

STAC32

Assignment 7

Due Thursday November 8 at 11:59pm

The first few questions (here the first three) are to be done in the tutorial, or on your own, and they come with solutions so that you can check what you did (or find out where you went wrong).

The question at the end (without solutions) is an assignment: you need to do this question yourself and hand it in (instructions below). These are due on the date shown above. An assignment handed in after the deadline is late, and may or may not be accepted (see course outline). My solutions to the assignment questions will be available when everyone has handed in their assignment.

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 the 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.

If your assignment is disorganized or otherwise difficult for the grader to deal with, you can expect to lose marks.

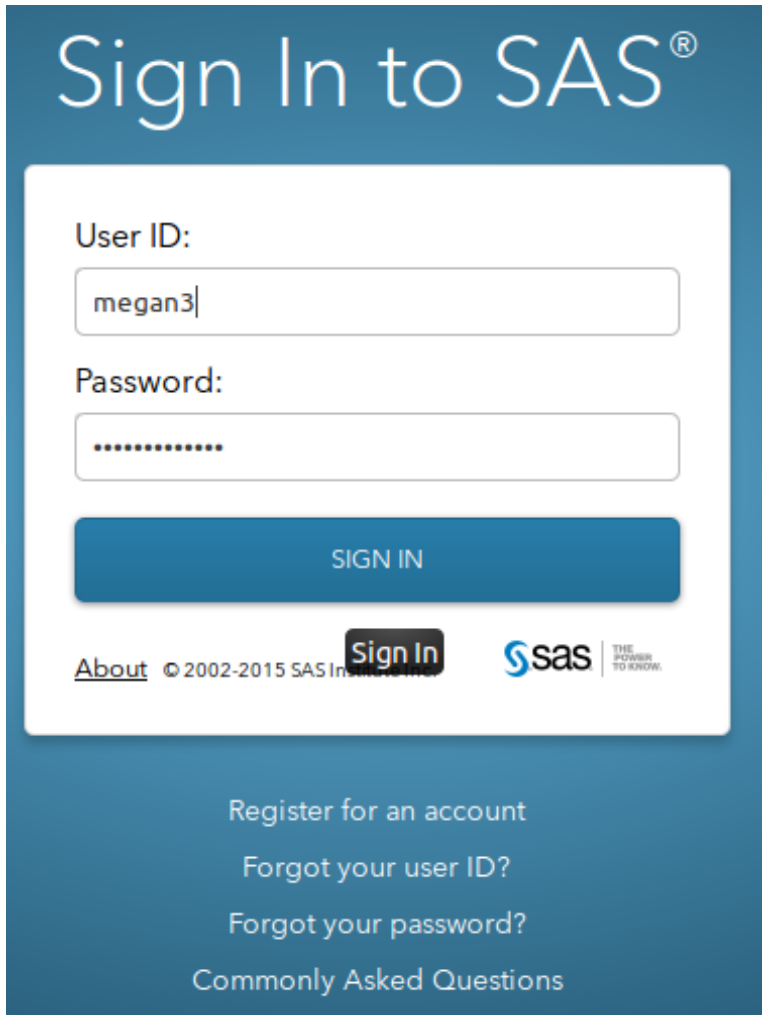
See <https://www.utoronto.ca/~butler/c32/quercus1.nb.html> for instructions on handing in assignments in Quercus. You can (and for SAS assignments, *should*) hand in a Word document.

Reminder: As with any other course involving software, *there are no extensions due to failure to access software*. It is *your* responsibility to make sure that you allow yourself enough time to get connected to SAS

Studio (likely the biggest source of problems) and to get your work done. Bear in mind that you will be competing with a lot of other people, here and around the world, for access to SAS's servers, so it is up to you to allow enough time.

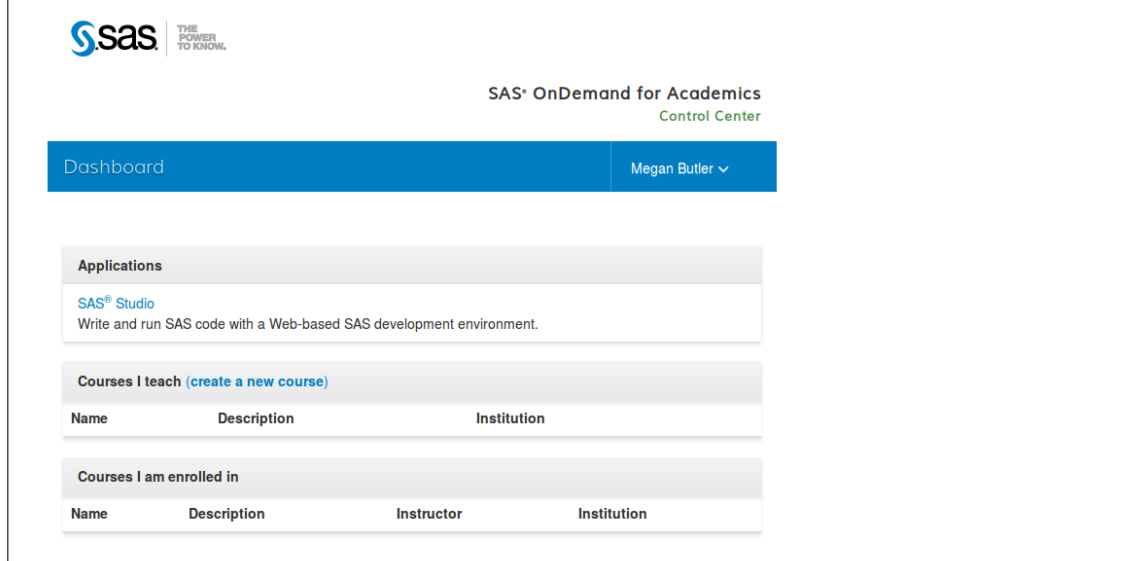
1. This question will introduce you to SAS.

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):



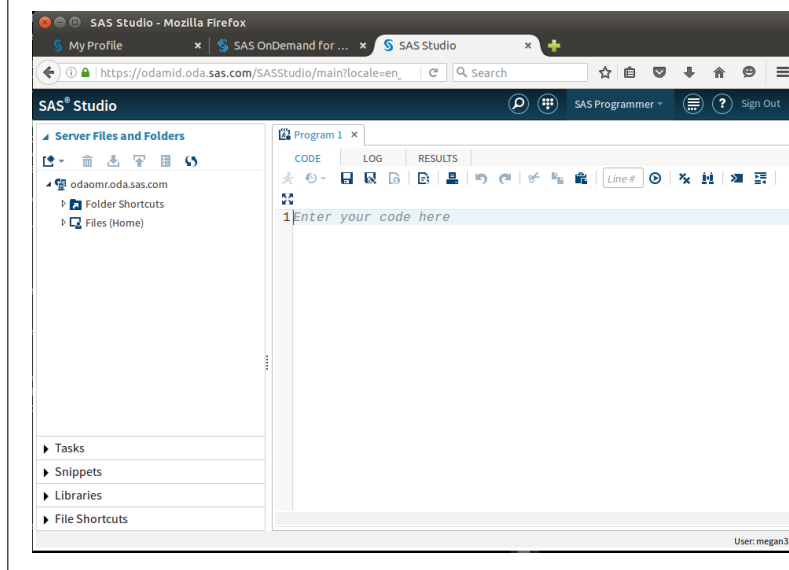
- (a) Click on Register for an Account (unless you happen to have one already, in which case sign in with username and password). Fill in your first name, last name, e-mail address (twice) and select Country. Click Submit.
- (b) Check your e-mail (the address you used above). There should be an e-mail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



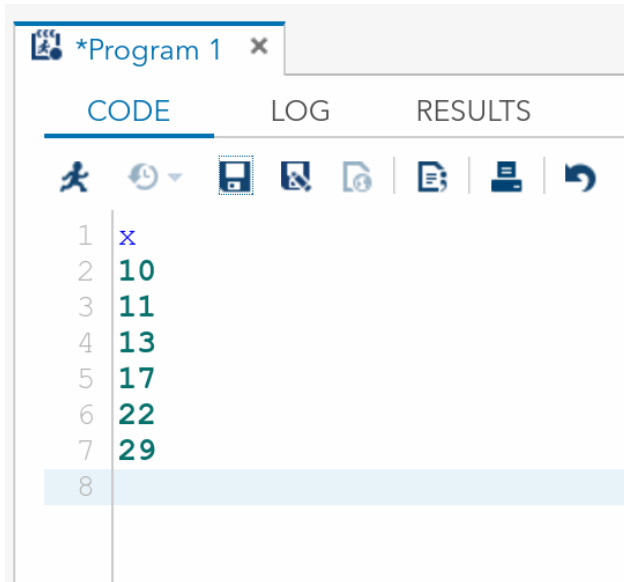
- (d) Click on SAS Studio. You will eventually see something like what appears in the Solution below. This is usually the slowest part of the whole operation. If it seems to be taking a long time,³ leave the rest of this question for now and come back to it later.

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

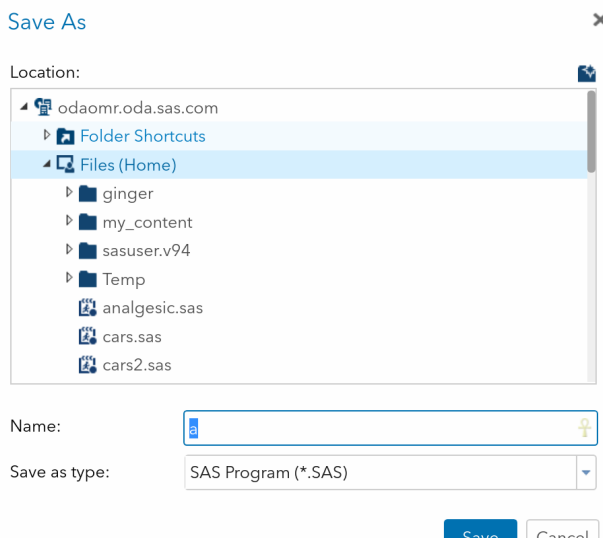


- (e) Go over to where it says "Enter your code here". We are going to do two things: first, we'll enter some data and save it, and then (in the next part), we'll write some code to read in the data and run that code.

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:



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:



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:

```
1 proc import
2 datafile='/home/megan3/a.sas'
3 dbms=csv
4 out=mydata
5 replace;
6 getnames=yes;
7
8 proc print;
9
```

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:

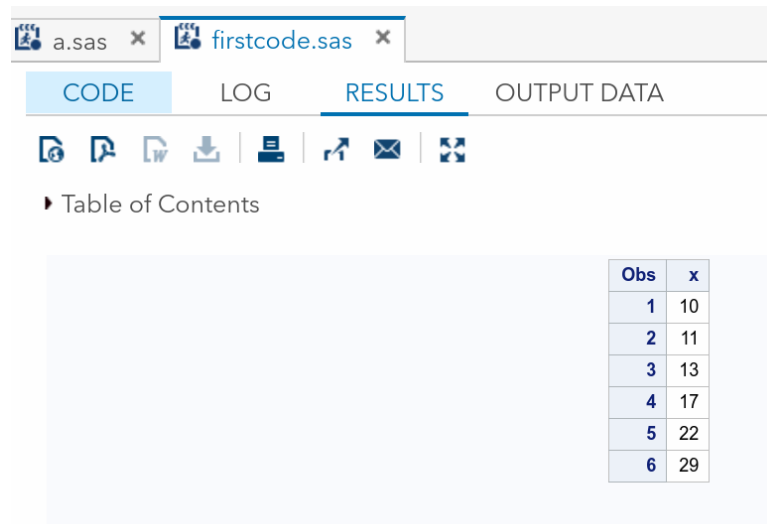
1. where the data file is stored. Mine means “the file called `a.sas` under the account with username `megan3` on SAS's servers”.
2. the kind of file it is. We are pretending that this is a `.csv` file, even though it isn't really.
3. the name of the SAS data set to create (which doesn't matter here 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.

