-19
  3           0             0            0            1            1 1997-01-01
  4           1             0            0            1            1 1997-02-17
  5           0             0            0            1            1 1997-03-01

Obs    disdate       fdate          los         dstat       lenfol        fstat

  1 1997-01-18 2002-12-31            5             0         2178            0
  2 1997-01-24 2002-12-31            5             0         2172            0
  3 1997-01-06 2002-12-31            5             0         2190            0
  4 1997-02-27 1997-12-11           10             0          297            1
  5 1997-03-07 2002-12-31            6             0         2131            0
```

This looks like success. The clue that it worked is that `lenfol` in the original data file was usually a big number and `fstat`, on the end of the line, was 0 or 1, and so they appear here. `admitdate`, `disdate` and `fdate` are all properly formatted with the year first.

(b) The variables `los` and `lenfol` are numbers of days. Create a new data set that contains the differences between each of your dates, and see if you can work out what `los` and `lenfol` actually are. (When you display the results, display only the columns you care about and only enough rows to convince yourself that it worked.)

Solution: The strategy is to make a copy of the data set you read in from the file, create your new variables and then either (i) keep only the variables you want and print out the whole resulting data set or (ii) `print` only the variables you care about. I think 20 rows is enough:

```
data whas2;
  set whas;
  diff1=disdate-admitdate;
  diff2=fdate-admitdate;
  diff3=fdate-disdate;
  keep diff1 diff2 diff3 los lenfol;


proc print data=whas2(obs=20);
```

```
         Obs              los           lenfol     diff1     diff2     diff3

           1                5             2178         5      2178      2173
           2                5             2172         5      2172      2167
           3                5             2190         5      2190      2185
           4               10              297        10       297       287
           5                6             2131         6      2131      2125
           6                1                1         1         1         0
           7                5             2122         5      2122      2117
           8                4             1496         4      1496      1492
           9                4              920         4       920       916
          10                5             2175         5      2175      2170
          11                5             2173         5      2173      2168
          12               10             1671        10      1671      1661
          13                7             2192         7      2192      2185
          14               21              865        21       865       844
          15                4             2166         4      2166      2162
          16                1             2168         1      2168      2167
          17               13              905        13       905       892
          18               14             2353        14      2353      2339
          19                6             2146         6      2146      2140
          20               17               61        17        61        44
```

Alternatively, less work on the `data` step and more on the `proc print`:

```
data whas3;
  set whas;
  diff1=disdate-admitdate;
  diff2=fdate-admitdate;
  diff3=fdate-disdate;


proc print data=whas3(obs=20);
  var diff1 diff2 diff3 los lenfol;
```

```
         Obs    diff1    diff2    diff3            los          lenfol

           1       5      2178     2173              5            2178
           2       5      2172     2167              5            2172
           3       5      2190     2185              5            2190
           4      10       297      287             10             297
           5       6      2131     2125              6            2131
           6       1         1        0              1               1
           7       5      2122     2117              5            2122
           8       4      1496     1492              4            1496
           9       4       920      916              4             920
          10       5      2175     2170              5            2175
          11       5      2173     2168              5            2173
          12      10      1671     1661             10            1671
          13       7      2192     2185              7            2192
          14      21       865      844             21             865
          15       4      2166     2162              4            2166
          16       1      2168     2167              1            2168
          17      13       905      892             13             905
          18      14      2353     2339             14            2353
          19       6      2146     2140              6            2146
          20      17        61       44             17              61
```

My `diff1` is the same as `los`. I calculated `diff1` as `disdate` minus `admitdate`. It doesn't take much imagination to realize that these are the dates each patient was admitted to and discharged from the hospital (but the other way around). This is the number of days each patient stayed in the hospital: that is, `los` is the "length of stay" in hospital.

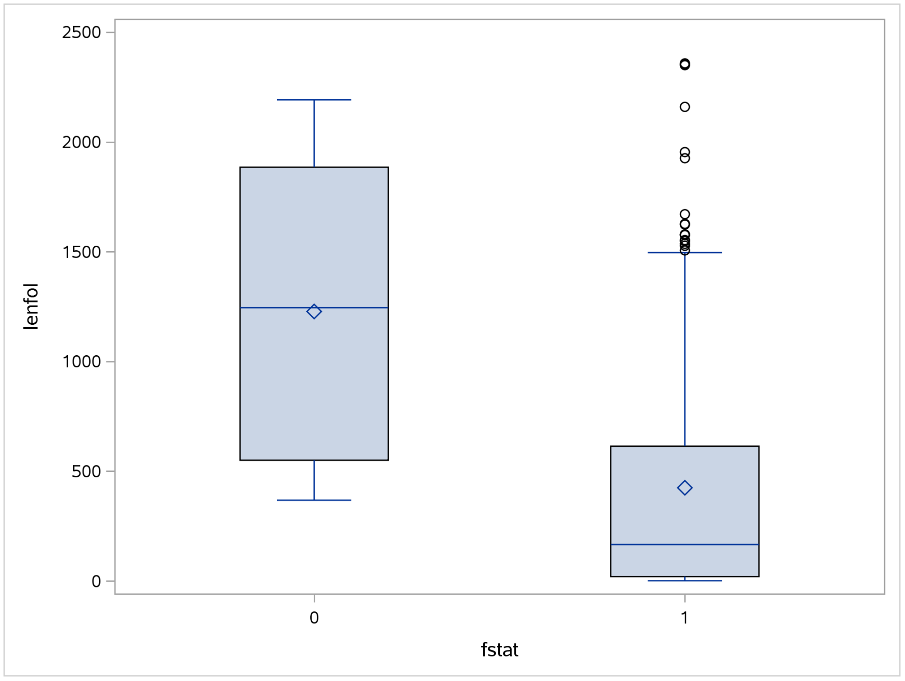
`lenfol` is the same as my `diff2`, which is the time from being admitted to hospital to `fdate`, which is the latest of any of the dates. What usually happens in a study like this is that the doctor[57] checks in with each patient every so often to see how they are doing, and thus `fdate` is the "latest followup date". This will be either the end of the study, or the date at which the patient was noted to have died. That is, `lenfol` is the "length of followup", from the first time the patient was seen to the last.

(c) What do you think `fstat` represents? To help you guess, make side-by-side boxplots of `lenfol` for each value of `fstat`. (That will mean going back to the dataset you read in from the file.)

Solution: The dataset I read in was called `whas`, so I need to specify that on the `proc sgplot`:

```
proc sgplot data=whas;
```

```
        vbox lenfol / category=fstat;
```



2500
2000
1500
1000
500
0
lenfol
0
1
fstat

What's showing here? When `fstat` is 0, `lenfol` has a nice symmetric distribution, with a maximum around 2200 (days) and no outliers. But when `fstat` is 1, `lenfol` has a very right-skewed distribution with a lot of outliers at the upper end. For these patients, the length of followup is usually short, with a few patients having longer followup. The likely meaning of this is that patients with `fstat` equal to 1 are the ones that died (often fairly quickly), while the patients with `fstat` of zero were the ones that were still alive at the end of the study.

Patients continued to be enrolled into the study at various different times, so the time between enrolment and last followup could be quite variable. If I was right about these patients being followed until the study ended, though, the time of last followup should be consistent for all of them:

