University of Toronto Scarborough
Department of Computer and Mathematical Sciences
STAC32 (K. Butler), Midterm Exam
October 26, 2015

Aids allowed:

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

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

This exam has 32 numbered pages of questions. Please check to see that you have all the pages.

In addition, you should have an additional booklet of output to refer to during the exam. Contact an invigilator if you do not have this.

Answer each question in the space provided (under the question). If you need more space, use the backs of the pages, but be sure to draw the marker's attention to where the rest of the answer may be found.

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

When giving SAS code, it is acceptable to use code that runs on the online version of SAS Studio, or on the version that runs on a virtual machine. Either version is acceptable.

Last name: ______________________________

First name: ______________________________

Student number: ______________________________

For marker's use only:

Page	Points	Score
1	2	
2	2	
5	4	
6	4	
8	6	
9	4	
13	2	
14	6	
16	7	
22	10	
23	2	
24	8	
25	2	
26	6	
27	2	
28	6	
30	11	
Total:	84	

1. Professional baseball in North America is played in two leagues, the American League (in which the Blue Jays play) and the National League (in which the Montreal Expos used to play). Figure 1 in the booklet of code and output shows, for each team, the league they play in and the number of home runs hit in the team's stadium. (The number of home runs is a total for the 2011 season; each team plays 81 home games.) The data have been saved in a file called `hr.txt`, both in the working folder of R Studio and in your home folder on SAS Studio.

   Give suitable code that will accomplish the tasks described below. You should need only between one and three lines of code in each case.

   (a) (2 marks) Read the data into an R data frame.

My answer: 2017 (only the second line needed by you):

```
R> library(tidyverse)
R> homers=read_delim("hr.txt"," ")
R> homers

Loading tidyverse: ggplot2
Loading tidyverse: tibble
Loading tidyverse: tidyr
Loading tidyverse: readr
Loading tidyverse: purrr
Loading tidyverse: dplyr
Conflicts with tidy packages ----------------------------------------------------
filter(): dplyr, stats
lag():    dplyr, stats
Parsed with column specification:
cols(
  league = col_character(),
  homeruns = col_integer()
)
# A tibble: 30 x 2
     league homeruns
      <chr>    <int>
 1 American      122
 2 American      103
 3 American      100
 4 American       96
 5 American       93
 6 American       86
 7 American       84
 8 American       80
 9 American       74
10 American       73
# ... with 20 more rows
```

2015:

```
R> homers2=read.table("hr.txt",header=T)
```

That's all you need, but I wanted to check that it worked:

```
R> homers2
```

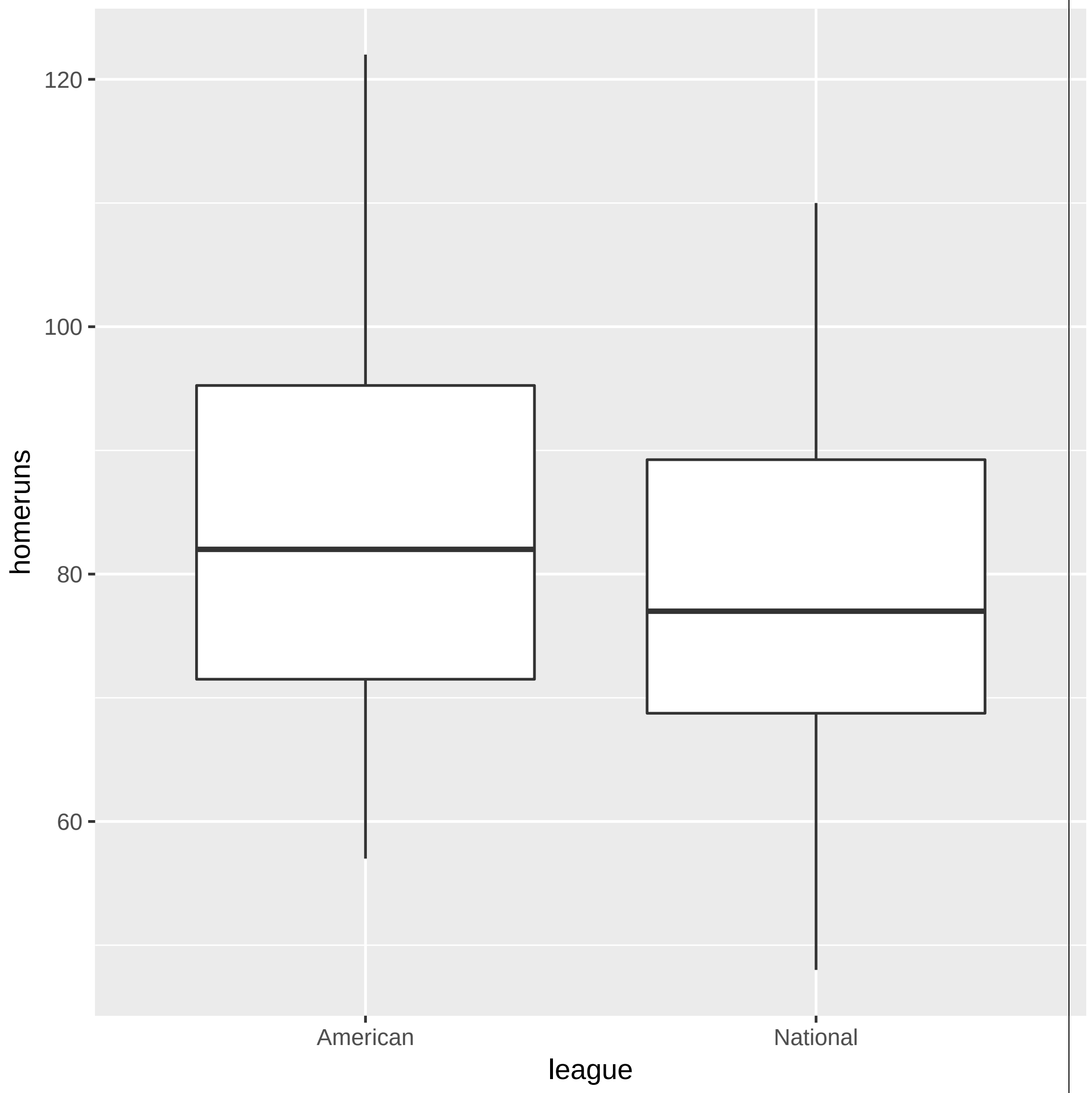
```
     league homeruns
1  American      122
2  American      103
3  American      100
4  American       96
5  American       93
6  American       86
7  American       84
8  American       80
9  American       74
10 American       73
11 American       71
12 American       64
13 American       64
14 American       57
15 National      110
16 National      106
17 National       94
18 National       90
19 National       89
20 National       86
21 National       80
22 National       77
23 National       77
24 National       76
25 National       74
26 National       70
27 National       65
28 National       63
29 National       60
30 National       48
```

It does. It's smart to have the name of your data frame *different* from the names of all the variables in it (otherwise confusing stuff will happen when you `attach` it).

(b) (2 marks) Draw side-by-side boxplots of home runs for each league, in R.