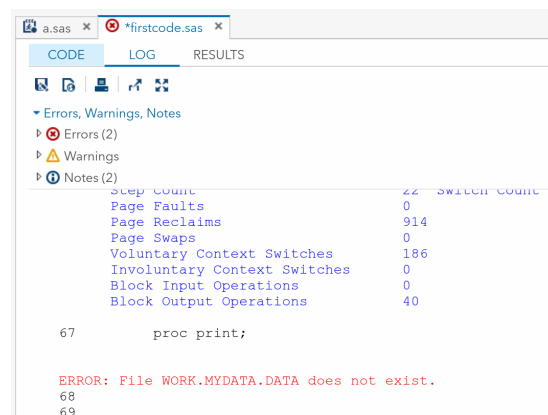
Solution: One of two things will happen: either it will work, and you'll see your data set displayed in the Results tab:



The screenshot shows the SAS Studio interface with two tabs: 'a.sas' and 'firstcode.sas'. The 'RESULTS' tab is active, displaying a table with two columns: 'Obs' and 'x'. The table contains six rows of data.

| 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**:



The screenshot shows the SAS Studio interface with the 'LOG' tab active. It displays a list of errors, warnings, and notes. The first error is highlighted in red.

| Step | Count | 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;
68
69      ERROR: File WORK.MYDATA.DATA does not exist.

```

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:

CODELOGRESULTS



```
1 proc import
2   datafile='/home/megan3/a.sas'
3   dbms=csv
4   out=mydata
5   replace;
6   getnames=yes;
7
8 proc print;
9
10 proc means;
11
12 proc sgplot;
13   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;

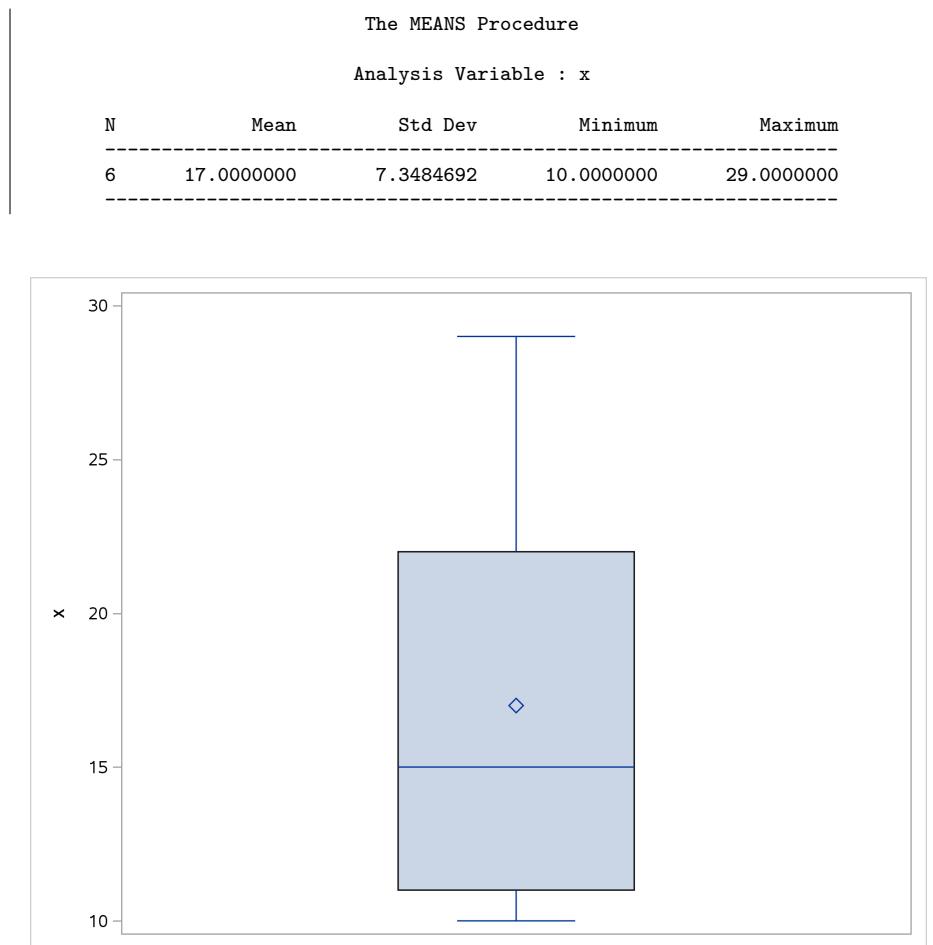
proc print;

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 |



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. 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:

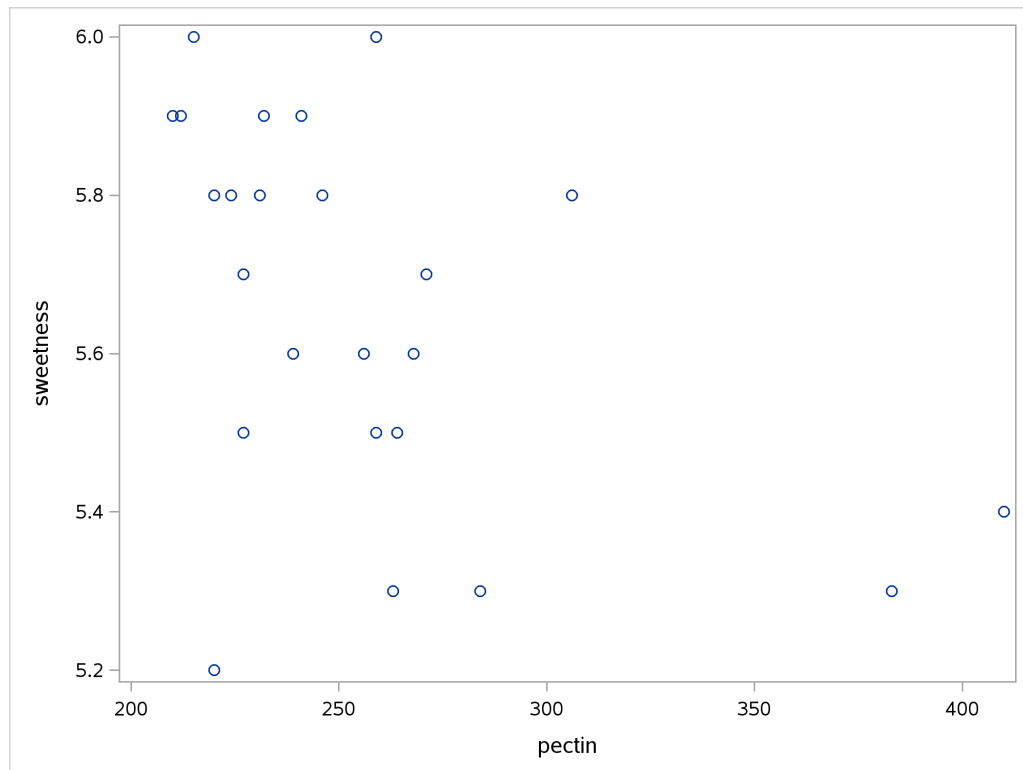
| Obs | run | sweetness | pectin |
|-----|-----|-----------|--------|
| 1 | 1 | 5.2 | 220 |
| 2 | 2 | 5.5 | 227 |
| 3 | 3 | 6 | 259 |
| 4 | 4 | 5.9 | 210 |
| 5 | 5 | 5.8 | 224 |
| 6 | 6 | 6 | 215 |
| 7 | 7 | 5.8 | 231 |
| 8 | 8 | 5.6 | 268 |
| 9 | 9 | 5.6 | 239 |
| 10 | 10 | 5.9 | 212 |
| 11 | 11 | 5.4 | 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;
```

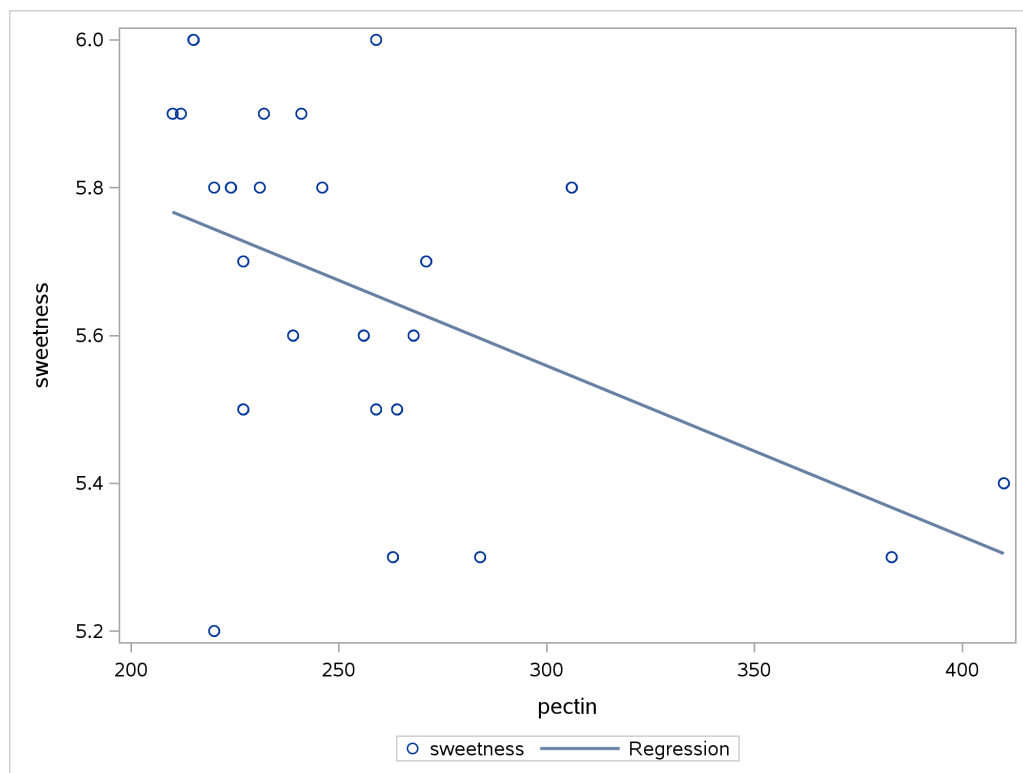


This came out the same as my R plot.

- (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!

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.uts.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.)

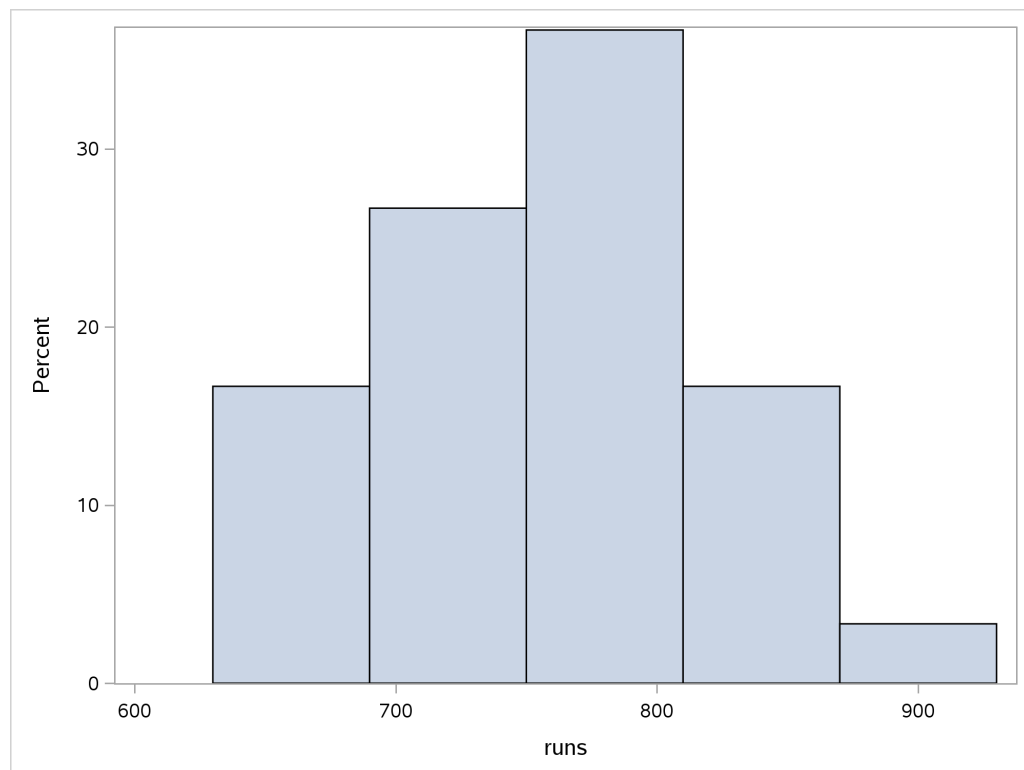
This was (when it was to be handed in) only one point, so if you've recognized that some of the team names have spaces in them, I'm good.

- (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;
```