```
proc means min q1 median q3 max;
  where fstat=0;
  var fdate;
```

```
                              The MEANS Procedure

                           Analysis Variable : fdate

                      Lower                                 Upper
      Minimum       Quartile          Median          Quartile          Maximum
   --------------------------------------------------------------------------------
     15705.00       15705.00        15705.00        15705.00         15705.00
   --------------------------------------------------------------------------------
```

For the patients still alive at the end of the study, the last followup of *all* of them was on the same date. I could even work out what date that was, something like this:

```
proc sort out=sorted;
  where fstat=0;
  by descending fdate;

proc print data=sorted(obs=10);
  var fdate;
```

```
                               Obs         fdate

                                 1    31/12/2002
                                 2    31/12/2002
                                 3    31/12/2002
                                 4    31/12/2002
                                 5    31/12/2002
                                 6    31/12/2002
                                 7    31/12/2002
                                 8    31/12/2002
                                 9    31/12/2002
                                10    31/12/2002
```

The last day of 2002.

It occurs to me that if you made your data set with the time differences by keeping all of the original variables, (and then sending only some of them to `proc print`) that data set will be your most recent one, and you'll be able to run the `proc`s above without specifying a data set, as I did on `proc means` just above.

The place we would go next in terms of analysis would be to think about whether survival time depends on treatment, after allowing for the effects of any of the other variables. This sounds like a regression. What confuses things here is that for the patients who "haven't died yet", we don't know how long they are going to live: all we know is they have lived 2200 days (or whatever) since being admitted to hospital and are still alive the last we heard. These patients are known in the jargon as "censored" (or, I suppose, their survival time is what's "censored" since it hasn't been observed yet). Also, time until death has a lower limit of 0 and (in principle) no upper limit, so it will have a right-skewed distribution, rather than the normal that we would like for a regression. With all this in mind, we would tend to reach for a "survival analysis", often Cox's Proportional Hazards model,[58] which you'll see if you take STAD29.

13.5. In 2010, a group of students planted some Mizuna lettuce seeds, and recorded how they grew. The data were saved in an Excel spreadsheet, which is at `http://www.utsc.utoronto.ca/~butler/c32/mizuna.xlsx`. The columns are: the date, the height (in cm) of (I presume) the tallest plant, the amount of water added since the previous date (ml), the temperature in the container where the seedlings were growing, and any additional notes that the students made (edited for length by me). The top line of the data file is variable names.

(a) Read the spreadsheet directly into SAS using `proc import`. That will mean (i) saving the spreadsheet somewhere on your computer (or finding it in your Downloads folder), (ii) uploading it to SAS Studio, (iii) getting the `proc import` right. *I do not want you doing any copying and pasting here.*

Display the whole of your data set (it is not very big).

(When I did this, the year came out wrong. If that happens to you, ignore it.)

Solution: Once the spreadsheet is in the right place, you'll need code like this:

```
proc import
  datafile='/home/ken/mizuna.xlsx'
  out=mizuna
  dbms=xlsx
  replace;
  getnames=yes;
```

There is only one sheet in the workbook, so you don't have to name it.

Did it work?

```
proc print;
```

```
Obs    date     height    water    temperature

  1    40225       0       400          21
  2    40227       0         0        22.5
  3    40228       0       200        20.9
  4    40231     3.2       100        20.8
  5    40232     4.5       100        22.9
  6    40234       6       100        21.8
  7    40235     6.5       200        21.2
  8    40238     9.5       200        21.8
  9    40240    11.1       200        21.7
 10    40242      13       250        21.9
 11    40245    14.5       500        22.5
 12    40247      16       200        21.2
 13    40254    18.5       800        20.8

Obs    notes

  1    planted seeds; water soaked up rapidly
  2    2 of 6 seeds not appeared; soil still moist
  3    4 of 6 plants broken surface
  4    Last seed hasnt broken surface
  5    Plants growing well.
  6    Last seed sprouted; plants looking green and healthy
  7
  8
  9    Plants needing more water
 10
 11    No water left, leaves droopy. Added water, came back to life
 12    Plants green and healthy
 13    Harvest. Tips of plants turning brown.
```

Well, almost everything worked: except for the dates, which came out as random-looking integers. Usually, you can go looking in the Log tab to see what formats `proc import` read and displayed the data as, but that seems not to be working for me with Excel spreadsheets. So let's re-do our `proc print` with the dates formatted in some friendly way like this:

```
proc print;
  format date yymmdd10.;
```

```
Obs          date    height    water    temperature

  1    2070-02-17       0        400          21
  2    2070-02-19       0          0        22.5
  3    2070-02-20       0        200        20.9
  4    2070-02-23     3.2        100        20.8
  5    2070-02-24     4.5        100        22.9
  6    2070-02-26       6        100        21.8
  7    2070-02-27     6.5        200        21.2
  8    2070-03-02     9.5        200        21.8
  9    2070-03-04    11.1        200        21.7
 10    2070-03-06      13        250        21.9
 11    2070-03-09    14.5        500        22.5
 12    2070-03-11      16        200        21.2
 13    2070-03-18    18.5        800        20.8

Obs    notes

  1    planted seeds; water soaked up rapidly
  2    2 of 6 seeds not appeared; soil still moist
  3    4 of 6 plants broken surface
  4    Last seed hasnt broken surface
  5    Plants growing well.
  6    Last seed sprouted; plants looking green and healthy
  7
  8
  9    Plants needing more water
 10
 11    No water left, leaves droopy. Added water, came back to life
 12    Plants green and healthy
 13    Harvest. Tips of plants turning brown.
```

OK, now we have dates, but they are *the wrong year!* They should be 2010, not 2070.

(If it worked out right for you, let me know; this might be one of those operating-system things.) I'm going to ignore the fact that the year is wrong, since the month and day is correct.

What actually happened, I think, is that the date as-a-number got read in wrong:

```
proc print;
  var date height water;
```

```
Obs    date     height    water

  1    40225       0       400
  2    40227       0         0
  3    40228       0       200
  4    40231     3.2       100
  5    40232     4.5       100
  6    40234       6       100
  7    40235     6.5       200
  8    40238     9.5       200
  9    40240    11.1       200
 10    40242      13       250
 11    40245    14.5       500
 12    40247      16       200
 13    40254    18.5       800
```

How many years after 1960[59] are those? The numbers are days, so divide by days in a year:

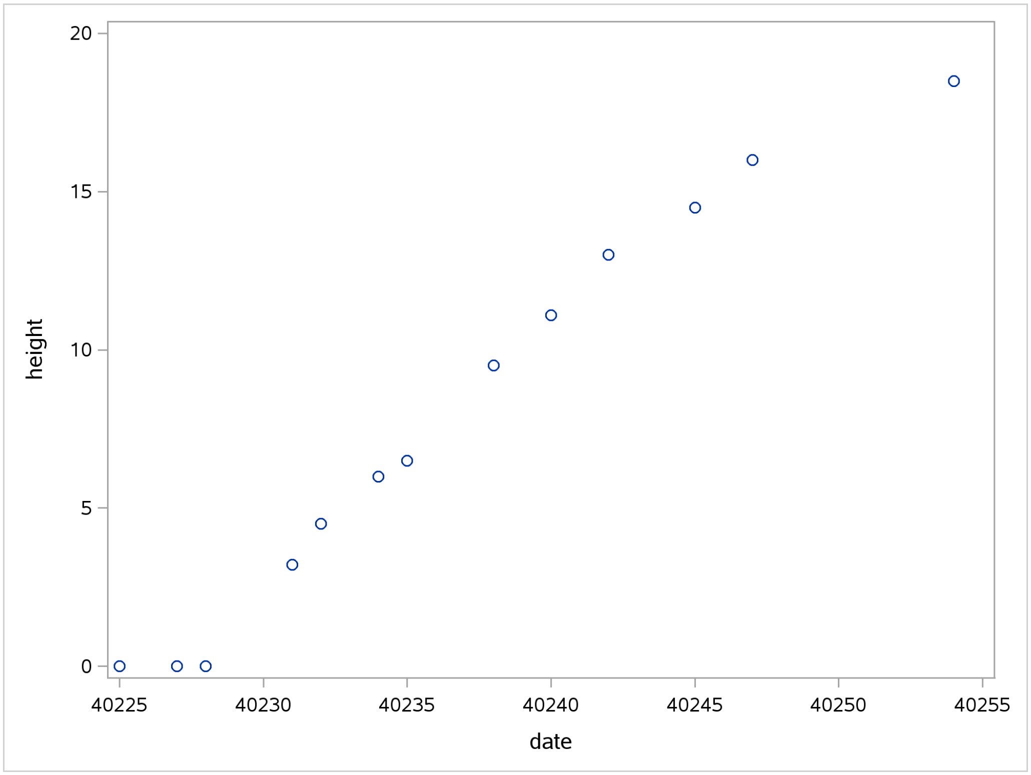
```
40235/365.25