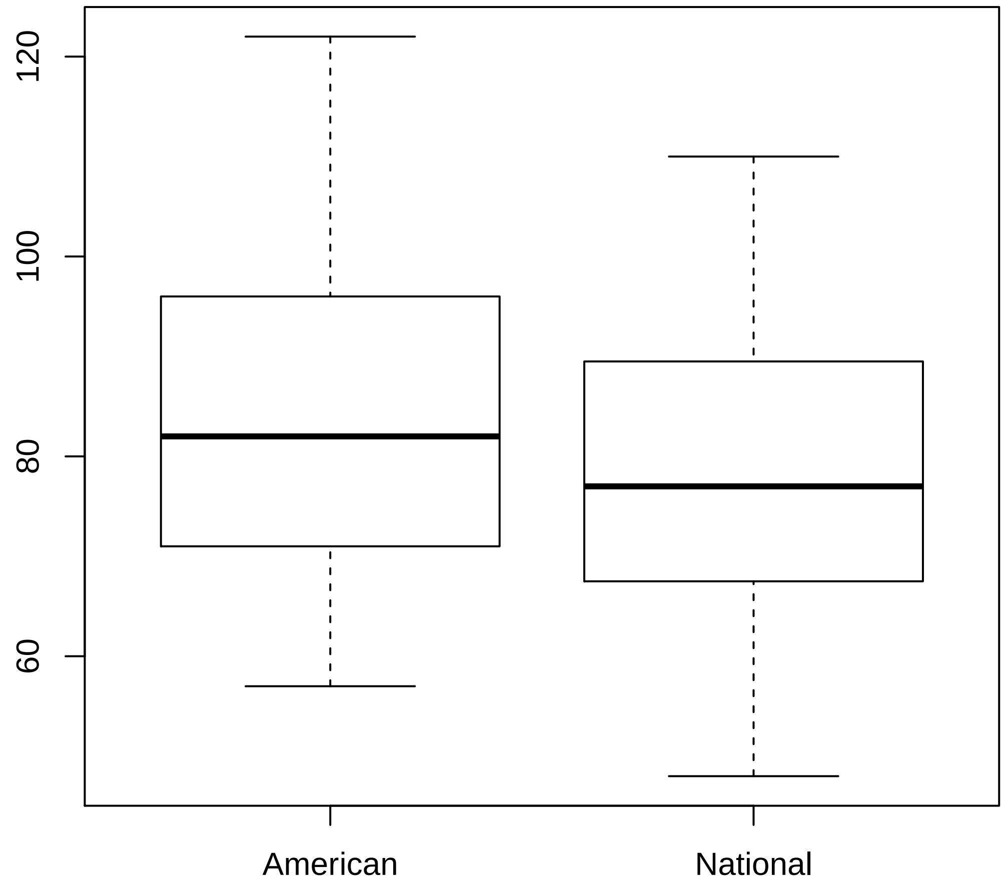
My answer: 2017:

```
R> ggplot(homers,aes(x=league,y=homeruns))+geom_boxplot()
```



2015:

```
R>   attach(homers)
R>   boxplot(homeruns~league)
R>   detach(homers)
```



I detached the data frame afterwards, to be tidy (and because I don't need it attached any more). You can do this, or not. It doesn't matter.

You need the `attach` or else R won't know what `leagues` is. `boxplot` is one of those functions that accepts a `data=`, so this also works, without `attach` (for full credit):

```
R>    boxplot(homeruns~league,data=homers)
```

Baseball fans among you may note that there tend to be more home runs at American League stadiums, and you may care to hypothesize that has something to do with the Designated Hitter rule.

For the rest of you: in the National League, the pitcher also has to bat. Pitchers tend not to be very good batters, because pitching (throwing) and batting (hitting) are very different skills. In the American league, the pitcher does not bat; in his place there is a player called the Designated Hitter, whose only job is to bat: he doesn't even field. This means that the overall level of batting ought to be a bit higher in the American league, and there should be a few more home runs on average than in the American league. As indeed there were in 2011.

The focus on stadiums is because, in a game between an American League and a National League team, the rules are according to the league of the *home* team. Thus, in an American League team's stadium, there is always a designated hitter (for both teams), and in a National League team's stadium, the pitchers have to bat (for both teams). This only used to be an issue in the World Series, but now there are inter-league games during the season, and it applies then as well.

(c) (2 marks) Calculate the mean number of home runs for each of the two leagues, in R.

My answer: 2017: `group_by` and `summarize`:

```
R> homers %>% group_by(league) %>%
R>   summarize(m=mean(homeruns))

# A tibble: 2 x 2
    league        m
     <chr>    <dbl>
1 American 83.35714
2 National 79.06250
```

2015: This calls for `aggregate`. You need to supply all the things: a "model formula" with a squiggle, a data frame, and a thing to calculate, in that order:

```
R> aggregate(homeruns~league,homers,mean)

    league homeruns
1 American 83.35714
2 National 79.06250
```

Once again, the American league comes out a little higher. (There were no outliers and no substantial skewness, so we'd expect the medians and means to tell the same story.)

(d) (2 marks) Read the data into a SAS data set. (For this part, assume that the top line of the data file shown in Figure 1 has been omitted. This is to make your job easier. The solution where that top line has been included will also be considered correct.)

My answer: 2015: This kind of thing, with your choice of names for everything:

```
SAS> data homers2;
SAS>   infile '/home/ken/hr.txt' firstobs=2;
SAS>   input league $ homeruns;
```

2017: `proc import`, ignoring the parenthetical remark:

```
SAS> proc import
SAS>   datafile='/home/ken/hr.txt'
SAS>   out=homers
SAS>   dbms=dlm
SAS>   replace;
SAS>   getnames=yes;
SAS>   delimiter=' ';
```

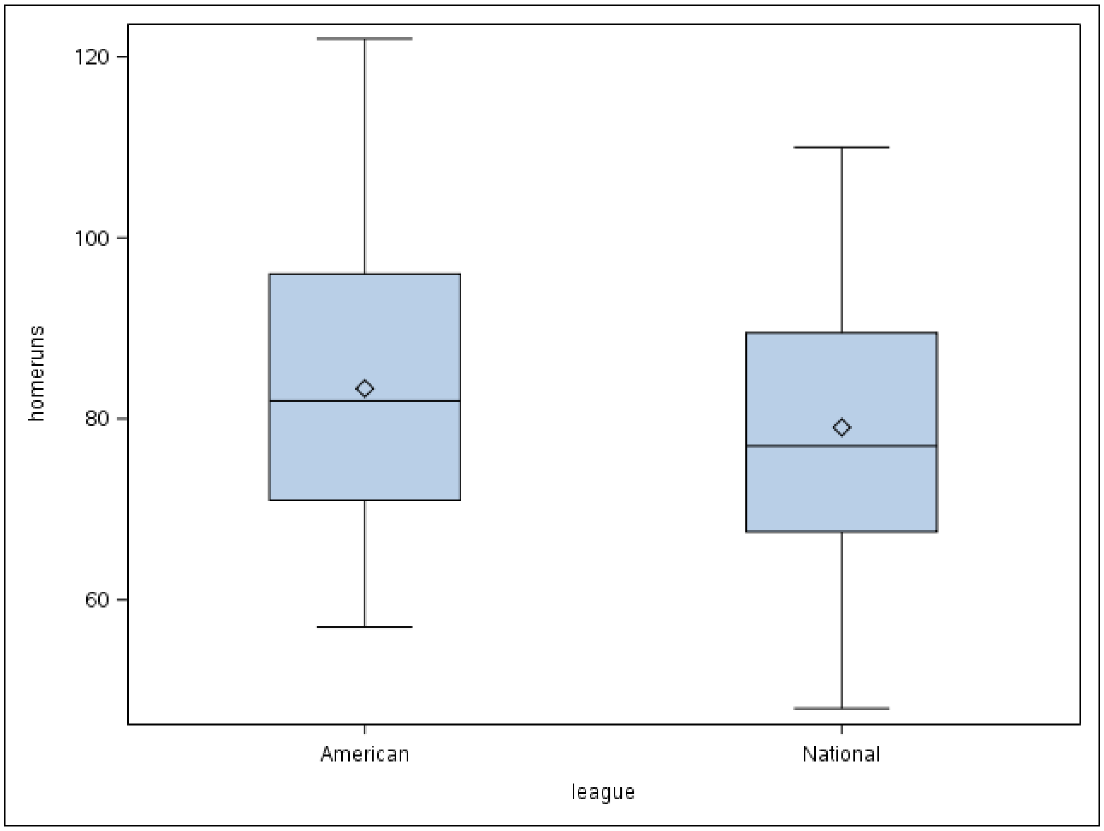
since the data values are separated by a single space.