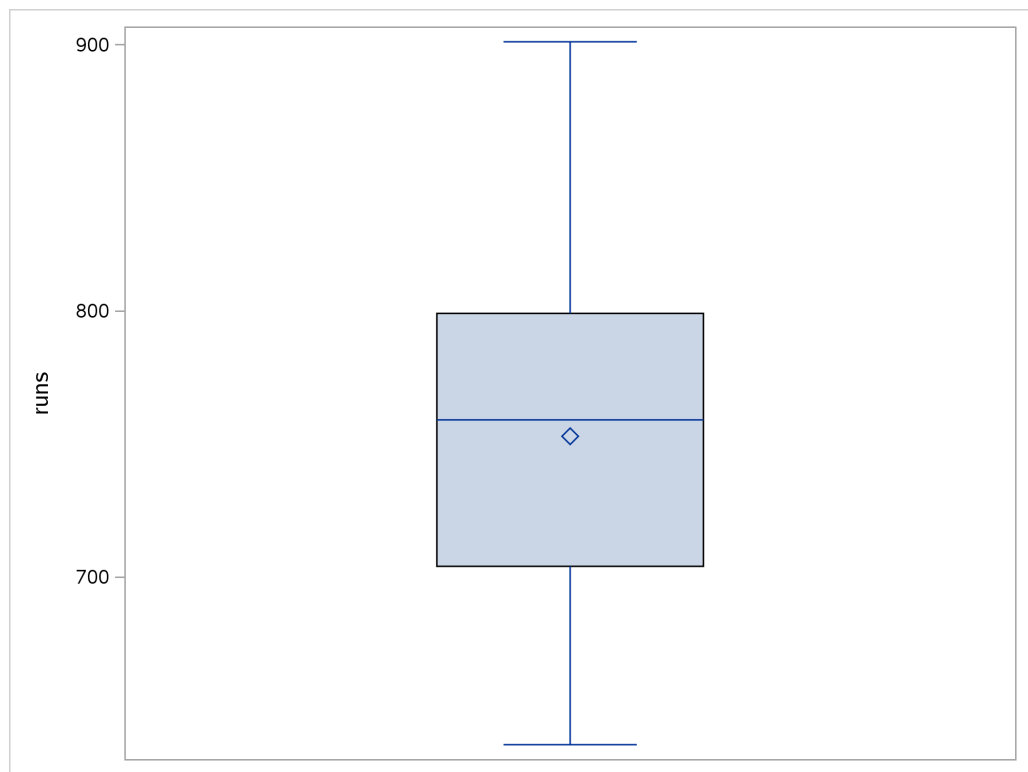
giving



Another possibility is a boxplot:

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

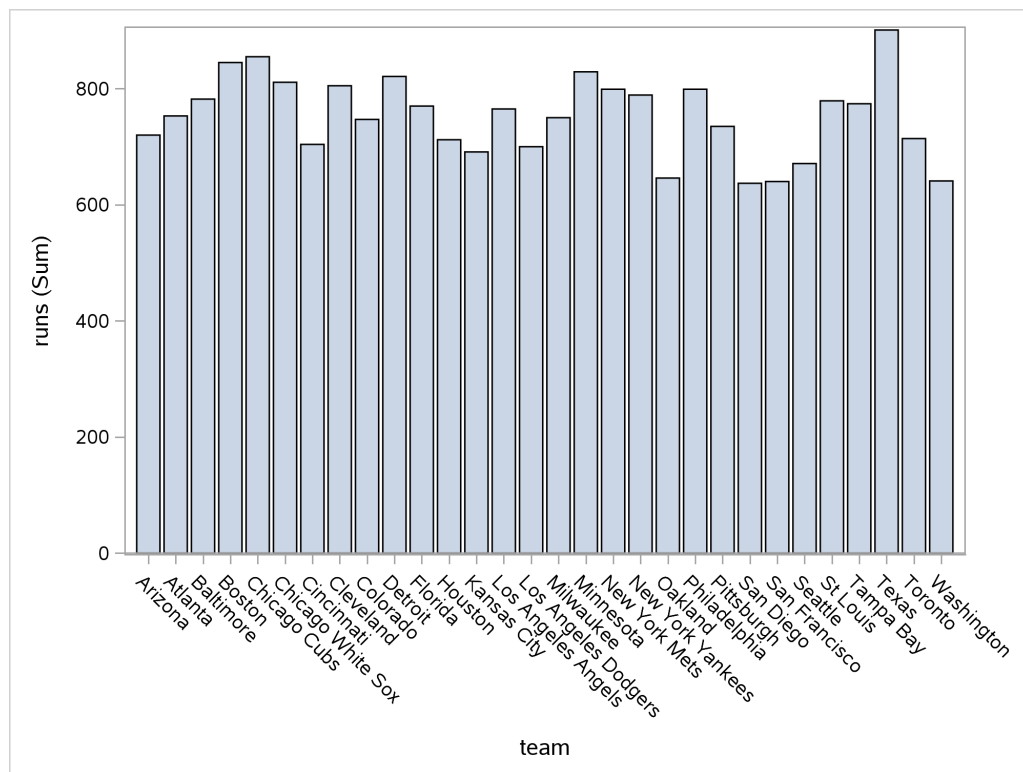
which gives



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

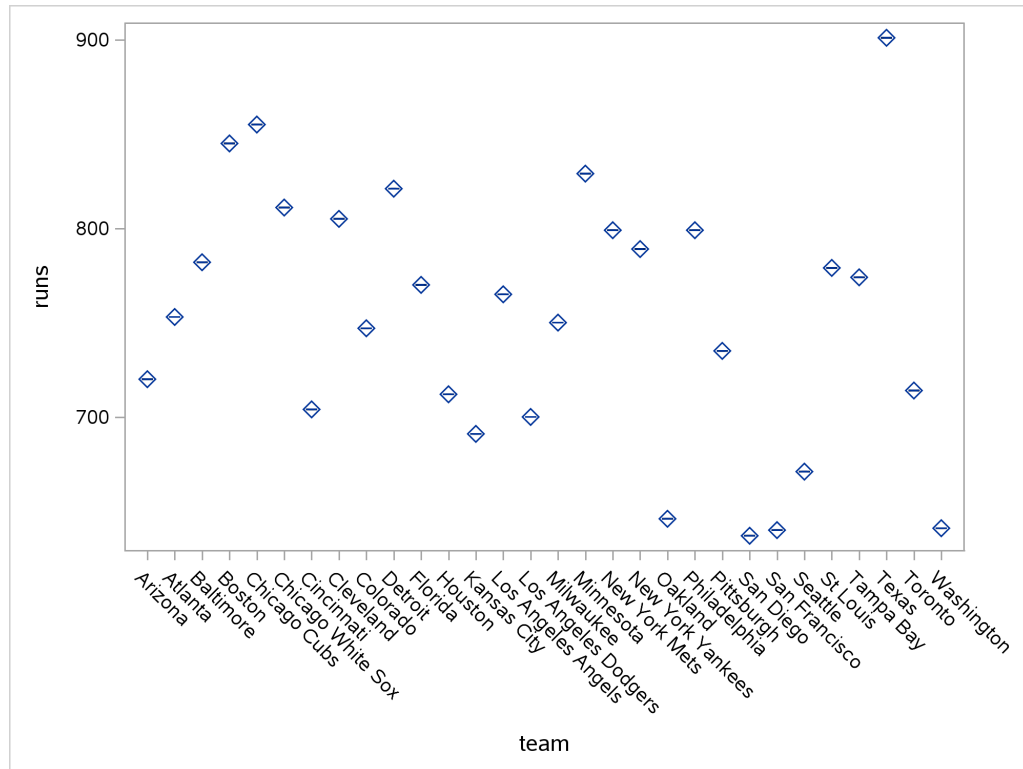
This was meant to be easy, but you could also read the question as asking you to show the team names on your graph as well, which makes it more difficult (and beyond what I showed you in class). I think the best graph along these lines would be a bar chart but instead of having frequency on the y axis,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