## [1] 110.1574
```

110 years after 1960, so 2070 is right. Something went wrong between my entering the numbers in my spreadsheet and them being read into SAS. SAS did the right conversion to go from days-since-1960 to dates, but got the wrong days-since-1960.

(b) Make a suitable plot that shows how the lettuce seeds grow over time.

Solution: My first thought is a scatterplot of height against date:

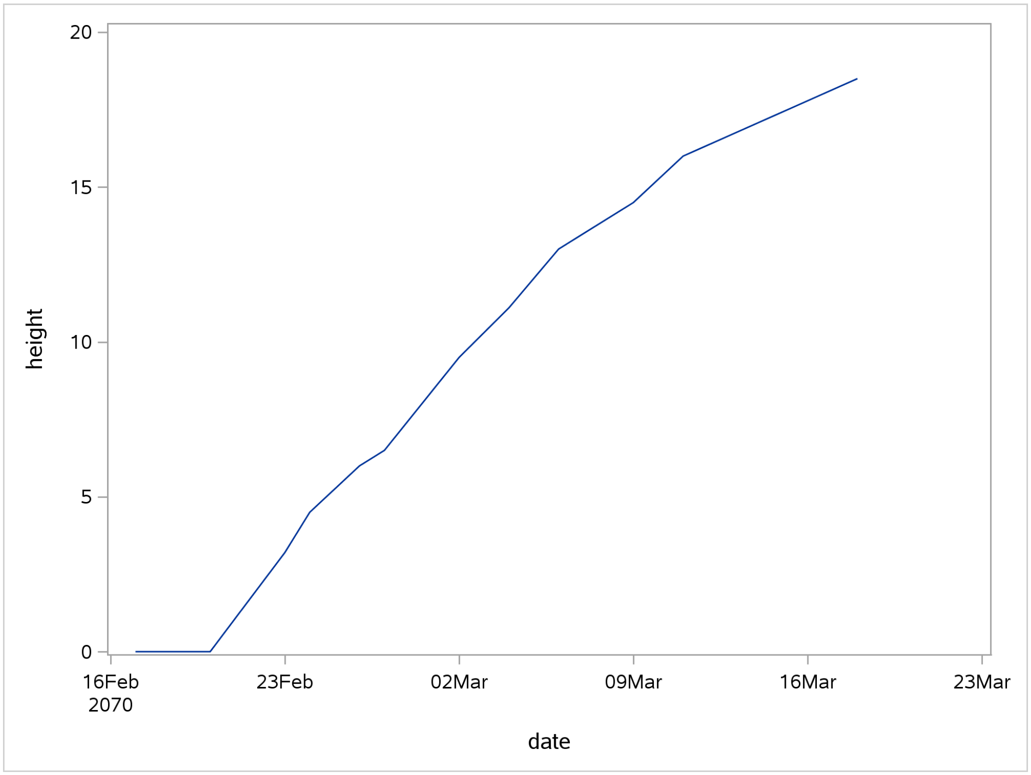
```
proc sgplot;
  scatter x=date y=height;
```



height
date
0
5
10
15
20
40225
40230
40235
40240
40245
40250
40255

There are a couple of things I would fix here. One, the date is not shown properly (it is days-since-1960). Two, the plants grow continuously over time, so I would join the points by lines as `series` does. This leads to:

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



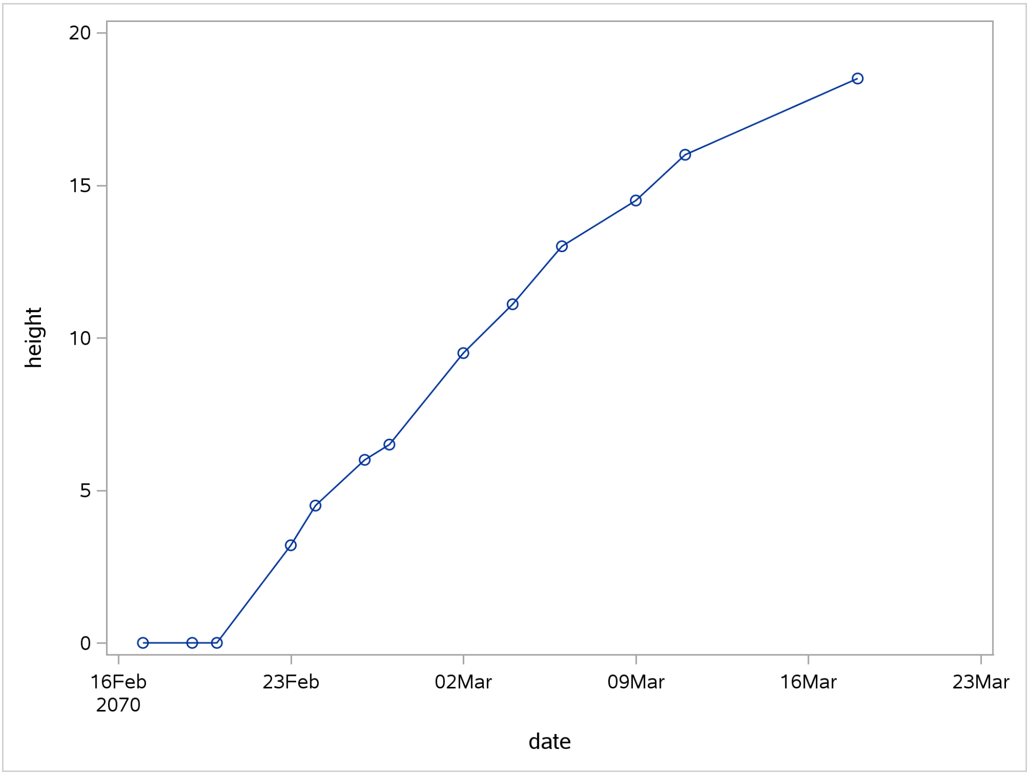
height
date
0
5
10
15
20
16Feb 2070
23Feb
02Mar
09Mar
16Mar
23Mar

Apart from the fact that the year is wrong, everything looks good, and you see that the growth of the plants slows down slightly towards the end (and that there is a delay of a few days at the beginning before any of the seedlings pop above the soil and there is any height to measure).

Note that `proc sgplot` can take a `format` just as `proc print` can.

Here's how to show the data points as well as the lines joining them:

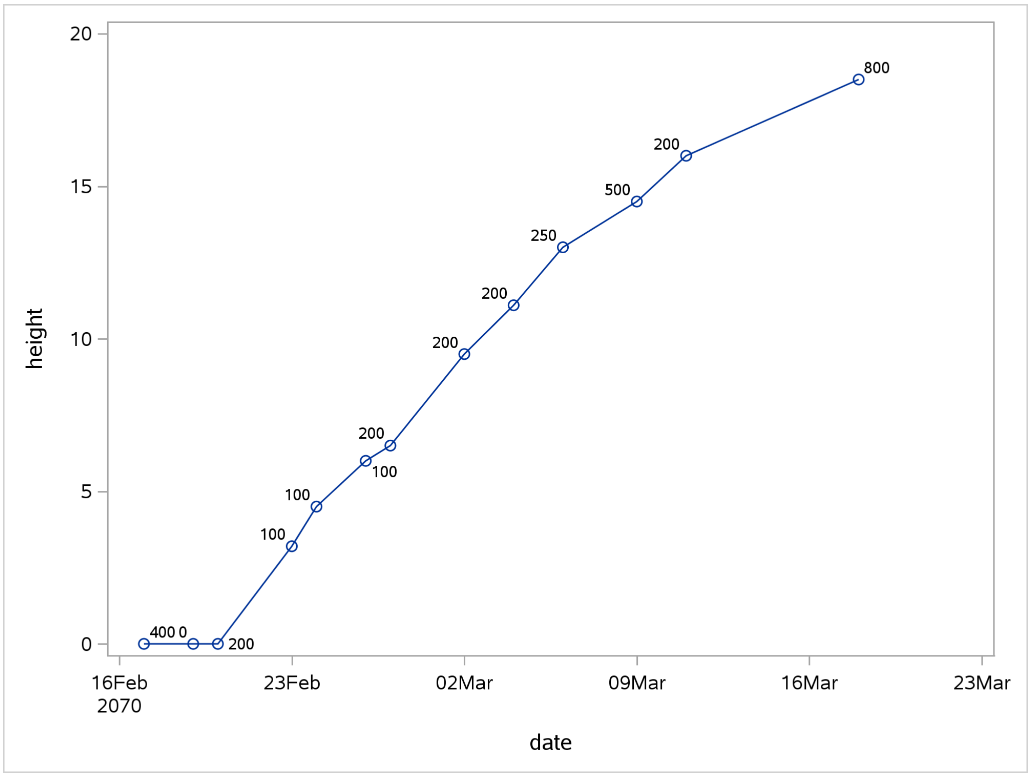
```
proc sgplot;
  series x=date y=height / markers;
  format date yymmdd10.;
```



(c) Label each point with the amount of water given to the plants between the previous time point and this one.

Solution: I had to try to find the words that would lead you towards using `water` rather than making it more complicated than it is. The magic word is `datalabel`. I've left the `markers` in, because it's easier to judge what the labels refer to when you can actually *see* the points. (Try it without the `markers`: is it clear or confusing?)