You need to put the `$` after `league`, because that variable is text rather than numbers. The `firstobs` is optional *for this question*, because you might have copied to SAS Studio the whole data file, or all of it except for the first line. This question is meant to be a nice gentle warmup, so I didn't want you to get hung up on that detail (as I tried to state in the question).

(e) (2 marks) Draw side-by-side boxplots of home runs for each league, in SAS.

My answer:

```
SAS> proc sgplot;
SAS>   vbox homeruns / category=league;
```



(f) (2 marks) Calculate the mean number of home runs for each of the two leagues, in SAS.

My answer: `proc means`:

```
SAS> proc means;
SAS>   var homeruns;
SAS>   class league;
```

```
The MEANS Procedure
                              Analysis Variable : homeruns
               N
league       Obs     N              Mean          Std Dev          Minimum          Maximum
```

```
-------------------------------------------------------------------------------------
American     14    14      83.3571429      18.0154085      57.0000000     122.0000000
National     16    16      79.0625000      16.4983585      48.0000000     110.0000000
-------------------------------------------------------------------------------------
```

`var homeruns` is actually optional here, because there is only one non-`class` variable. (If there had been more than one, leaving out the `var homeruns` would have given you means for all of them, separately by league.)

2. The Finger Lakes is a district of New York State famous for producing wines. There are several different vineyards along three of the lakes. For each vineyard, the selling price of a case of wine is recorded, and each vineyard is classified by which lake it is on (labelled `location` in the data set). SAS boxplots are shown in Figure 2 in the booklet of code and output.

   (a) (2 marks) Describe how the locations differ in terms of average case price, if at all. (You will have to decide what you mean by "average".)

   My answer: You first need to decide what "average" means in this context: median or mean. Either is good, but you should decide. The story is the same both ways: Keuka has clearly the largest mean (median), while the other two locations are similar, or Cayuga is slightly bigger (either is good).

   (b) (2 marks) Describe how the locations differ in terms of inter-quartile range, if at all.

   My answer: Inter-quartile range is specifically the height of the box, so don't consider the whiskers. Once again, Keuka is different from the other two, in having a much smaller IQR than the other two, which are again similar. Visually, I cannot detect a difference in IQR between these two.

   (c) (2 marks) For the location that has the smallest inter-quartile range, how might the *standard deviation* be a misleading measure of spread? Explain briefly. (If you found more than one location that has the smallest IQR, pick any one of them, say which one you picked, and then answer the question.)

   My answer: Keuka has the smallest IQR, and also has a big (low) outlier, so that the standard deviation for this group would be a lot larger than the IQR would suggest. (You cannot compare SD and IQR directly, so you have to be a bit careful with the wording.) This is misleading, because the SD is large *only* because this observation is so small: if the observations were not there, the SD would be a lot smaller.

   You need to note the outlier, but you also need to say something about its affect on the SD. Simply saying something like "remove the outlier" does not answer the question.

3. A study was carried out on boys with attention deficit hyperactivity disorder (ADHD). The researchers rated boys' performance on a number of tasks. One task was "has difficulty organizing work", and each boy was rated on a 0–4 scale, with 0 meaning "has no difficulty" and 4 meaning "has a lot of difficulty". 282 boys with ADHD were rated. *Only* the integer values $0, 1, 2, 3, 4$ were used for ratings; there were no decimal ratings. Some output is shown in Figures 3 and 4. You may refer to the task as "this task" as you answer the questions below, rather than writing out "has difficulty organizing work" every time.

   (a) (2 marks) Figure 3 shows some of the data, and a table of how many boys received each rating. Explain briefly what is assumed about the data in order to use a t-test (or t confidence interval) and why that assumption cannot be satisfied exactly here.

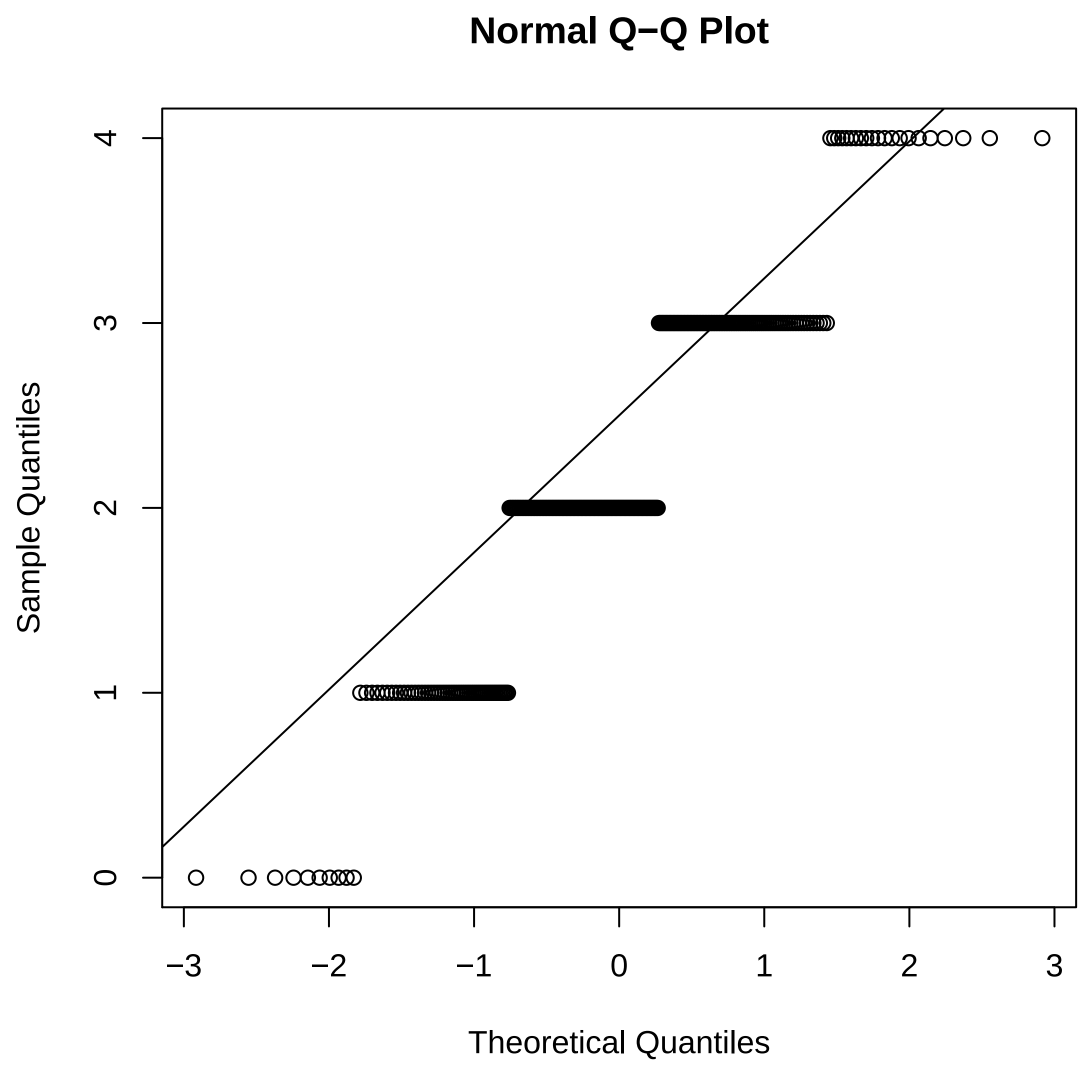
   My answer: The data are supposed to come from a normal distribution, but the distribution of ratings is discrete (there are only five possible values), so it cannot be normal (which is continuous).