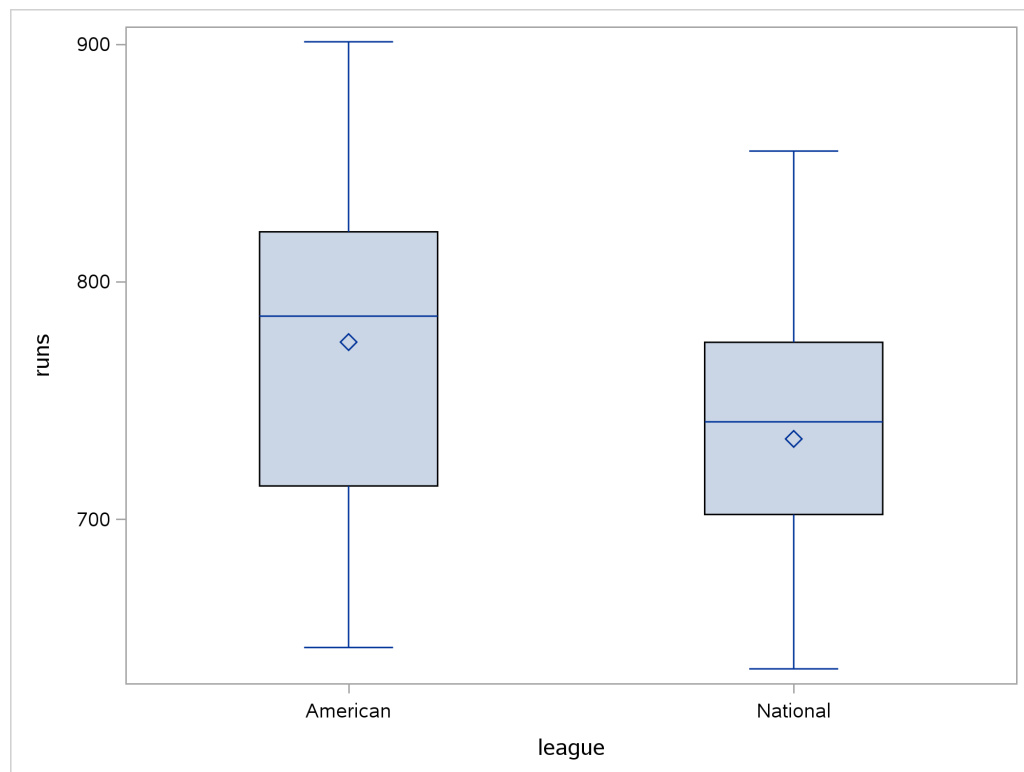
- (d) The American League has a rule called the “designated hitter rule”. This means that in games where an American League team is playing on their home field, both teams have a player who only bats (does not field), who bats in place of the pitcher. When a National League team is playing at home, the 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.

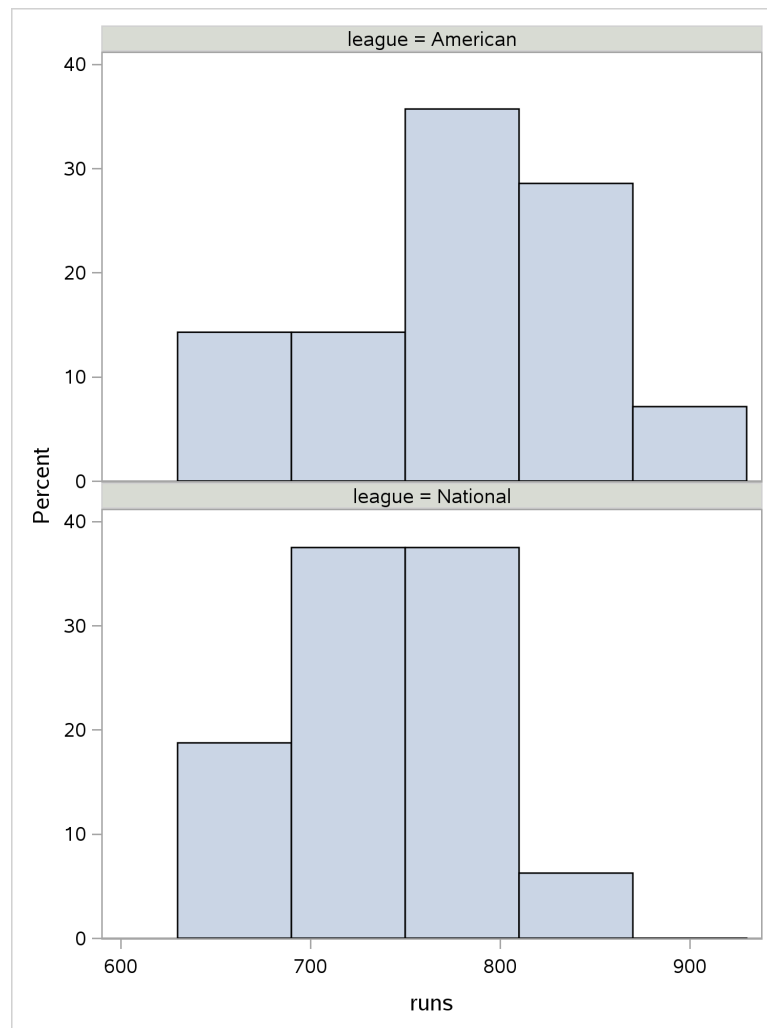
Or, you can say that there might be other factors that would also explain the difference between the two leagues (and then name one or two). The one that first comes to my mind is stadium size: it might be (I haven't investigated) that American league teams typically have smaller stadiums, in which it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

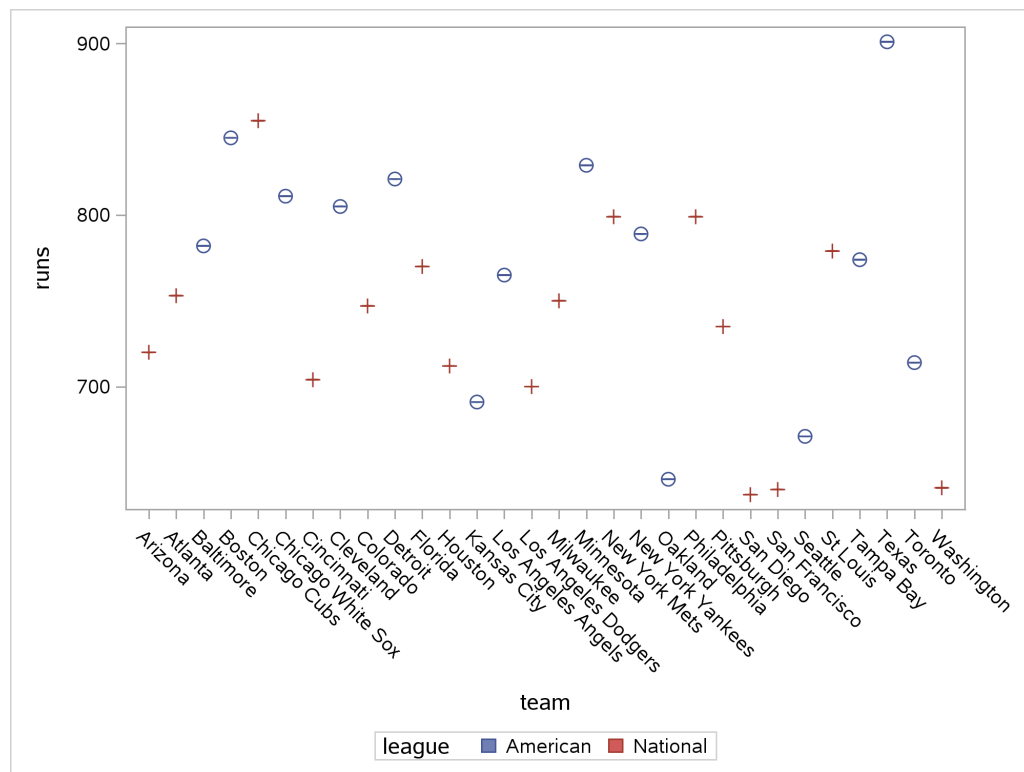
That produces:



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.

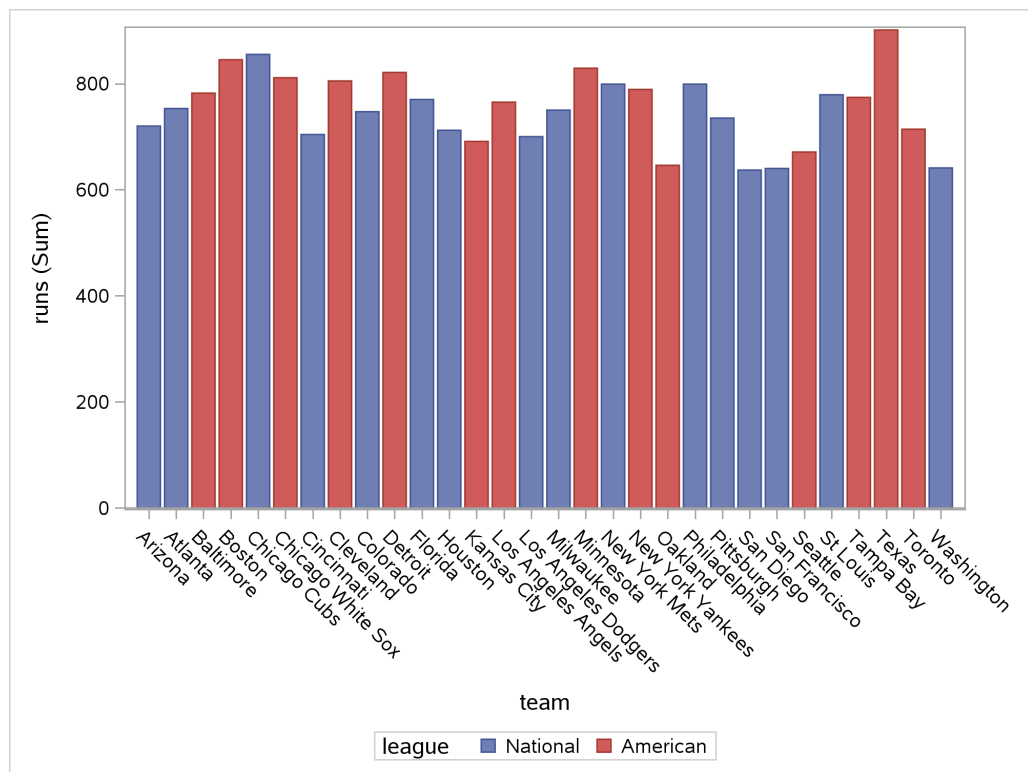
If you thought that you needed to show team names on your graph again, then you need to distinguish the leagues somehow. I think the easiest way to do that is to start from the not-really-boxplot:

```
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:

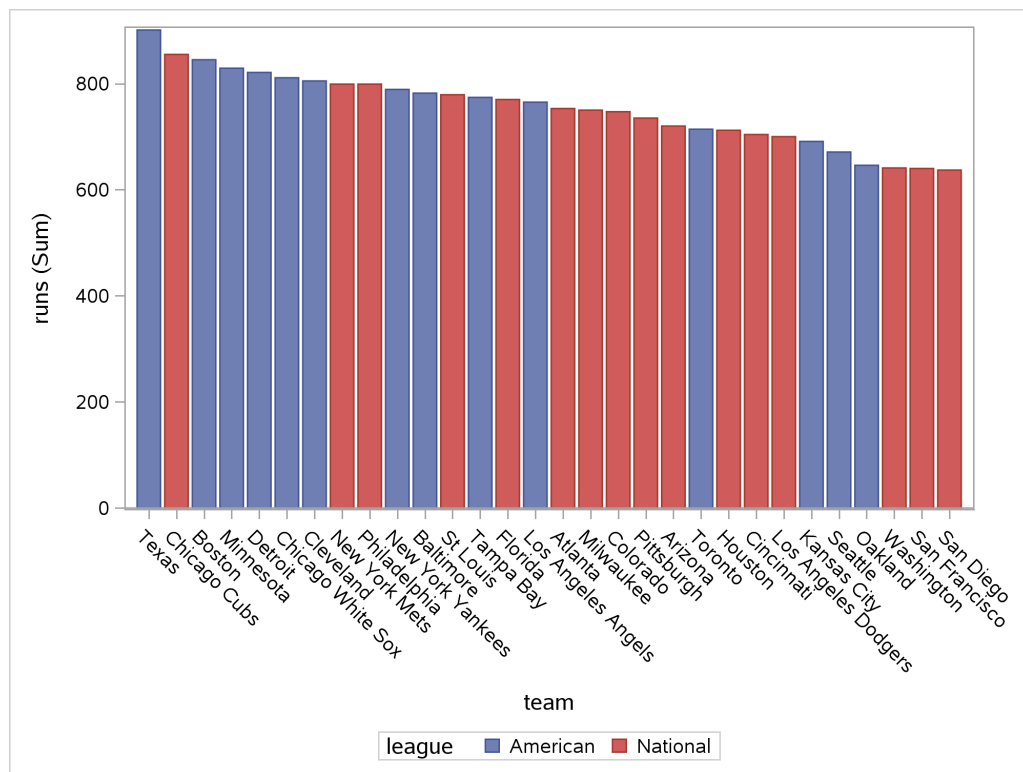
```
proc sgplot;
  vbar team / response=runs group=league;
```



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 | Infy | | 66.3890 |
| Diff (1-2) | Satterthwaite | 40.7589 | -1.1183 | Infy | | |
| 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.

4. 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);
```