```
proc sgplot;
  series x=date y=height / markers datalabel=water;
  format date yymmdd10.;
```



height
date
0
5
10
15
20
16Feb 2070
23Feb
02Mar
09Mar
16Mar
23Mar
400
0
200
100
100
100
200
200
200
250
500
200
800

SAS chooses the orientation of the labels so that you can best see them (unlike R, which needs to be told where to put the labels, unless you use something like `ggrepel`).

It occurs to me that the amount of water is over a differing number of days, and that what really matters is how much water was supplied *per day* over the time period in question. That means counting the number of days between each date and the previous one, and working out the water per day. This is what I came up with (making a new data set, since we are making a new variable or two):

```
data miz2;
  set mizuna;
  drop notes;
  datediff=dif(date);
  waterperday=water/datediff;


proc print;
  format date date9.;
proc sgplot;
  series x=date y=height / markers datalabel=waterperday;
  format date yymmdd10.;
```

The new data set is below. I got rid of `notes`; `dif` calculates the difference between each value and the previous one. In this case, you can see that it figured out the number of days between the date on that line and the previous one. (Since the dates are stored as days internally, subtracting them will give a number of days, which is what we want.)

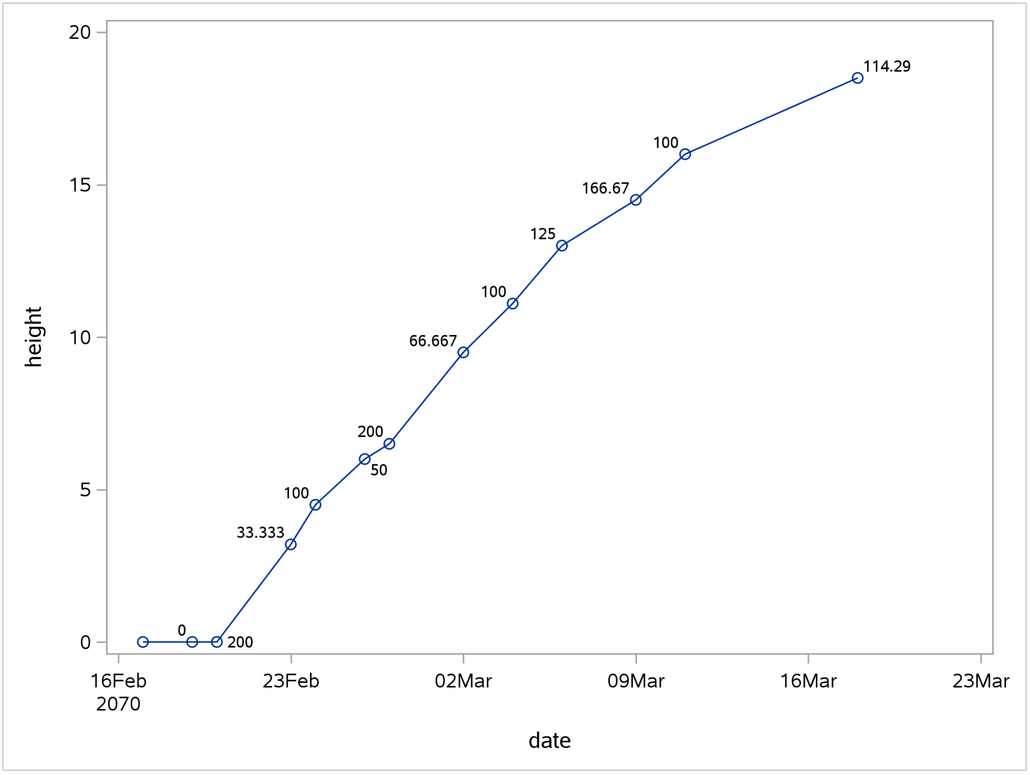
I used a different `format` for the dates, just for fun.

The first date doesn't have a number of days since the previous one (because there *isn't* a previous one), so the first `datediff`, and hence the first `waterperday`, are both missing.