   (b) (2 marks) Despite what you said in part (a), explain briefly why you would have no real hesitations about using a t-test (or t confidence interval) here.

   My answer: The distribution appears normal-like with only a little skewness and no outliers (by the nature of the rating scale). With such a big sample, 282 boys, I would have no problems using t-distribution inference here. (I would expect the Central Limit Theorem to have "kicked in" well before a sample size of 282.)

   I was curious about what a normal quantile plot would look like for these data, so I drew one:

```
R> qqnorm(boys)
R> qqline(boys)
```



The discreteness of the data shows up in the large horizontal patches of points. That much is unavoidable. The relevant question is really "are those patches of points near the line?". I'd say they more or less are; the only real problem is at the bottom, where the 0's and 1's are a bit lower than you'd expect with normality, but overall I'd say that this is pretty good.

The other thing we have going for us is the Central Limit Theorem. We have a large sample of $n = 282$ boys. Remember that what makes the t-procedures work well is not the actual population distribution, but the *sampling distribution of the sample mean*, which is more normal when the sample size is larger.

Now, we don't have the population, so we can't assess that directly, but we can borrow an idea called the "bootstrap" and take samples of size 282 from our data *with replacement* (so that they come out different every time), and take the sample mean, like this:

```
R> z=sample(boys,282,replace=T)
R> mean(z)
```

```
[1] 2.390071
```

Then we put that in a function, which can have as input the data and the sample size:

```
R> bootstrap.mean=function(mydata,n) {
R>   z=sample(mydata,n,replace=T)
R>   return(mean(z))
R> }
```

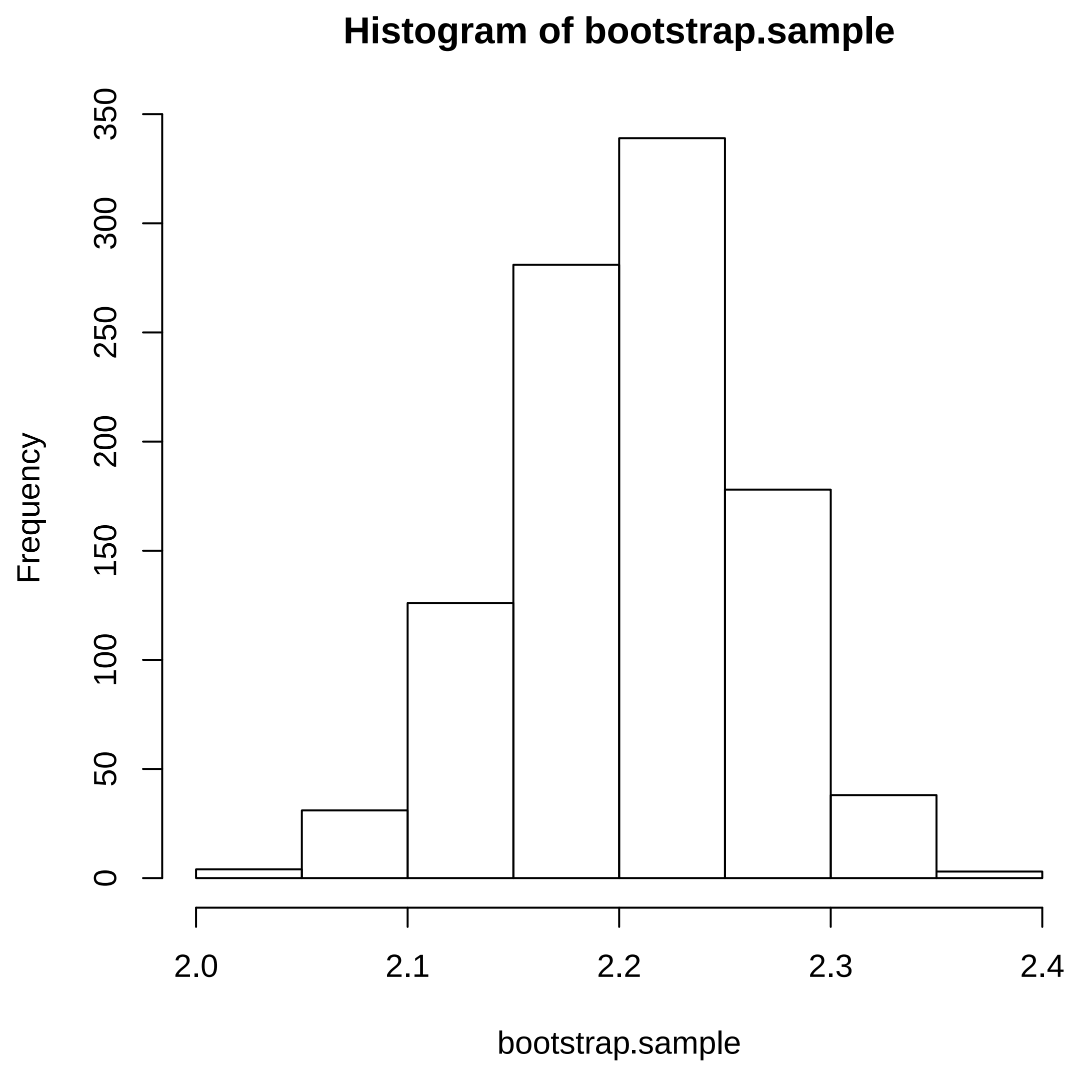
Try it a couple of times:

```
R> bootstrap.mean(boys,282)
R> bootstrap.mean(boys,282)
```

```
[1] 2.042553
[1] 2.365248
```