| Obs | Father_Age | Mother_Age | Weeks_ Gestation | Pre_natal_ Visits | Marital_ 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 |

| Obs | Mother_ Weight_ Gained | Low_Birthweight_ | Weight__ 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 | | |
| ----- | | ----- | | |

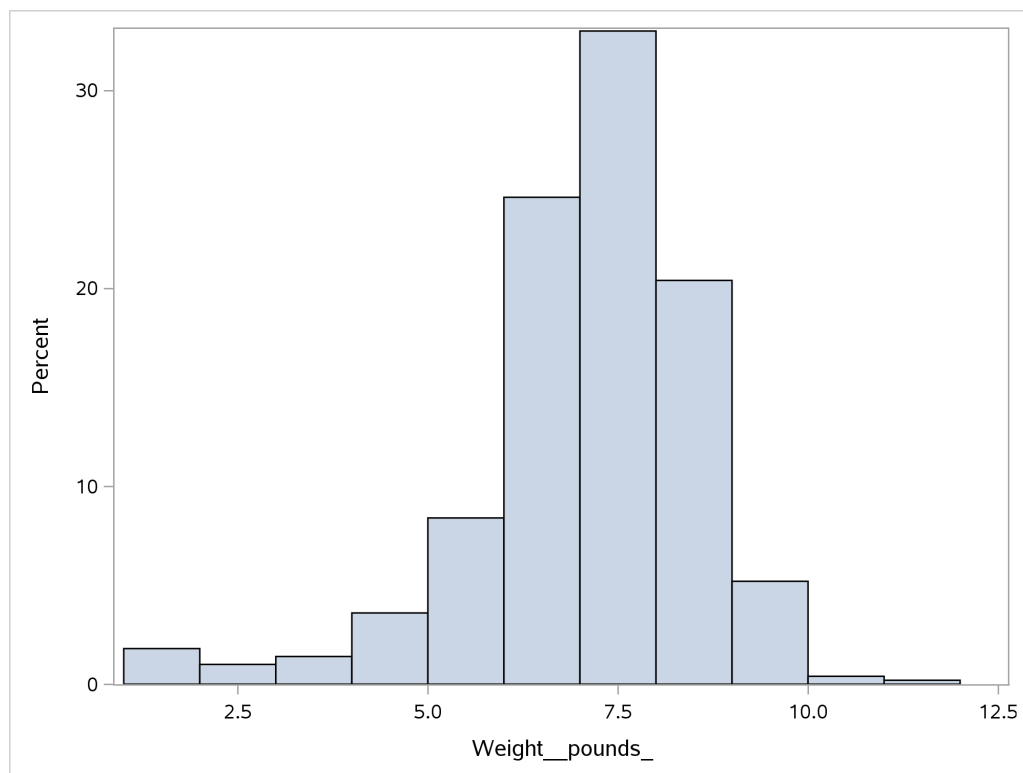
The value in the N column is the number of observations for each variable. Or, more precisely, since we know there were 500 rows, these are the number of *non-missing* observations for each variable. Looking at all 500 rows, the father's age was the most often missing, something we would have guessed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

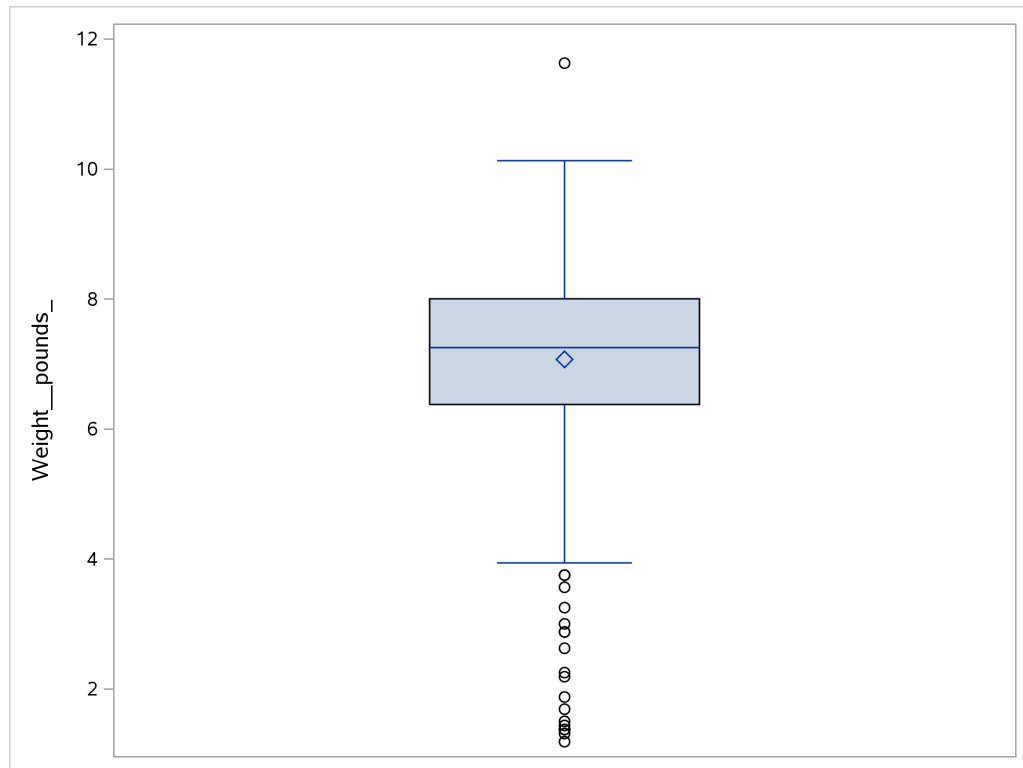


SAS chose 11 bins (with bin width 1 and bin boundaries on the whole numbers). This is similar to the 10 bins that Sturges' rule gives.

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.

5. 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_ | | | | | | |
| Premie_ | 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.)

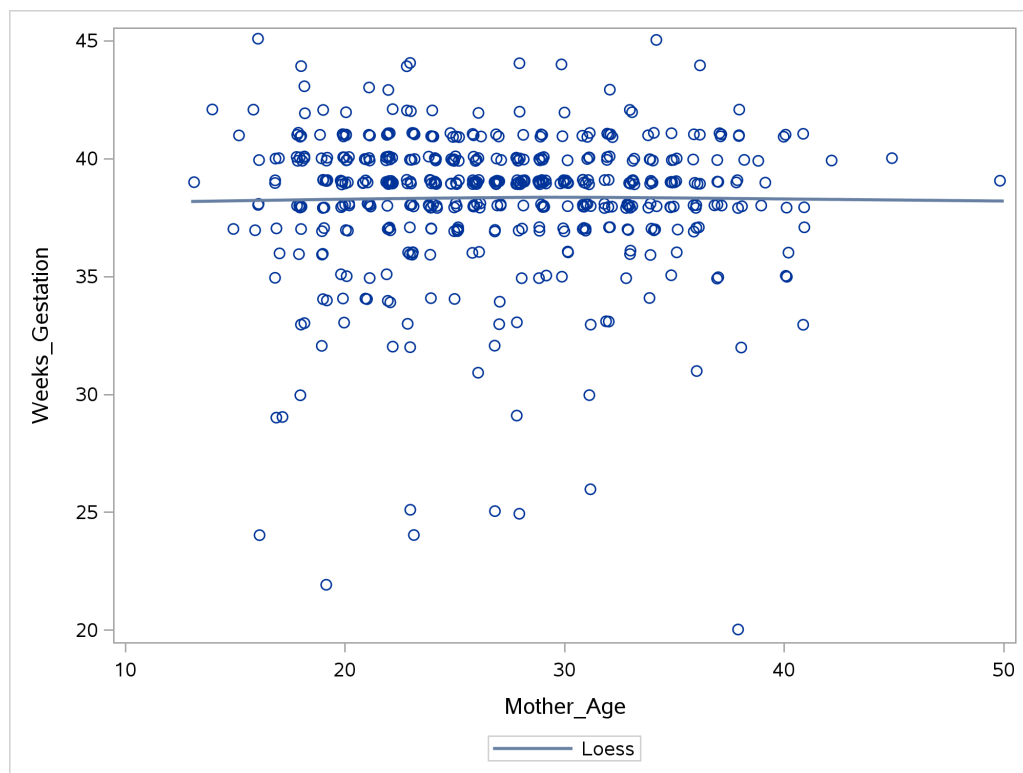
```
proc sgplot;
  scatter y=Weeks_Gestation x=Mother_Age;
```


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;
```



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;
```