```
Obs         date    height    water    temperature    datediff    waterperday

  1    17FEB2070       0       400          21            .             .
  2    19FEB2070       0         0        22.5            2          0.000
  3    20FEB2070       0       200        20.9            1        200.000
  4    23FEB2070     3.2       100        20.8            3         33.333
  5    24FEB2070     4.5       100        22.9            1        100.000
  6    26FEB2070       6       100        21.8            2         50.000
  7    27FEB2070     6.5       200        21.2            1        200.000
  8    02MAR2070     9.5       200        21.8            3         66.667
  9    04MAR2070    11.1       200        21.7            2        100.000
 10    06MAR2070      13       250        21.9            2        125.000
 11    09MAR2070    14.5       500        22.5            3        166.667
 12    11MAR2070      16       200        21.2            2        100.000
 13    18MAR2070    18.5       800        20.8            7        114.286
```

You can check that the other `waterperday` values make sense.

The new plot looks like this, with the points labelled by water per day (since the last time the plants were measured):



I don't really see any patterns here. I think they watered the plants when the soil seemed to be getting dry, or perhaps on a Friday before they left for the weekend. Note that SAS has rounded off the water-per-day values.

13.6. Childbirths can be of two types: a "vaginal" birth in which the child is born through the mother's vagina in the normal fashion, and a "cesarean section" where a surgeon cuts through the wall of the mother's abdomen, and the baby is delivered through the incision. Cesarean births are used when there are difficulties in pregnancy or during childbirth that would make a vaginal birth too risky.

A hospital kept track of the number of vaginal and Cesarean births for the twelve months of 2012. Of interest is whether the Cesarean rate (the ratio of Cesarean births to all births) was increasing, decreasing or remaining stable over that time.

The data may be found at `http://www.utsc.utoronto.ca/~butler/c32/birthtypes.txt`. The columns are the names of the months (in 2012), the number of cesarean births and the number of vaginal births. (The data are not real, but are typical of the kind of thing you would observe.)

We did this in R before.

(a) Read the same data into SAS and display all 12 rows.

Solution: The usual stuff first:

```
filename myurl url 'http://www.utsc.utoronto.ca/~butler/c32/birthtypes.txt';

proc import
  datafile=myurl
  out=births
  dbms=dlm
  replace;
  getnames=yes;
  delimiter=' ';

proc print;
```

```
                    Obs    month       cesarean        vaginal

                      1     Jan           11              68
                      2     Feb            9              63
                      3     Mar           10              72
                      4     Apr           18             105
                      5     May           10              90
                      6     Jun           10              92
                      7     Jul           11              78
                      8     Aug            9              83
                      9     Sep            9              90
                     10     Oct           15             101
                     11     Nov           12             130
                     12     Dec            8             101
```

Next, we'll fix this up to make actual dates.

(b) Those "months" are really just the names of the months as text. Construct a new data set that makes real dates out of the month names (and a supplied year and day-of-month), and display it in such a way as to show that you now have dates. You might need to use `cat` and `input` in your data step (which in turn would mean finding out what they do). You might it convenient to note that a date in the form 01Jan1972 is what SAS calls `date9.`, with a dot on the end.

Solution: There are two steps: one, to make a piece of text that looks like a `date9.` date, which is what `cat` does, and two, to persuade SAS to treat this as a date, which is what `input` does.

cat works like paste in R (or, more precisely, str_c or paste0, since it glues the text together without any intervening spaces). It takes any number of inputs, which can be literal text or variables. In our case, it is the text 01 followed by the month name followed by the text 2012, to make a date in date9. format.

input takes a variable and a format (precisely, an "informat") and expresses the value of the variable as input in the appropriate format. In this case, it converts the text-that-looks-like-a-date (in thedate below) into a real date (in realdate).

Then, to display the real dates so that they look like dates, you run proc print with a format. You can use any date format you like; I chose one that looks different from the dates-as-text in thedate. If you forget the format, the real dates will be output as days since Jan 1, 1960, which might convince you that they are real dates, but not *which* real dates they are.

After all that preamble, the code, followed by the output that it produces:

```
data births2;
  set births;
  thedate=cat('01',month,'2012');
  realdate=input(thedate,date9.);

proc print;
  format realdate yymmdd10.;
```

```
     Obs    month        cesarean         vaginal     thedate       realdate

       1     Jan               11              68    01Jan2012    2012-01-01
       2     Feb                9              63    01Feb2012    2012-02-01
       3     Mar               10              72    01Mar2012    2012-03-01
       4     Apr               18             105    01Apr2012    2012-04-01
       5     May               10              90    01May2012    2012-05-01
       6     Jun               10              92    01Jun2012    2012-06-01
       7     Jul               11              78    01Jul2012    2012-07-01
       8     Aug                9              83    01Aug2012    2012-08-01
       9     Sep                9              90    01Sep2012    2012-09-01
      10     Oct               15             101    01Oct2012    2012-10-01
      11     Nov               12             130    01Nov2012    2012-11-01
      12     Dec                8             101    01Dec2012    2012-12-01
```

The dates in realdate are the same dates as in thedate, but written a different way. The fact that SAS re-formats the dates this way *and gets them right* is evidence that it is handling the dates properly.

The mdy idea from the lecture notes doesn't work, because our months are *names* and they need to be *numbers* for mdy to work. It seems to be necessary to construct a piece of text that looks like a date and then use input to convert it into an actual date.

If you get stuck, go back and edit (a copy of) the data file to include the years and month days and then read that in. This will enable you to do the remaining parts, but don't expect to get more than one point for this part if that's what you do.

(c) With yet another new data set, create a variable containing the cesarian rates, as you did earlier.

Solution: This is just a matter of keeping your head after the craziness of the previous part. There is no craziness here:

```
data births3;
  set births2;
  ces_rate=cesarean/(cesarean+vaginal);

proc print;
  format realdate ddmmyy8.;
```

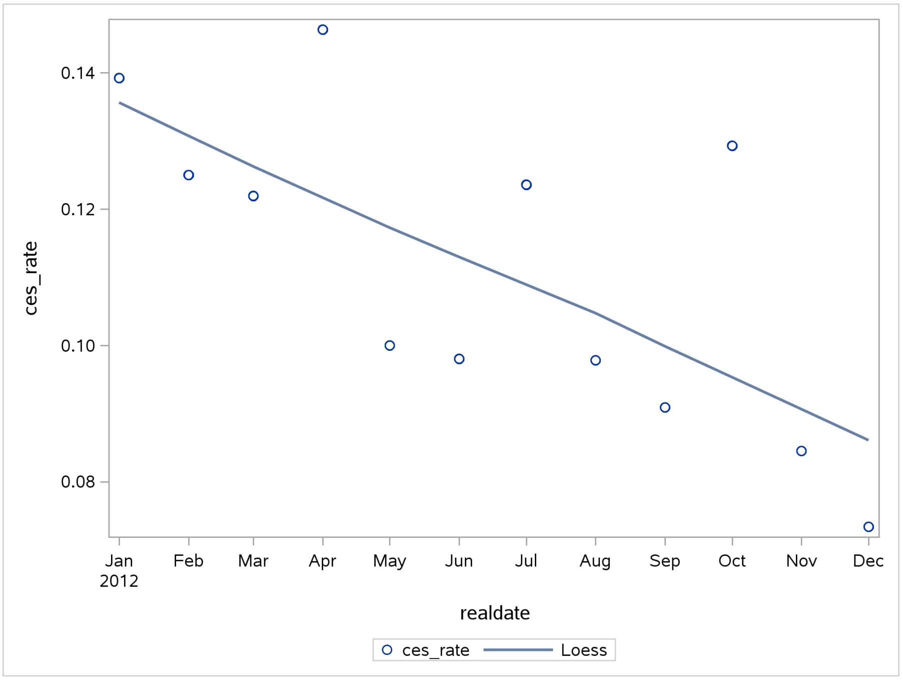
```
 Obs   month       cesarean        vaginal    thedate    realdate   ces_rate