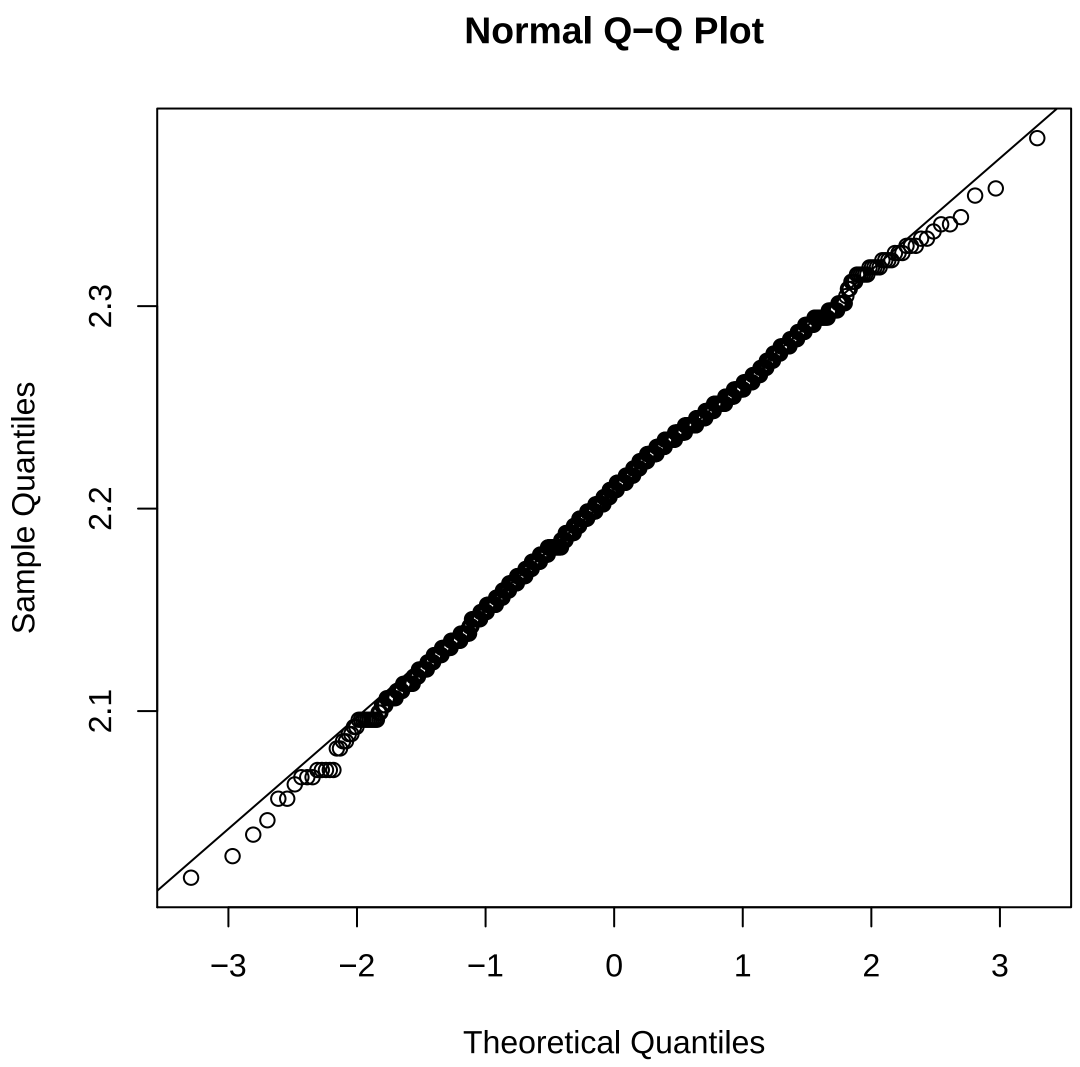
OK so far. Then, just as we do with a randomization test, we use `replicate` to run it a bunch of times:

```
R> bootstrap.sample=replicate(1000,bootstrap.mean(boys,282))
R> hist(bootstrap.sample)
```



Well, that looks pretty normal:

```
R>   qqnorm(bootstrap.sample)
R>   qqline(bootstrap.sample)
```



Oh yeah. Even the discreteness doesn't show up much when you take the mean of 282 observations (there are a lot of possible means with a sample that big, so the distribution is "effectively" continuous).

I have no doubts whatever about the *t*-procedures now.

(c) (2 marks) The output in Figure 4 contains a confidence interval. Explain *precisely* what that confidence interval means.

My answer: The precise answer is this: The procedure by which that interval was calculated will produce an interval containing the population mean (mean rating on this task for *all* ADHD boys) in 90% of all possible samples.

The answer "the interval produced by this method has probability 0.90 of containing the population mean" is true, but not precise enough, because you need to say that the probability is over all possible samples, and not just over the one sample that you happened to get.

An answer like "the interval from 2.12 to 2.30 has probability 0.90 of containing the population mean" is *wrong*, because the interval either *does* contain the population mean or it *doesn't*. (You need to be a Bayesian statistician obtaining a "90% credible interval" from the posterior distribution; if you did *that*, then you can make the statement, but not otherwise.)

(d) (3 marks) Figure 4 also contains a hypothesis test. Give the null and alternative hypotheses for this test, defining any symbols that you use. Also, what do you conclude in the context of the data?

My answer: Let μ be the mean rating on this task for *all* ADHD boys. (That is, the population mean that is being estimated here.) The null hypothesis is $H_0 : \mu = 2.35$, and the alternative hypothesis is $H_a : \mu \neq 2.35$ (two-sided since I didn't say otherwise in the code).

At $\alpha = 0.05$ (you should say or imply what α you are using), the P-value is *less* than α, so we can reject the null hypothesis in favour of the alternative: that is, we conclude that the mean rating on this task is *not* 2.35 for all boys with ADHD. (If you chose $\alpha = 0.01$, which is fine, you *cannot* reject the null, and then you have to say "there is no evidence that the mean score on this task for all boys with ADHD differs from 2.35".)

(e) (3 marks) R obtained a P-value for the test in Figure 4. Describe the process by which you would obtain a P-value for this test using either statistical tables or an R function *other* than `t.test`. I don't want a numerical answer, just a description of the process. Use the fact that the t-distribution with large degrees of freedom can be closely approximated by a standard normal distribution. Your normal table, if you go that way, gives the probability of observing a value *less* than the given z.

My answer: The strategy is to find the probability of a value as extreme or more extreme than the one observed. The observed value is the t statistic of -2.4834 (on the P-value line of the output).

Your normal table gives you the probability of obtaining a value less than -2.48, which is what you want, since less is more extreme. My table gives 0.0066. The test was two-sided, though, so I have to double that, giving 0.0132. (You won't have numbers, but you need to say "find the probability of less than -2.48 and then double it". My value is very close to the exact 0.0136 that came out of `t.test`.)

With R, you use `pnorm`, and the probability of a value less than -2.4834 is this:

```
R> prob=pnorm(-2.4834)
R> prob

[1] 0.006506743
```

and again multiply by 2 to get the two-sided P-value:

```
R> prob*2

[1] 0.01301349
```

which is basically the same as `t.test`. The answer using the normal distribution is in each case off by a bit, but not nearly enough to make us change our conclusion.

If you're wondering, `pnorm` will handle any normal distribution; you give the mean and SD (if they are not 0 and 1) *afterwards*, thus the probability of a value less than 40 in normal distribution with mean 50 and SD 10 is

```
R> pnorm(40,50,10)
```

```
[1] 0.1586553
```

or you could have turned 40 into $z = (40 - 50)/10 = -1$ and done:

```
R> pnorm(-1)

[1] 0.1586553
```

to get the same answer. This is the 16% that would have come out of the 68–95–99.7 rule.

4. I have a spreadsheet that contains names and information about different types of coffee drinks served in a certain coffee shop. Some of the spreadsheet is shown in Figure 5 of the booklet of code and output. I want to read this information into SAS.