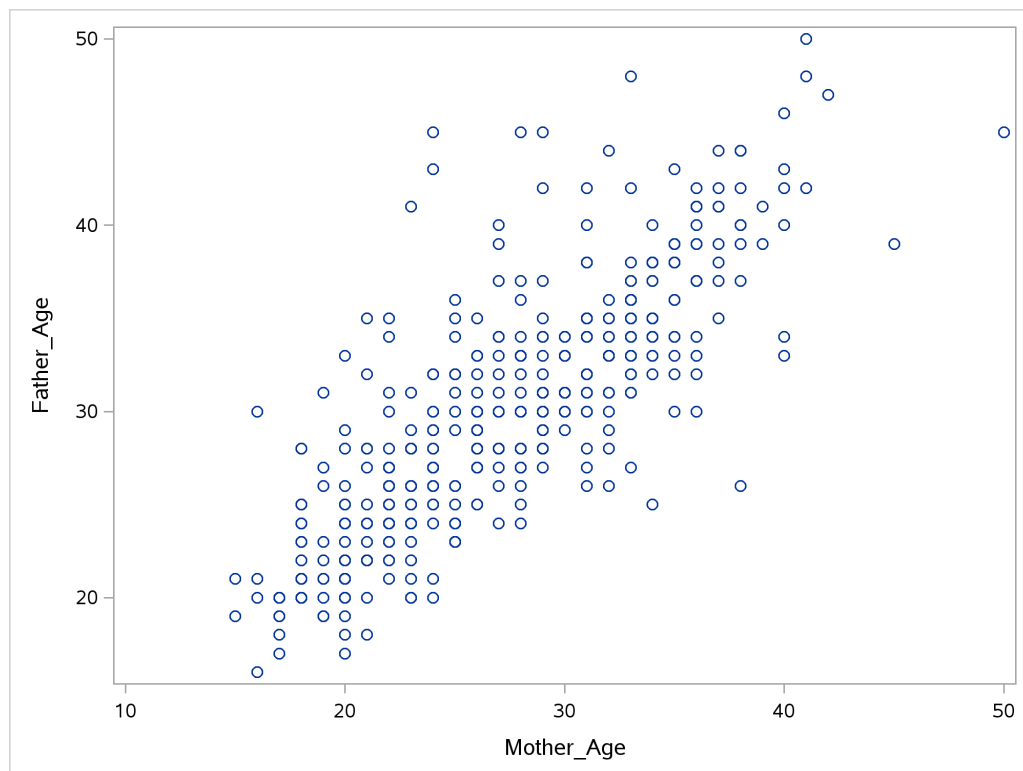
| The MEANS Procedure | | | | | | |
|-----------------------------|----|-----|-----------|-----------|---------|-----------|
| Analysis Variable : Premie_ | | | | | | |
| Mother_Age | N | Obs | Mean | Std Dev | Minimum | Maximum |
| 13 | 1 | 1 | 0 | . | 0 | 0 |
| 14 | 1 | 1 | 0 | . | 0 | 0 |
| 15 | 2 | 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?

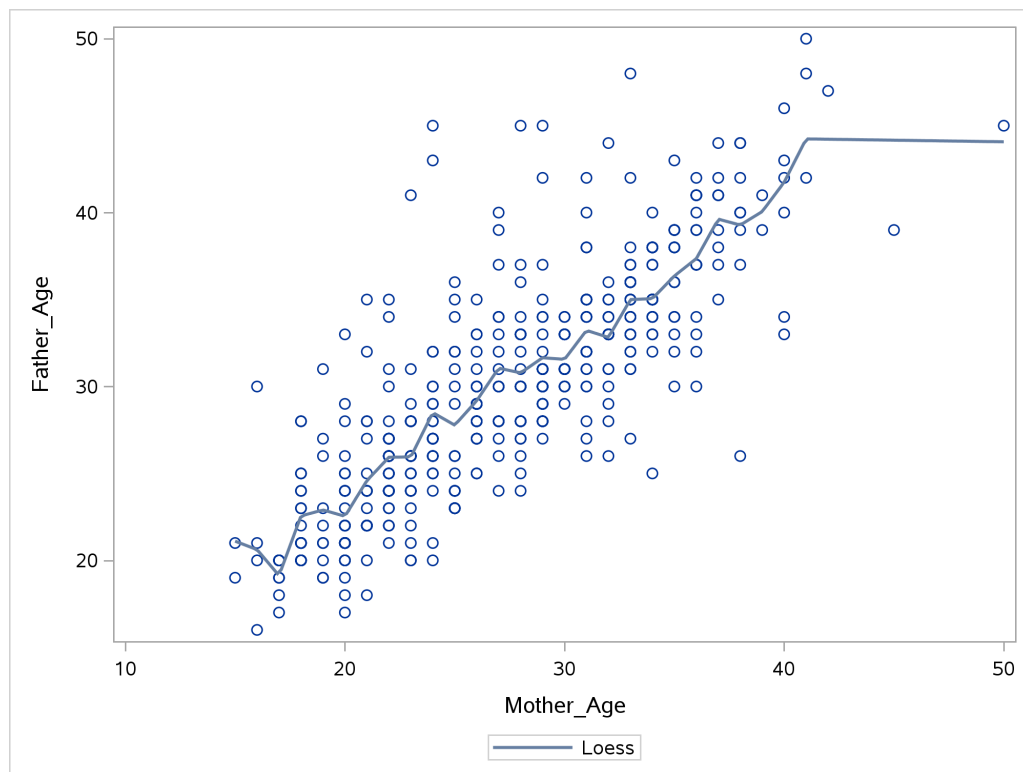
Solution: This one is also most obviously a scatter plot:

```
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_Age | Father_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 | | | | | |
| Source | DF | Sum of Squares | Mean 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 | | | | | |
| Variable | DF | Parameter Estimate | Standard 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.

- 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.

- 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.utoronto.ca/~butler/c32/nenana.txt";

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

proc means;
```

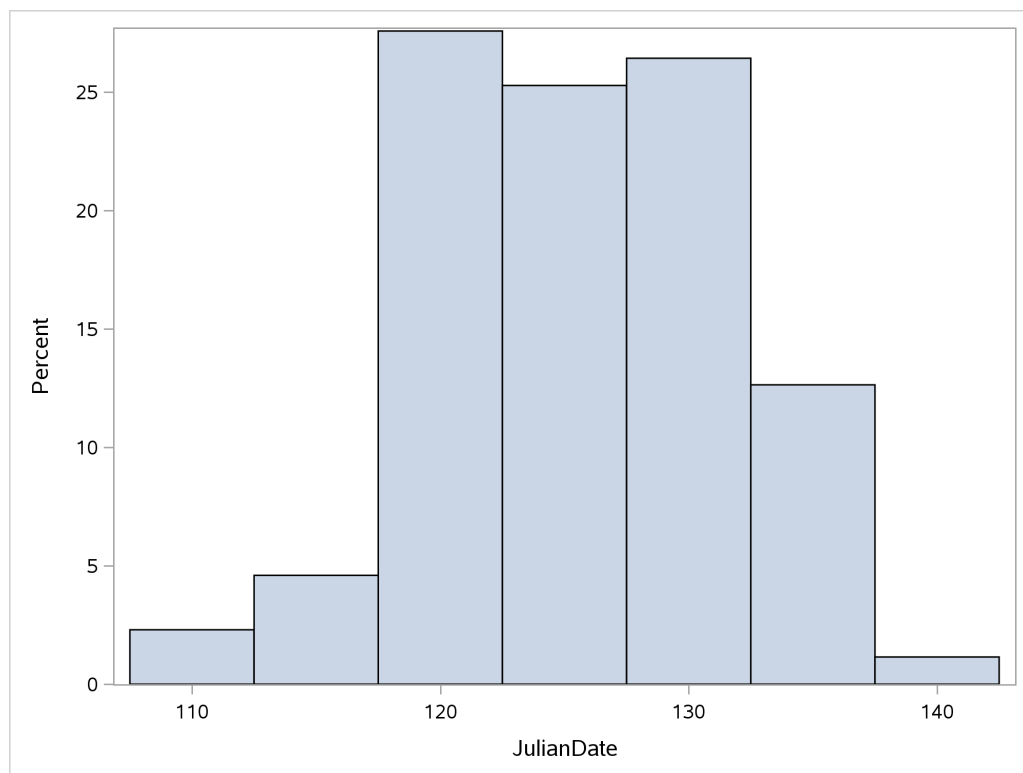
| The MEANS Procedure | | | | | |
|---------------------|----|-------------|------------|-------------|-------------|
| Variable | N | Mean | Std Dev | Minimum | Maximum |
| Year | 87 | 1960.00 | 25.2586619 | 1917.00 | 2003.00 |
| JulianDate | 87 | 125.5443126 | 5.9317755 | 110.7045000 | 141.4872000 |

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;
```

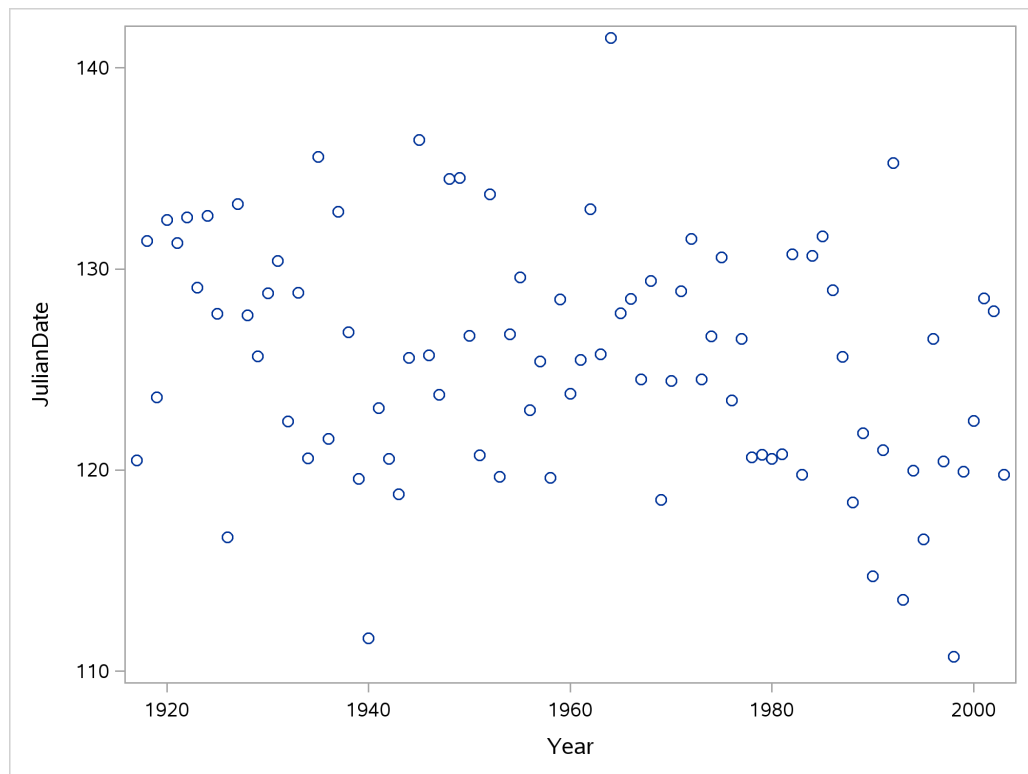


That looks more normal than my R histogram, because SAS used a different number of bins, seven rather than eight, and the choice of bins makes a difference to the look.

- (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.

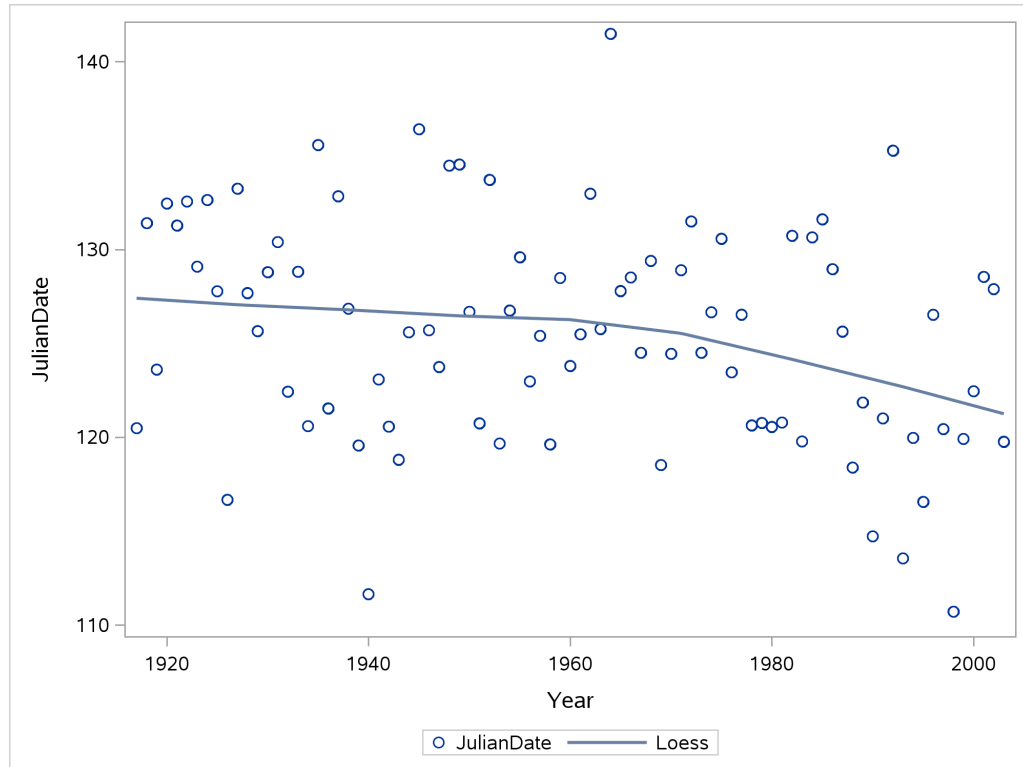
Solution:

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



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).

7. 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;
```