   1    Jan              11             68   01Jan2012   01/01/12    0.13924
   2    Feb               9             63   01Feb2012   01/02/12    0.12500
   3    Mar              10             72   01Mar2012   01/03/12    0.12195
   4    Apr              18            105   01Apr2012   01/04/12    0.14634
   5    May              10             90   01May2012   01/05/12    0.10000
   6    Jun              10             92   01Jun2012   01/06/12    0.09804
   7    Jul              11             78   01Jul2012   01/07/12    0.12360
   8    Aug               9             83   01Aug2012   01/08/12    0.09783
   9    Sep               9             90   01Sep2012   01/09/12    0.09091
  10    Oct              15            101   01Oct2012   01/10/12    0.12931
  11    Nov              12            130   01Nov2012   01/11/12    0.08451
  12    Dec               8            101   01Dec2012   01/12/12    0.07339
```

Well, apart from the craziness I introduced with the display of the dates this time, at any rate. Again, you can use any date format you like for these. The point of this part is to get the cesarean rates right.

(d) Make a scatterplot of cesarean rates against time. Add a smooth trend.

Solution: This should look about the same as the R one. You need to add a `format` to the x-axis, or else it will come out as seconds since Jan 1, 1960. Having one `format` seems to be enough, though there's no harm in having a format after `scatter` and one after `loess` as well. (The `format` actually belongs to `proc sgplot`, so one format applies to the whole plot, no matter how many points, lines or curves are on it.)

```
proc sgplot;
  scatter x=realdate y=ces_rate;
  loess x=realdate y=ces_rate;
  format realdate date9.;
```



Any date format will do. (If you use a format like `yymmdd10.` that displays the month as a number, you might get numbers on the x-axis. I'm OK with that.)

This trend is a rather more evidently downward one than R's, probably because SAS does a bit more smoothing than R does. The amount of smoothing done by `geom_smooth` is controlled by something called `span`,[60] which has a default value of 0.75, so making `span` a bit bigger smooths out the smooth trend:

```
ggplot(b2,aes(x=thedate,y=cesarean_rate))+geom_point()+
  geom_smooth(span=1)

## Error in ggplot(b2, aes(x = thedate, y = cesarean_rate)):  object 'b2' not found
```

This still has a bit more of a wiggle than SAS's picture, but it tells the same story. Making `span` smaller would make the trend more wiggly, and in my opinion makes it over-react to where data values happen to be.

(e) Use `proc corr` to find the Kendall correlation with time (the "Mann-Kendall correlation").[61] This will involve digging into the help for `proc corr`. You can find this by entering `proc corr` into your favourite search engine, and looking at the results that start with `support.sas.com`. The output should also give you a P-value.

Solution: A small amount of digging reveals two things:

1. to get the Kendall correlation (officially called "Kendall's tau-b"), you add `kendall` to the `proc corr` line.
2. to specify the variables to calculate correlations between, put them on a `var` line, as you would for `proc means`. (If you don't, you get correlations for all the pairs of variables.)

Thus:

```
proc corr kendall;
  var realdate ces_rate;
```

```
                                The CORR Procedure

                         2  Variables:    realdate ces_rate
                                Simple Statistics

Variable         N        Mean     Std Dev      Median     Minimum     Maximum

realdate        12       19160   109.93249       19160       18993       19328
ces_rate        12     0.11084     0.02304     0.11098     0.07339     0.14634
                 Kendall Tau b Correlation Coefficients, N = 12
                          Prob > |tau| under H0: Tau=0

                                realdate      ces_rate

                  realdate       1.00000      -0.60606
                                                0.0061

                  ces_rate      -0.60606       1.00000
                                  0.0061
```

The correlation with time is -0.606 (with a P-value of 0.0061 that we use below).

Extra stuff: if you try to calculate the Kendall correlation with time in R, it doesn't work. First we have to read in the data, and then handle the dates:

```
library(tidyverse)
library(lubridate)

##
## Attaching package:  'lubridate'

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

then

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/birthtypes.txt"
births=read_delim(my_url, " ")

## Parsed with column specification:
## cols(
##   month = col_character(),
##   cesarean = col_double(),
##   vaginal = col_double()
## )

births

## # A tibble: 12 x 3
##    month cesarean vaginal
##    <chr>    <dbl>   <dbl>
##  1 Jan         11      68
##  2 Feb          9      63
##  3 Mar         10      72
##  4 Apr         18     105
##  5 May         10      90
##  6 Jun         10      92
##  7 Jul         11      78
##  8 Aug          9      83
##  9 Sep          9      90
## 10 Oct         15     101
## 11 Nov         12     130
## 12 Dec          8     101

b2 = births %>% mutate(datestr=str_c("2012",month,"1",sep=" ")) %>%
  mutate(thedate=ymd(datestr)) %>%
  mutate(cesarean_rate=cesarean/(cesarean+vaginal))
b2

## # A tibble: 12 x 6
##    month cesarean vaginal datestr    thedate    cesarean_rate
##    <chr>    <dbl>   <dbl> <chr>      <date>             <dbl>
##  1 Jan         11      68 2012 Jan 1 2012-01-01        0.139
##  2 Feb          9      63 2012 Feb 1 2012-02-01        0.125
##  3 Mar         10      72 2012 Mar 1 2012-03-01        0.122
##  4 Apr         18     105 2012 Apr 1 2012-04-01        0.146
##  5 May         10      90 2012 May 1 2012-05-01        0.1
##  6 Jun         10      92 2012 Jun 1 2012-06-01        0.0980
##  7 Jul         11      78 2012 Jul 1 2012-07-01        0.124
##  8 Aug          9      83 2012 Aug 1 2012-08-01        0.0978
##  9 Sep          9      90 2012 Sep 1 2012-09-01        0.0909
## 10 Oct         15     101 2012 Oct 1 2012-10-01        0.129
## 11 Nov         12     130 2012 Nov 1 2012-11-01        0.0845
## 12 Dec          8     101 2012 Dec 1 2012-12-01        0.0734
```

and then:

```
with(b2,cor(thedate,cesarean_rate),method="kendall")

## Error in cor(thedate, cesarean_rate):  'x' must be numeric
```

Why did it work in SAS but not in R? The error message says that `x`, the first input to `cor`, was not numeric:

```
class(b2$thedate)

## [1] "Date"
```

and R won't even pretend it's a number. SAS, on the other hand, stores dates as numbers (days since Jan 1, 1960), and the number is rather more visible: you have to do the `format` thing to even make them display as dates, otherwise you get that number. Thus, in SAS, when you calculate a correlation with a date (or a time), SAS will use the underlying number, and will get you a correlation with time. (A higher number means a later date or time, so it is doing what you want.)

It is a matter of opinion whether you think of dates as being "ordinal" or "interval" — for example, do you think it makes sense to calculate a *mean* date? — but the Kendall correlation only uses the order of the dates, not how far apart they are, so it is good if you think dates are only ordinal, while the regular Pearson correlation is not.

With R, you have to literally make the date into a number:

```
nd=as.numeric(b2$thedate)
nd

##  [1] 15340 15371 15400 15431 15461 15492 15522 15553 15584 15614 15645
## [12] 15675
```

These are days since January 1, *1970.* This is the Unix "epoch date". Unix was developed in the early 1970s; the way time was measured on early Unix systems meant that you had to have a zero date as something in the recent past, and so it was first 1971 and then 1970.[62] R was first developed on Unix systems, so it inherited date-and-time handling from there, as well as other things like `ls` for "list my objects" and `rm` for "delete".

SAS began life in the late 1960s, before Unix even existed, and so its "zero" date was chosen independently to be Jan 1 1960.

Anyway, by expressing our variable `thedate` in R as a literal number, we can now calculate the Kendall correlation with time:

```
cor(nd,b2$cesarean_rate,method="kendall")

## [1] -0.6060606
```

which gives the same answer as SAS, and we can do the test this way:

```
cor.test(nd,b2$cesarean_rate,method="kendall",exact=F)

##
##  Kendall's rank correlation tau
##
## data:  nd and b2$cesarean_rate
## z = -2.7429, p-value = 0.00609
## alternative hypothesis: true tau is not equal to 0
## sample estimates:
##        tau
## -0.6060606
```

The reason for the `exact=F` is to get the same P-value as SAS. The Kendall correlation has an approximate normal distribution (an approximation that improves as the sample size increases: here we only have 12 pairs of observations, which is not exactly large). SAS's default is to use the normal approximation and R's default is to calculate the P-value exactly for small samples, but they are both tweakable. (The exact P-value, from R, is about 0.0054, so it doesn't make much difference here.)

(f) What do you conclude from the P-value you obtained in the last part? Explain briefly.

Solution: You can try to reason out what is being tested, or you can look in the documentation. The first example, at `http://support.sas.com/documentation/cdl/en/procstat/66703/HTML/default/viewer.htm#procstat_corr_examples01.htm`, is a good place to look. From there, the P-value is for a test of "is there a time trend", with a small P-value meaning that there is a trend. If you prefer to reason things out, this is just like testing for a regression slope: the null hypothesis is that the slope is zero, and therefore that there is no relationship, and the two-sided alternative is that the slope is not zero, ie. that there is either an upward or downward trend.

Thus, the P-value of 0.0061 here is small, and therefore we *reject* the null hypothesis (that says there is no trend) in favour of the alternative, and therefore conclude that there *is* a trend, or that the trend observed over time on the plot is "real" or "reproducible" or some word like that.

In practice, you would typically have a much longer time series of measurements than this, such as monthly measurements for several years. In looking at only one year, like we did here, we could get trapped by seasonal effects: for example, cesarean rates might always go down through the year and then jump up again in January. Looking at several years would enable us to disentangle seasonal effects that happen every year from long-term trends. (As an example of this, think of Toronto snowfall: there is almost always snow in the winter and there is never snow in the summer, a seasonal effect, but in assessing climate change, you want to think about long-term trends in snowfall, after allowing for which month you're looking at.)

13.7. Previously, we did some row and column selection with my cars data set, `http://www.utsc.utoronto.ca/~butler/c32/cars.csv`. This time, we're going to create a permanent data set out of these data, and then demonstrate that we can use it in a `proc` without reading it in again.

(a) Read the data in and create a permanent data set. To do this, modify your `proc import` from before along the lines of the lecture notes, replacing the `ken` that is there with your username.

Solution: This is the `proc import` from before:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/cars.csv";

proc import
  datafile=myurl
  dbms=csv
  out=cars
  replace;
  getnames=yes;
```

You need to do two things to this: first, you create a `libname` up above the `proc import` that says where you want this permanent data set created, and second, you put the `libname` before the data set name on the `out` line. If you do that, you'll get something like this:

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/cars.csv";

libname mylib V9 '/home/ken';

proc import
  datafile=myurl
  dbms=csv
  out=mylib.mycars
  replace;
  getnames=yes;
```

In place of the `mylib` you can use any name you like (the same in both places), and the `ken` at the end should be replaced by your username, but otherwise use that line as is.

I think the `V9` on the `libname` line has to be there because the format of the permanent dataset file changed with version 9 of SAS, and SAS didn't want to break any code that created permanent data sets from earlier versions of SAS. Backward compatibility, and all that.

To check whether it worked, look in your SAS file storage, on the left side of SAS Studio, under Files (Home). You should see a file with the improbable name of `cars.sas7bdat`. That's the permanent data set. Fortunately, we never have to use that crazy extension.

I actually have SAS running on my computer rather than SAS Studio (in any of its flavours), so that I can check for the existence of a file (in Linux)[63] like this:

```
ls -l /home/ken/mycars.*

## -rw-rw-r-- 1 ken ken 131072 Nov 13 17:49 /home/ken/mycars.sas7bdat
```

There it is, and just created (as I write this).

(b) Close down SAS, and start it up again. Find the mean gas mileage for cars with each number of cylinders, *without* reading the data in again (that is to say, without using `proc import` or trickery involving `data` steps).

Solution: *Start* with the `proc means`, and to that append a `data=` that tells SAS to use your permanent data set. This is done by putting the data set name in quotes and starting it with a `/home/username`, as if it were a file, but with *no extension.* For me, that goes like this:

```
proc means data='/home/ken/mycars';
  var MPG;
  class cylinders;
```

For you, replace `ken` with your username. Here's what I got; you should get the same thing:

```
                               The MEANS Procedure

                             Analysis Variable : MPG

                  N
     cylinders  Obs   N          Mean       Std Dev       Minimum       Maximum
   ----------------------------------------------------------------------------
             4   19  19    30.0210526     4.1824473    21.5000000    37.3000000

             5    1   1    20.3000000             .    20.3000000    20.3000000

             6   10  10    21.0800000     4.0775265    16.2000000    28.8000000

             8    8   8    17.4250000     1.1925363    15.5000000    19.2000000
   ----------------------------------------------------------------------------
```

And that works, using my permanent data set.

Another way to do it is via the `libname` thing, same as you did when you saved the permanent data set:

```
libname mylib V9 '/home/ken';

proc means data=mylib.mycars;
  var MPG;
  class cylinders;
```

```
                              The MEANS Procedure

                            Analysis Variable : MPG

                  N
    cylinders   Obs   N          Mean       Std Dev       Minimum       Maximum
  ------------------------------------------------------------------------------
            4    19  19    30.0210526     4.1824473    21.5000000    37.3000000