   (a) (4 marks) Tell me how to get the data in the spreadsheet into SAS Studio so that (in the next part) it can be read into a SAS data set. You need to provide enough detail so that I can reproduce your process myself and end up with some kind of file in SAS Studio. (You can assume that I know how to do basic spreadsheet operations.)

   My answer: 2017: my current preferred method is to *upload* the `.xlsx` spreadsheet file from my computer to SAS Studio, and then to use the `proc import` at the bottom of my answer to (b). An alternative is to save the file as `.csv`, then upload it to SAS Studio, then read in with `dbms=csv`. (The implication is that the spreadsheet is on your computer, or mine, currently.)

   2015: There are several steps in what I think is the best solution:

   - Save the data from the spreadsheet in `.csv` format.
   - Open the `.csv` file in something like Notepad (or even R Studio, strange though this may seem: the point is that you need something that will *display* the file as it is, which spreadsheet software won't).
   - Copy the data *from there* into SAS Studio.
   - Save and rename the file as usual. (An alternative to these steps is to *upload* the `.csv` file to SAS Studio. That avoids having to rename it.)

   See the solution to the next part for more discussion, including some things that won't work and other ideas that will work. If you come up with anything that will work (as well as this), you'll get full credit. (The two parts of this question work together: if your whole process produces a suitable SAS data set, that's good: you can rescue what I think is a sub-optimal (a) by an excellent (b).)

   (b) (3 marks) Give code that will read the file that you saved on SAS Studio into a SAS data set, preserving the names of the coffee drinks as well as possible. Depending on what you did in the previous part, your code might be quite simple.

   My answer: 2017: use the `proc import` described at the end, or the corresponding thing for reading in a `.csv` file if that's what you made.

   2015: This reads the `.csv` file that I created in the last part:

```
SAS> data coffees;
SAS>   infile '/home/ken/starbucks.csv' firstobs=2 dlm=',';
SAS>   input product $ calories fat carbs fibre protein;
```

   To verify that this works at least reasonably well:

```
SAS> proc print;
```

```
Obs    product     calories     fat    carbs    fibre    protein
  1    Caffe La       190       7.0      18       0         12
  2    Caffe Mo       260       8.0      41       2         13
  3    Cappucci       120       4.0      12       0          8
  4    Caramel        240       7.0      34       0         10
  5    Cinnamon       260       6.0      40       0         11
  6    Flavoure       250       6.0      36       0         12
  7    Iced Caf       130       4.5      13       0          8
  8    Iced Caf       200       6.0      35       2          9
  9    Iced Car       230       6.0      33       0         10
 10    Iced Cin       200       4.0      34       0          7
 11    Iced Fla       250       6.0      36       0         12
 12    Iced Pep       260       6.0      52       2          8
 13    Iced Pep       400       9.0      72       0         10
 14    Iced Pum       250       4.0      44       0         10
 15    Iced Ski       110       4.0      12       0          7
 16    Iced Tof       280       3.5      51       2         12
 17    Iced Whi       340       9.0      55       0         10
 18    Peppermi       330       8.0      57       2         12
 19    Peppermi       470      12.0      78       0         14
 20    Pumpkin        310       6.0      49       0         14
 21    Skinny C       180       6.0      18       0         12
 22    Skinny F       180       6.0      18       0         12
 23    Toffee M       350       7.0      58       2         17
 24    White Ch       400      11.0      61       0         15
```

Well, that went as well as could be expected, didn't it? (A quote from `http://www.dailymotion.com/video/x1szu1a_wallace-and-gromit-in-the-wrong-trousers_shortfilms`. Watch from about 1:37 on for the context, such as it is.)

We only get the first 8 characters of the product names. This is the best we can do until we learn about "informats" (much later). So I'm not expecting you to do any better than I did.

This works sufficiently well to get the mean `protein` value:

```
SAS> proc means;
SAS>   var protein;

The MEANS Procedure
                  Analysis Variable : protein
 N           Mean         Std Dev         Minimum         Maximum
-----------------------------------------------------------------
24      11.0416667       2.5448869       7.0000000      17.0000000
-----------------------------------------------------------------
```

The `/folders/myfolders/` thing is also good, since you could be running SAS on a virtual machine.

Some things that won't work:

- Copying the data directly from the spreadsheet into SAS Studio (unless you are very careful: see below.)
- Using `expandtabs`. That won't work *in this case* because the names of the drinks have spaces *in* them, and then SAS won't be able to distinguish between the spaces within a drink name and the spaces that separate one column from the next. (It will work, in that the SAS code will run, but you will get an unholy mess with things all in the wrong columns. See below.)

If you copy and paste directly from the spreadsheet, you *can* make it work, but you have to get the code right. The values get copied separated by *tabs*, and you have to specify that this is what the values are separated by, using `dlm`. The problem is that you can't type a tab character as you can type a comma; the way around it is to know or find out that the tab character has (ASCII) hex code 09, so that this works:

```
SAS> data coffees;
SAS>   infile '/home/ken/starbucks2.txt' firstobs=2 dlm='09'x;
SAS>   input product $ calories fat carbs fibre protein;
SAS>
SAS> proc print;

Obs    product     calories     fat    carbs    fibre    protein
  1    Caffe La       190       7.0      18       0         12
  2    Caffe Mo       260       8.0      41       2         13
  3    Cappucci       120       4.0      12       0          8
  4    Caramel        240       7.0      34       0         10
  5    Cinnamon       260       6.0      40       0         11
  6    Flavoure       250       6.0      36       0         12
  7    Iced Caf       130       4.5      13       0          8
  8    Iced Caf       200       6.0      35       2          9
  9    Iced Car       230       6.0      33       0         10
 10    Iced Cin       200       4.0      34       0          7
 11    Iced Fla       250       6.0      36       0         12
 12    Iced Pep       260       6.0      52       2          8
 13    Iced Pep       400       9.0      72       0         10
 14    Iced Pum       250       4.0      44       0         10
 15    Iced Ski       110       4.0      12       0          7
 16    Iced Tof       280       3.5      51       2         12
 17    Iced Whi       340       9.0      55       0         10
 18    Peppermi       330       8.0      57       2         12
 19    Peppermi       470      12.0      78       0         14
 20    Pumpkin        310       6.0      49       0         14
 21    Skinny C       180       6.0      18       0         12
 22    Skinny F       180       6.0      18       0         12
 23    Toffee M       350       7.0      58       2         17
 24    White Ch       400      11.0      61       0         15
```

To show you that `expandtabs` won't work, let's start with `starbucks2.txt`, which came from copying and pasting the spreadsheet data:

```
SAS> data ugh;
SAS>   infile '/home/ken/starbucks2.txt' firstobs=2 expandtabs;
SAS>   input product $ calories fat carbs fibre protein;
SAS>
SAS> proc print;

Obs    product     calories    fat    carbs    fibre    protein
  1    Caffe           .       190      7      18.0        0
  2    Caffe           .       260      8      41.0        2
  3    Cappucci       120        4     12       0.0        8
  4    Caramel         .       240      7      34.0        0
  5    Cinnamon        .         .    260       6.0       40
  6    Flavoure        .       250      6      36.0        0
  7    Iced            .         .    130       4.5       13
  8    Iced              .        .      200      6.0       35
  9    Iced              .        .      230      6.0       33
 10    Iced              .        .        .    200.0        4
 11    Iced              .        .      250      6.0       36
 12    Iced              .        .      260      6.0       52
 13    Iced              .        .        .        .      400
 14    Iced              .        .        .    250.0        4
 15    Iced              .        .        .    110.0        4
 16    Iced              .        .      280      3.5       51
 17    Iced              .        .        .    340.0        9
 18    Peppermi          .      330        8     57.0        2
 19    Peppermi          .        .        .    470.0       12
 20    Pumpkin           .        .      310      6.0       49
 21    Skinny            .        .        .    180.0        6
 22    Skinny            .        .      180      6.0       18
 23    Toffee            .      350        7     58.0        2
 24    White             .        .      400     11.0       61
```