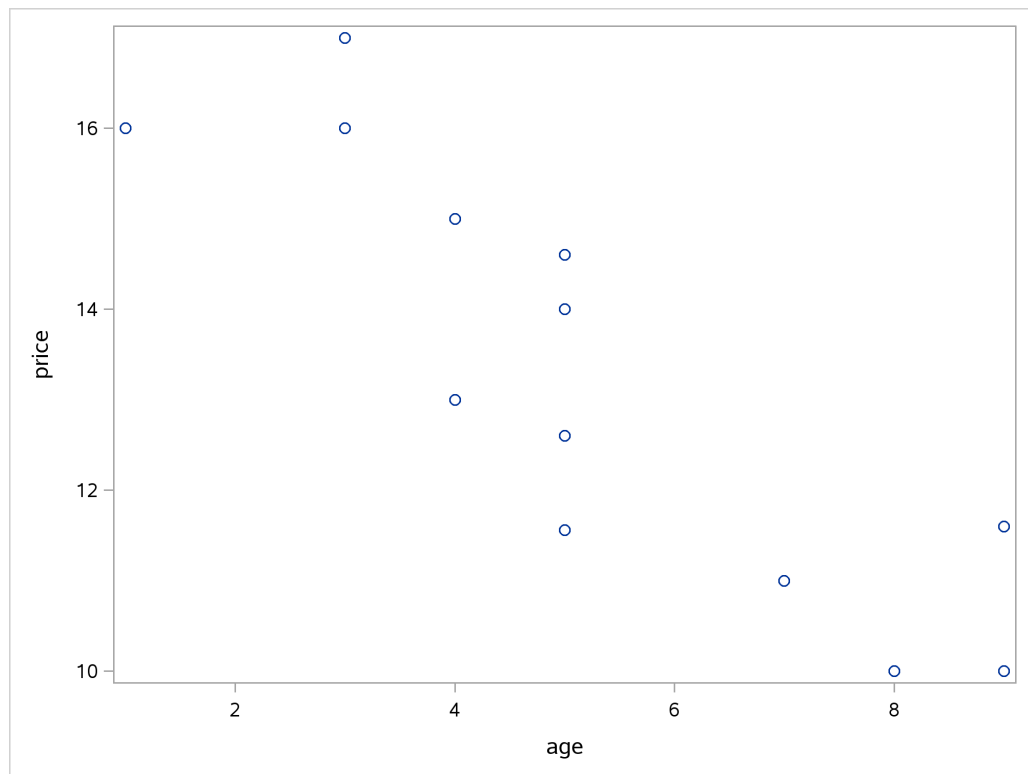
| Obs | age | price |
|-----|-----|-------|
| 1 | 9 | 11.6 |
| 2 | 4 | 15 |
| 3 | 4 | 13 |
| 4 | 7 | 11 |
| 5 | 3 | 16 |
| 6 | 5 | 14.6 |
| 7 | 5 | 11.56 |
| 8 | 8 | 10 |
| 9 | 9 | 10 |
| 10 | 1 | 16 |
| 11 | 5 | 12.6 |
| 12 | 3 | 17 |
| 13 | 5 | 14 |

(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):

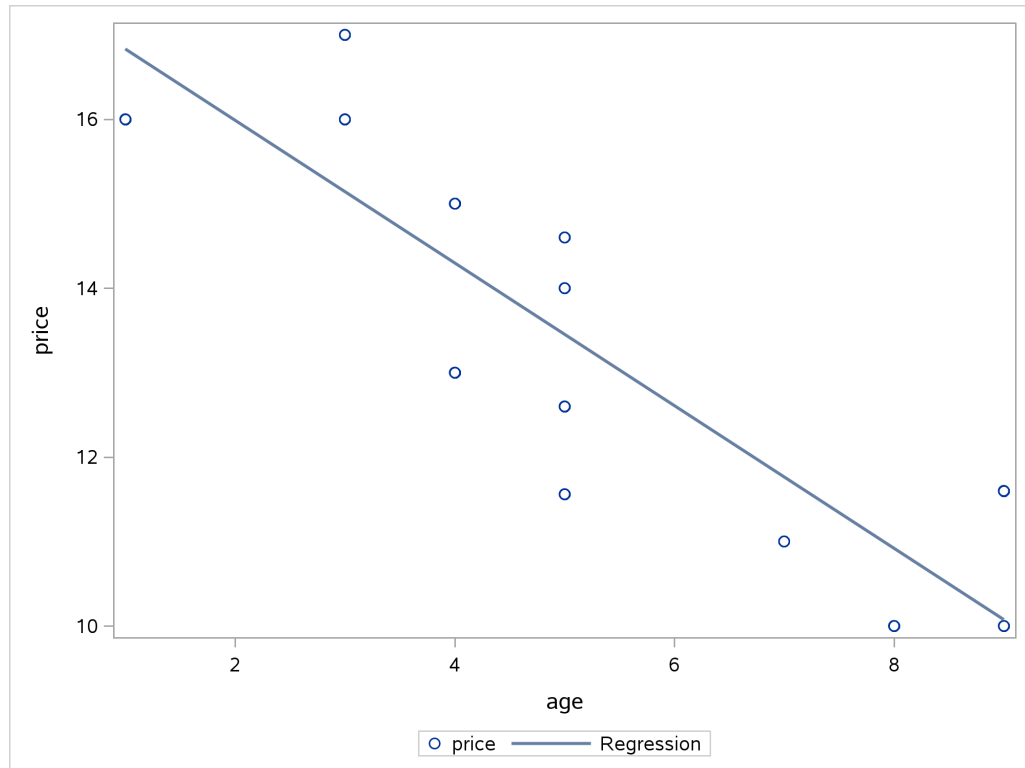
```
proc sgplot;  
  scatter y=price x=age;
```

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



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

| Median | Quartile 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.

Hand the next two questions in:

8. A statistics course had two lecture sections, taught by the same instructor. One section used a typical variety of examples from different subject areas, while the other section used only sports-themed examples. The lecture sections were labelled as such, so that a student knew which type of section they were enrolling in, and could choose the one they preferred. The sections are labelled **Regular** and **Sports** in the data file. The sports-themed section had its lectures earlier in the day than the regular section.

The data are in <https://www.utsc.utoronto.ca/~butler/c32/SportsExamples.csv>. For each student, the total number of points earned in the course was recorded, as well as which section they were in and their letter grade.

- (a) (4 marks) Read the data into SAS, and display the mean **TotalPoints** for each lecture section.

Solution: This is a .csv file (making life easier for you). It's a file on the web, so do the **filename** thing first:

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

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

Extra: for *yourself*, you should probably run **proc print** and convince yourself you have the right thing. Don't hand it in, though:

```
proc print;
```

| Obs | Total_Points | Section | Grade |
|-----|--------------|---------|-------|
| 1 | 340 | Regular | B |
| 2 | 322 | Regular | B |
| 3 | 302 | Regular | C |
| 4 | 382 | Regular | A |
| 5 | 304 | Regular | C |
| 6 | 349 | Regular | B |
| 7 | 360 | Regular | A |
| 8 | 347 | Regular | B |
| 9 | 376 | Regular | A |
| 10 | 364 | Regular | A |
| 11 | 332 | Regular | B |
| 12 | 310 | Regular | C |
| 13 | 324 | Regular | B |
| 14 | 353 | Regular | B |
| 15 | 361 | Regular | A |
| 16 | 265 | Regular | D |
| 17 | 377 | Regular | A |
| 18 | 332 | Regular | B |
| 19 | 275 | Regular | D |
| 20 | 322 | Regular | B |
| 21 | 352 | Regular | B |
| 22 | 368 | Regular | A |
| 23 | 317 | Regular | C |
| 24 | 352 | Regular | B |
| 25 | 341 | Regular | B |
| 26 | 337 | Regular | B |
| 27 | 378 | Regular | A |
| 28 | 296 | Regular | C |
| 29 | 366 | Regular | A |
| 30 | 289 | Sports | C |
| 31 | 315 | Sports | C |
| 32 | 284 | Sports | C |
| 33 | 203 | Sports | F |
| 34 | 292 | Sports | C |
| 35 | 300 | Sports | C |
| 36 | 199 | Sports | F |
| 37 | 249 | Sports | D |
| 38 | 337 | Sports | B |
| 39 | 237 | Sports | F |
| 40 | 339 | Sports | B |
| 41 | 346 | Sports | B |
| 42 | 360 | Sports | A |
| 43 | 303 | Sports | C |
| 44 | 397 | Sports | A |
| 45 | 367 | Sports | A |
| 46 | 370 | Sports | A |
| 47 | 336 | Sports | B |
| 48 | 365 | Sports | A |
| 49 | 373 | Sports | A |
| 50 | 332 | Sports | B |
| 51 | 242 | Sports | D |
| 52 | 236 | Sports | F |
| 53 | 286 | Sports | C |
| 54 | 336 | Sports | B |
| 55 | 302 | Sports | C |
| 56 | 324 | Sports | B |
| 57 | 284 | Sports | C |

That looks like the right kind of thing. I don't know what the Total Points is out of. Looks like it might be 400. (This is actually all of the students; there were 57 of them.)

Somewhere you need to discover that "Total Points" is actually `Total_Points` with uppercase and an underscore in it. (Actually, SAS is not case-sensitive, so the uppercase doesn't matter, but the underscore *does*.)