            5     1   1    20.3000000             .    20.3000000    20.3000000

            6    10  10    21.0800000     4.0775265    16.2000000    28.8000000

            8     8   8    17.4250000     1.1925363    15.5000000    19.2000000
  ------------------------------------------------------------------------------
```

Success. Either way is good.

If you're in a company that uses SAS, permanent data sets are a good way to share data, since the person receiving the data doesn't have to do anything to open it: they can just start running `proc`s right away.

Notes

[1]If you forget it, there's a way to get it back, but it's better not to have to go that way.

[2]This is the same screen as you bookmarked earlier. You can use your bookmark if you prefer, now or later.

[3]This is because you are competing against everyone else in the world for access to SAS's servers.

[4]Downloading as PDF looks nice, but you cannot easily add it to another document with your code and explanation.

[5]Which stands for "rich text format"

[6]My guess is more or less correct; the re-writing in C happened in 1985, according to Wikipedia.

[7]The moral of this story is that you also need to think about what you want your graph to tell you, when thinking about what variables to include on it.

[8]Before about 1997, there was no interleague play, so that American league teams played only other American league teams, right up until the World Series. Back in those days, our calculation would have been a good bit less muddy.

[9]I originally typed this as Federation, and then I realized that many years ago WWF also stood for *World Wrestling Federation*! This is the organization now known as the WWE.

[10]When SAS first began, in the days of punched cards, everything was in UPPERCASE, which is how you used to have to run SAS as well. You still occasionally see SAS code in textbooks written this way.

[11]Applied Statistics is an art as much as a science at times.

[12]Which was me.

[13]"The default number of bins is determined by the system", it says, helpfully.

[14]In power calculations done with `proc power` later, the number `1` is used, since the results are the same whichever one side is used.

[15]For reasons of statistical power, which we study later, it is good to aim for the same number of individuals in each group. That is, if you have the same number in each group, you maximize your chances of finding a difference, if there really is one.

[16]It is possible to get outliers at both ends, and then you might be wondering about skewness. I would say in that case that your major problem is outliers at both ends, and leave it at that. Boxplots were designed for small-to-moderate data sets, which is typically what you saw in the 1950s when Tukey invented boxplots; with a bigger data set, you can happen to get a lot more outliers just by chance, so if your data set is big, it may not be worth worrying about a few outliers.

[17]You'll recall that the t-tests are "robust" to non-normality, which means that you can get away with the data not being all that normal, at least if you have large samples.

[18]A lowercase `l` also works, but I prefer not to use it because it looks too much like a number `1`. When you specify a one-sided alternative, you need to say *which* one side you want, lower or upper.

[19]Compare this boxplot to the R one from earlier, where the spreads of the two groups look a lot more similar. R and SAS have different default definitions for quartiles (there are actually a *lot* of possible definitions), so the boxplots look different. The consequence of this is that your choice of test might be different when using R vs. when using SAS, though the conclusion, as you see from the two P-values, is pretty much identical either way.

[20]Throwing your hat across the room.

[21]I think that is free to view. Find it via `library.utoronto.ca` if not.

[22]With ν degrees of freedom, the variance is $\nu/(\nu - 2)$.

[23]Some people call this the "Behrens-Fisher problem". See that name Fisher again?

[24]R calls it the "Welch t-test".

[25]This is the issue around `guessingrows` which I talk about elsewhere.

[26]If you sample *without* replacement, you just get the original samples back, which is kind of pointless.

[27]You might call this a "stratified resample".

[28]This is known as "curly-curly" and comes from the `rlang` package.

[29]Actually, it was a Libre Office spreadsheet before that, but I am trying not to confuse you too much.

[30]Rounded up: SAS has done the proper rounding for you.

[31]This is the value of *keeping all your work* and *being able to find it later.*

[32]Which is in the UK. I interviewed there once, a long time ago.

[33]Idiom; something equivalent is "this smells fishy".

[34]There are also boxplots in with the `proc anova` output.

[35]If we were doing two-way ANOVA, which we are not, we would ignore the Model line and get the tests for each grouping variable from the second table. But here, it doesn't matter. Either place is good. It's the same P-value twice.

[36]This talks about *means* rather than individual observations; in individual cases, sometimes even drug A will come out best. But we're interested in population means, since we want to do the greatest good for the greatest number.

[37]"Greatest good for the greatest number" is from Jeremy Bentham, 1748–1832, British philosopher and advocate of utilitarianism.

[38]This might be because riding mowers are bigger, so their blades would be bigger too.

[39]In the one-way ANOVAs we have seen before, it isn't necessary to have the same number of observations in each group, but as soon as you go beyond that, having unequal numbers of observations per combination makes the analysis a lot more awkward. This poses a problem for real-life experiments, because animal or human subjects may drop out of the experiment and you don't get a response value for those, suddenly making your experiment unbalanced.

[40]This is characteristic of exponential growth, which is key to what is coming up.

[41]Or exponential decay, with $b < 0$.

[42]These are residuals on the log scale: they say how far different the observed and predicted `logyy` values are

[43]There are only 6 of them anyway; with this few points we are likely to get some kind of apparent pattern just by chance.

[44]Devotees of other languages will note that R arrays start at 1, not 0.

[45]Or, in my case, you *know*.

[46]But, of course, real-life data won't cooperate quite as nicely as this.

[47]If you're as old as I am, you'll remember a TV series called *ER* starring a very young George Clooney, which was set in Cook County Hospital, in Chicago.

[48]I made a whole set of residual plots this way while on the bus, in case this was the way it had to be done. When riding the TTC I try to grab one of the sideways-facing seats, so that I have room to open my laptop.

[49]In fact, it is believed that wolves help keep caribou herds strong by preventing over-population: that is, the weakest caribou are the ones taken by wolves.

[50]The survey is always taken in the fall, but the date varies.

[51]Counting animals in a region, especially rare, hard-to-find animals, is a whole science in itself. These numbers are probably estimates (with some uncertainty).

[52]You are probably too young to remember that. Everyone thought the world was going to end when all the computers went to year `00` and we thought they would interpret it as 1900.

[53]Because the wolves have more to eat.

[54]Baby wolves.

[55]Ken makes a rude noise.

[56]The two dashes between the variable names are like `:` in R: "the first one through the second one, inclusive."

[57]Or, more likely, one of the doctor's sleep-deprived surgical residents.

[58]The same Sir David Cox of Box-Cox.

[59]Jan 1, 1960 is SAS's "zero" date.

[60]If `span` is less than 1, only that nearest fraction of points to where you are fitting is used, with nearer points having a higher weight. If `span` is 1 or bigger, all the points are used, but nearby points still have a higher weight.

[61]Environmental scientists like the Mann-Kendall correlation, rather than the standard "Pearson" correlation, because it is not assuming a linear trend, just a "monotone" one that keeps going up or keeps going down, and it is not affected by outliers, which natural data tends to have rather a lot of. The Kendall correlation also treats the variables as "ordinal" rather than "interval": like the sign test, it only thinks about whether the points are indicating an uphill or downhill trend, not how big of a trend it is. For *that*, they use a thing called the Theil-Sen slope, which is the median of the pairwise slopes between the points, again not affected by outliers.

[62]For those who know about "32-bit signed integers", starting in 1970 means that dates in Unix and Linux can be represented by one of these up until January 19, 2038, at which date it will wrap around to December 13, 1901: as many days before Jan 1 1970 as Jan 19 2038 is after. Many of us on Unix-like systems (including Macs) are now using 64-bit signed integers; Wikipedia tells me:

> Using a signed 64-bit value introduces a new wraparound date that is over twenty times greater than the estimated age of the universe: approximately 292 billion years from now, at 15:30:08 UTC on Sunday, 4 December 292,277,026,596.

I guess this problem is now solved.

[63]If you have a Mac, you can do something very similar by firing up a terminal window. I think Windows has `bash` now, which is what this is.