What has happened is that SAS is trying to use one or more spaces to separate one variable from the next. When the name of the drink has no spaces in it, everything is good (line 3, the name is just "Cappuccino"). But when the name of the drink is two words, SAS uses the second word as the value for `Calories`, which fails since that should be a number, then uses the number for `Calories` as the value for `Fat`, and so on, off by one all the way down the line. If the name of the drink has *three* words, there are two words that can't be numbers, and then everything is off by *two* columns from there on. And so on. Ugly.

A third way around this is to use `proc import`, which is described in the SAS text, pages 65–67 (in my edition). I don't expect you to know this, but if you have discovered it and can properly describe how to make it work, you get full credit.

The first thing is to upload the Excel version of the SAS spreadsheet to SAS Studio. (You have to upload it, rather than copying and pasting, because the internal format of an Excel spreadsheet contains all sorts of special characters that won't properly copy and paste.) I uploaded the spreadsheet to `starbucks.xls` (it was an ancient spreadsheet). Then:

```
SAS> proc import out=coffees datafile='/home/ken/starbucks.xls' dbms=xls replace;
SAS>   sheet="Sheet1";
SAS>   getnames=yes;
SAS>
SAS> proc print;
```

```
Obs    Product                                      Calories
  1    Caffe Latte                                       190
  2    Caffe Mocha                                       260
  3    Cappuccino                                        120
  4    Caramel Macchiato                                 240
  5    Cinnamon Dolce Latte                              260
  6    Flavoured Latte                                   250
  7    Iced Caffe Latte                                  130
  8    Iced Caffe Mocha                                  200
  9    Iced Caramel Macchiato                            230
 10    Iced Cinnamon Dolce Latte                         200
 11    Iced Flavoured Latte                              250
 12    Iced Peppermint Mocha                             260
 13    Iced Peppermint White Chocolate Mocha             400
 14    Iced Pumpkin Spice Latte                          250
 15    Iced Skinny Flavoured Latte                        110
 16    Iced Toffee Mocha                                  280
 17    Iced White Chocolate Mocha                         340
 18    Peppermint Mocha                                   330
 19    Peppermint White Chocolate Mocha                   470
 20    Pumpkin Spice Latte                                310
 21    Skinny Cinnamon Dolce Latte                        180
 22    Skinny Flavoured Latte                             180
 23    Toffee Mocha                                       350
 24    White Chocolate Mocha                              400
Obs           Fat          Carbs          Fibre        Protein
  1             7             18              0             12
  2             8             41              2             13
  3             4             12              0              8
  4             7             34              0             10
  5             6             40              0             11
  6             6             36              0             12
  7           4.5             13              0              8
  8             6             35              2              9
  9             6             33              0             10
 10             4             34              0              7
 11             6             36              0             12
 12             6             52              2              8
 13             9             72              0             10
 14             4             44              0             10
 15             4             12              0              7
 16           3.5             51              2             12
 17             9             55              0             10
 18             8             57              2             12
 19            12             78              0             14
 20             6             49              0             14
 21             6             18              0             12
 22             6             18              0             12
 23             7             58              2             17
 24            11             61              0             15
```

That works, and has the additional benefit of reading the *full* product names, rather than chopping them off after 8 characters. (I'm now thinking I should have shown you this in class.) The stuff in the code is:

- `out=`: the name of the SAS dataset, that would otherwise go on the `data` line
- `datafile=`: where the spreadsheet is on SAS Studio, specified the same way as any other file
- `dbms`: the kind of Excel file. If you have an `.xls` file (from an older edition of Excel), replace `xlsx` by `xls` throughout.
- `replace`: replace any SAS data set of the same name by this one (which you usually want to do, rather than, say, appending to the end)
- `sheet=`: which sheet you want in the workbook, by name. I didn't change the name, so mine is the default `Sheet1`.
- `getnames=yes`: top row is variable names (like `header=T` in R).

My edition of the text offers `dbms=excel`, which no longer works, but I will also accept that. (It dates, I think, from before the `.xlsx` format existed.)

If you do it this way, you won't have to do much in (a), but you'll have some extra work in (b). If it's all good, you get all the points.

5. A student suspected that files might take longer to download at certain times of the day. To examine this, he placed a file on a server, and then, at certain times of the day on randomly chosen days, he downloaded the file and noted how many seconds it took. The times of day were 7:00am (denoted "early" in the data set), 5:00pm ("evening") and midnight ("late"). The data are shown in the booklet of code and output as Figure 6. Some analysis is shown in Figure 7 and Figure 8.

   (a) (1 mark) In Figure 7, what *null hypothesis* is being tested?

   My answer: The null hypothesis is that the mean download time is the same for all the different times of day. (Or, that time of day has no impact on download time. I like this second one because it makes ANOVA look like regression: "does the time of day help in predicting download time?". I always liked regression better than ANOVA.)

   Or you can do it with symbols, as long as you define them, eg. let μ_1 be the population mean early download time, μ_2 be the population mean evening download time, and let μ_3 be the population mean late download time. Then the null hypothesis is $H_0 : \mu_1 = \mu_2 = \mu_3$. But you have to define your symbols.

   (b) (1 mark) In Figure 7, what *alternative hypothesis* is being tested?

   My answer: The mean download times at different times of day *are not all the same*. Or that time of day has *some* effect on download time. You have to be careful about this: "the mean download times are different" is not clear enough, because do you mean that they are *all* different, or that some of them are different? (Two the same and one different belongs to the alternative hypothesis.)

   It's difficult to write the alternative hypothesis here with symbols. "It is not true that $\mu_1 = \mu_2 = \mu_3$" is about the best I can do.

   (c) (2 marks) What do you conclude from the analysis of Figure 7?

   My answer: The P-value is less than 0.05 (or 0.01 or whatever α you used), so that we reject the null hypothesis and conclude that the mean download times are not all the same.

   This is as far as we can go, until we look at Tukey.

   (d) (2 marks) Why is the analysis in Figure 8 worth doing? Explain briefly.

   My answer: This is Tukey's method. We rejected the null hypothesis, so we know that there are some differences between times of day to find. Tukey's method will help us figure out which times of day differ from which in terms of mean download times.

   (e) (2 marks) What do you conclude from Figure 8?

   My answer: All three Tukey P-values are less than 0.05 (one of them is even zero to the accuracy shown). So *now* we can conclude that all three times of day have different mean download times.

   (f) (2 marks) Which time of day has quickest downloads, on average? Is it significantly quicker than all the other times of day? (Note that a *small* time goes with a quick download.)

   My answer: This requires a bit of detective work with the Tukey output. The `Difference` column is the difference between sample means for the groups being compared. Thus, evening is slower than early, late is slower than early, and late is quicker than evening. Early is quicker than everything else.