OK, the mean "total points" for each section. This is `proc means`:

```
proc means;
  var Total_Points;
  class Section;
```

I was careful about Uppercase, but it works just fine without (and so is fine for you).

Results:

| The MEANS Procedure | | | | | | |
|----------------------------------|-----|----|-------------|------------|-------------|-------------|
| Analysis Variable : Total_Points | | | | | | |
| Section | N | | | | | |
| | Obs | N | Mean | Std Dev | Minimum | Maximum |
| Regular | 29 | 29 | 338.0689655 | 30.6045410 | 265.0000000 | 382.0000000 |
| Sports | 28 | 28 | 307.2500000 | 52.5107573 | 199.0000000 | 397.0000000 |

The highest score observed overall is 397, which further supports the maximum possible score being 400.

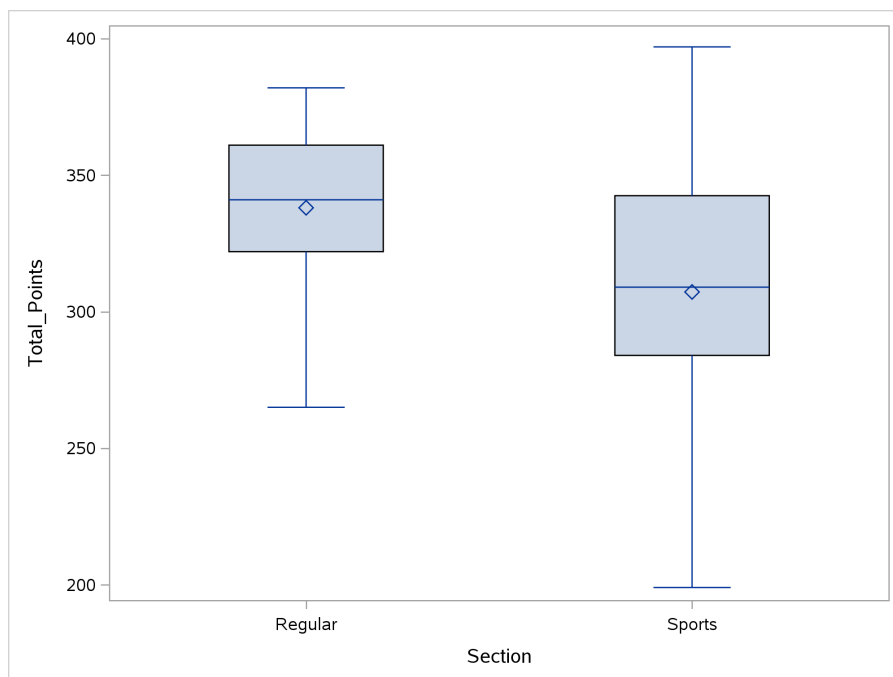
The reason for giving you this is mainly to give you practice at getting the means, but if you like you can comment that the mean score for the **Regular** group seems a fair bit higher than for the **Sports** group. (Then you can compare this conclusion with the one from the boxplot in the next part.)

- (b) (3 marks) Obtain a boxplot of total points for each section. Does one of the sections have a clearly higher average than the other? Explain briefly.

Solution: `proc sgplot`. The syntax needs to be figured out, but once you have it, boxplots in SAS are really all the same:

```
proc sgplot;
  vbox Total_Points / category = Section;
```

Here's the plot:



I would say that the average (mean or median) is clearly higher for the Regular section than for the Sports section, because most of the left-hand boxplot is higher than the right-hand one (or the median and quartiles are all noticeably higher). Have an opinion. If the opinion is “there is not much difference between the two sections because there is a lot of variability”, I’m OK with that too. (An opinion plus a reason for that opinion is what I want to see.)

- (c) (2 marks) What is it about the design of this study that would make you hesitant to say that having only sports examples is a bad idea? Explain briefly.

Solution: The key thing I’m after here is that the students were *not* randomly assigned to sections; the students could choose which section they enrolled in, knowing what kind of examples they would see. It could be that the weaker students chose the section with sports examples because they thought this would be more interesting, for example (in which case if you randomly assigned students to sections, the **Sports** section wouldn’t look so bad).

Another way to argue it is that the **Sports** section was earlier in the day, and so students in that section were sleep-deprived, or were otherwise less capable of taking in the material. This would be another reason, independent of the subject matter, why students would do less well.

I did say that the same instructor taught both sections, so this is not a reason why the sections may have come out differently. (If the instructors had been different, that could have been another reason for the observed difference.)

If the students had been randomly assigned to sections that were *only* different because of the kind of examples, and this kind of difference in mean/median had been observed, *that* would have been a reason to say that the kind of examples makes a difference. That’s why we design experiments the way we do: have only *one* thing be different, and *randomize* subjects to experimental conditions. That did not happen here.

Extra: this part is an argument for not doing any kind of inference, but with 28 and 29 students in each section, we could otherwise justify doing a two-sample t -test (there are no outliers and there is only moderate skewness). If we had done that test, and it had come out significant (which I think it will), it's not at all clear what that would have told us: yes, there is a difference in mean Total Points between the two sections, but that difference could be for any number of different reasons, not only because one kind of examples is better than the other.

9. The “ecological footprint” of a person, city or country is defined by the World Wildlife Fund⁹ 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);
```

| Obs | Country | Region | Eco_ 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 |

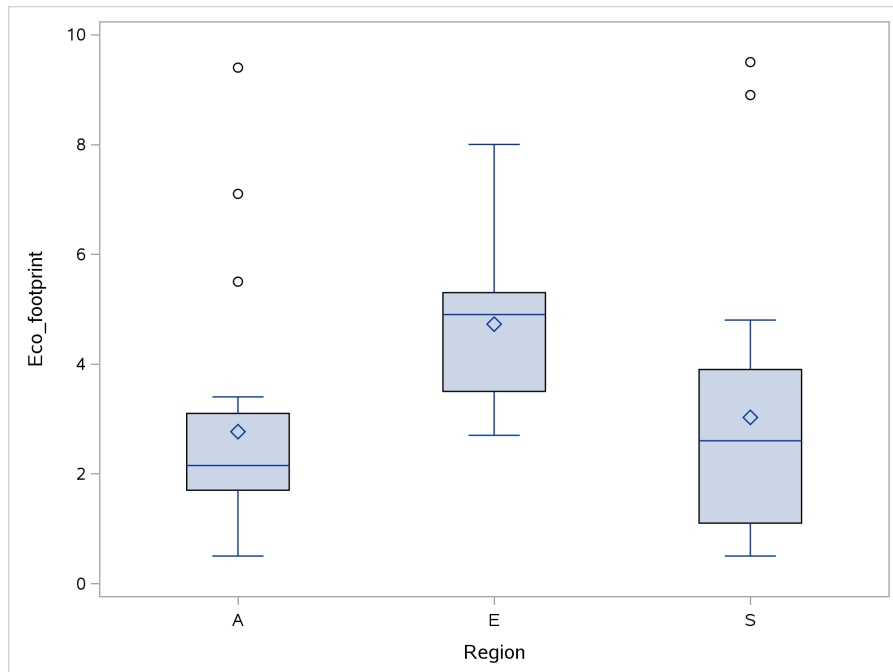
To display only some rows, you need to use the data set name (the one on `out`; whatever you choose is fine, but it has to be the same in both places) and `obs=20` in brackets afterwards.

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.

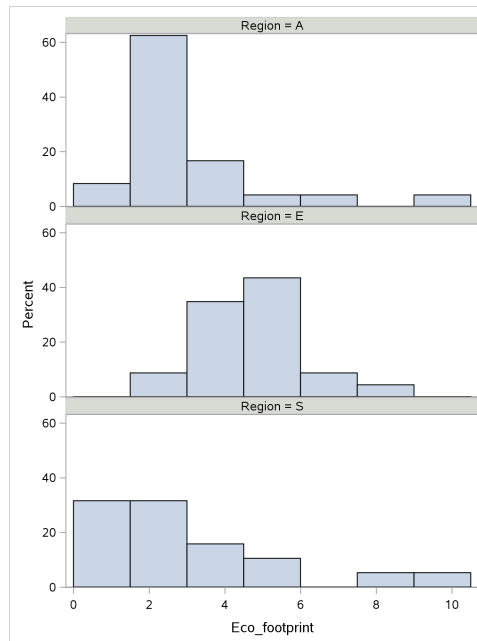
Solution: With a quantitative variable (`Eco_footprint`) and a categorical one `Region`, this suggests a boxplot, which I think is best. I am using lowercase for my variable names, for ease of typing.¹⁰

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



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

| Region | N 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;
```

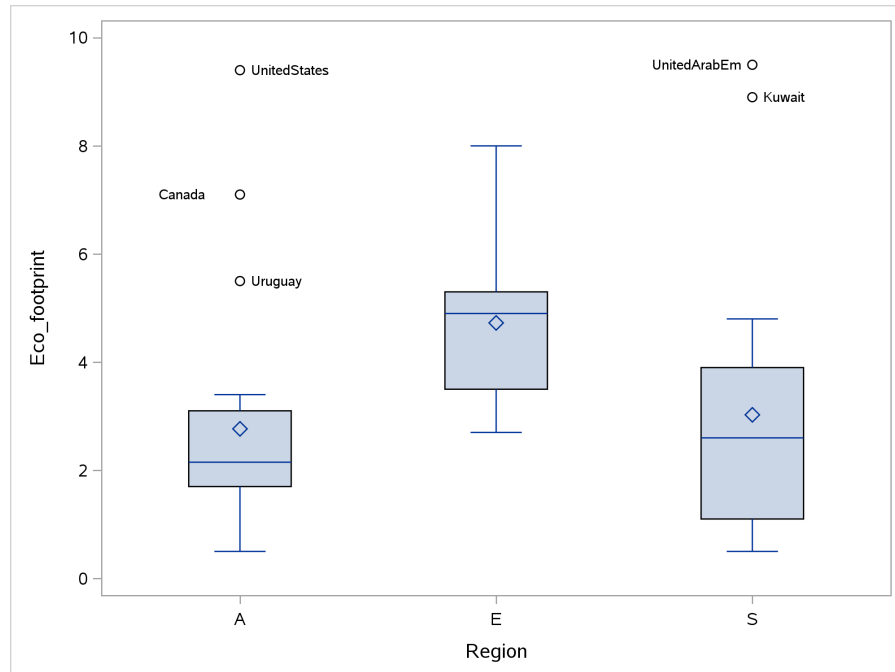
The MEANS Procedure

| Region | N 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.

Notes

¹If you forget it, there's a way to get it back, but it's better not to have to go that way.

²This is the same screen as you bookmarked earlier. You can use your bookmark if you prefer, now or later.

³This is because you are competing against everyone else in the world for access to SAS's servers.

⁴Downloading as PDF looks nice, but you cannot easily add it to another document with your code and explanation.

⁵Which stands for "rich text format"

⁶My guess is more or less correct; the re-writing in C happened in 1985, according to Wikipedia.

⁷The 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.

⁸Before 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.

⁹I 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.

¹⁰When 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.

¹¹Applied Statistics is an art as much as a science at times.

¹²Which was me.