Because all three times of day differ significantly, `early` must be *significantly* quicker than everything else.

(g) (2 marks) Look at Figure 9 of the booklet of code and output, which shows the distributions of download times at each different time of day. Concerning your conclusions above, do you have (i) no doubts, (ii) moderate doubts or (iii) severe doubts about their validity? Explain briefly.

My answer: This is asking about the assumptions underlying the analysis of variance that we just did. The assumptions are that we have normally-distributed data with equal spreads in the three groups. (Like the t-tests, ANOVA can handle moderate departures from these assumptions).

I'd say that the boxplots indicate roughly equal spreads (the IQRs or box heights are more or less equal), and the evening and late times have symmetric distributions, but the early download times are noticeably right-skewed. There are, however, no outliers.

I actually think the ANOVA may be able to handle this amount of skewness (with 16 observations in each group), but probably the best response is to have moderate doubts. Not severe doubts, because the assumptions are not *seriously* violated.

6. In Question 5, we looked at an analysis of download times for a file downloaded at different times of day. Figures 7 and 8 showed the output from the analyses, but the code was not shown. Your job in this question is to give the code. You may assume that the data in Figure 6 have already been read into a data frame `dl`.

   (a) (4 marks) What two or three lines of R code will produce the output in Figure 7?

My answer: I have to read in the data first, so that I can show you that my code works, but you don't need to do that. The two lines below that, or equivalent to them, are what you need:

```
R>   dl=read_delim("downloads.txt"," ")
R>   dl.1=aov(seconds~timeofday,data=dl)
R>   summary(dl.1)

Parsed with column specification:
cols(
  timeofday = col_character(),
  seconds = col_integer()
)
            Df Sum Sq Mean Sq F value   Pr(>F)
timeofday    2 204641  102320   46.03 1.31e-11 ***
Residuals   45 100020    2223
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

   (b) (2 marks) What one or two lines of R code will produce the output in Figure 8?

My answer: One line (to produce the Tukey) is all you need:

```
R> TukeyHSD(dl.1)

  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = seconds ~ timeofday, data = dl)

$timeofday
                  diff        lwr       upr    p adj
evening-early 159.9375  119.53988 200.33512 0.00e+00
late-early     79.6875   39.28988 120.08512 5.57e-05
late-evening  -80.2500 -120.64762 -39.85238 4.99e-05
```

That's all, but you need to get the capitalization in the right places!

7. Arsenic is toxic to humans. People can be exposed to it through contaminated drinking water, food, dust and soil. A new way of examining a person's exposure to arsenic is to examine their toenail clippings. Levels of arsenic, in parts per million, measured from the toenail clippings of 19 people in New Hampshire, are shown in Figure 10 in the booklet of code and output. A boxplot of the arsenic levels is shown in Figure 11.

   (a) (2 marks) Why would a sign test be better here than a t-test here as a test of "centre" or "location"? Explain briefly but precisely. (I am looking for *two* points.)

My answer: The boxplot shows noticeable skewness and two big outliers, so that the median would be a better description of centre or location than the mean would. (One thing.) The sign test does inference for the median, and therefore is better here than the t-test, which does inference for the mean (second thing).

The two crucial things are that the median is better than the mean, and the sign test is the one that tests the median.

(b) (2 marks) I read the data into a SAS data set containing a variable named `arsenic`. Give SAS code that will obtain a sign test that the median is 0.400 (against a two-sided alternative).

My answer: I'm including my (2015) reading of the file (which you don't need). What you *do* need is the `proc univariate` at the bottom of the code below. The `var` line is actually optional here, since there is only one variable.

```
SAS> data toenails;
SAS>   infile '/home/ken/arsenic.txt';
SAS>   input arsenic;
SAS>
SAS> proc univariate location=0.400;
SAS>   var arsenic;
```

```
The UNIVARIATE Procedure
Variable:  arsenic
                            Moments
N                        19    Sum Weights               19
Mean             0.27189474    Sum Observations       5.166
Std Deviation    0.23653797    Variance          0.05595021
Skewness         1.62668256    Kurtosis          1.94023175
Uncorrected SS     2.411712    Corrected SS      1.00710379
Coeff Variation  86.9961556    Std Error Mean    0.05426553

              Basic Statistical Measures
    Location                    Variability
Mean     0.271895     Std Deviation           0.23654
Median   0.158000     Variance                0.05595
Mode     0.118000     Range                   0.77800
                      Interquartile Range     0.24000

          Tests for Location: Mu0=0.4
Test           -Statistic-    -----p Value------
Student's t    t  -2.36071    Pr > |t|    0.0297
Sign           M      -5.5    Pr >= |M|   0.0192
Signed Rank    S       -53    Pr >= |S|   0.0317

Quantiles (Definition 5)
Level         Quantile
100% Max         0.851
99%              0.851
95%              0.851
90%              0.832
75% Q3           0.358
50% Median       0.158
```

```
25% Q1          0.118
10%             0.080
5%              0.073
1%              0.073
0% Min          0.073

         Extreme Observations
-----Lowest----        ----Highest----
 Value      Obs          Value      Obs
 0.073       11          0.358        6
 0.080        7          0.433       16
 0.099        3          0.517       13
 0.105       10          0.832       12
 0.118        4          0.851       14
```

Instead of `location`, you can say `mu0`, which is equally good. (I wrote in the appropriate assignment solutions about why I prefer `location` for myself, but `mu0` works just as well, so it gets full marks too.)

(c) (1 mark) The output is quite lengthy. Tell me something that would be next to the sign test in the output so that I can find it more easily. (No explanation needed.)

My answer: The sign test comes in a section called "Tests for location", next to the t-test that the *mean* is 0.400. Saying either of those things is good. (I was intending you to say "next to the t-test", but "in with the Tests for Location" is equally good.)

(d) (2 marks) Figure 12 shows an R function I wrote that will take a hypothesized median and run the sign test for that hypothesized median on the input data. It returns the two-sided P-value for the sign test in question. Figure 13 shows the P-value of the sign test on the arsenic data for various different null medians. What is the P-value for the test that you gave code for in (b)? What do you conclude about the median from this?

My answer: Just read off the P-value for the 0.40 line: 0.0192. This means that at $\alpha = 0.05$ we reject the null hypothesis that the median is 0.40, in favour of the alternative that it is not 0.40.

(e) (3 marks) Use the output in Figure 13 to obtain a 90% confidence interval for the population median. Explain briefly how you obtained your interval.

My answer: We need to use the idea that the confidence interval contains all the population medians that *are not* rejected at the right α by the test. Since we want a 90% CI, $\alpha = 0.10$, and thus the confidence interval goes from 0.12 to 0.31, since these are the values of the population median with P-values greater than 0.10.

8. The cardiovascular system in humans consists of the heart and blood vessels, which circulate blood around the body. It is important that this system operates properly, and there are a number of "risk factors" that might prevent it from doing so. One of these risk factors is a sedentary lifestyle.

   One study compared a group of runners (who averaged at least 15 miles per week of running) with a control group of "generally sedentary" people. The gender of each person was also noted. The outcome variable was the number of heart beats per minute after running on a treadmill for 6 minutes. The dataset we will use for this question is shown in Figure 14. (This is an excerpt of a much larger dataset.)

   In this question, we will be using the R package `dplyr`. (2017 note: `dplyr` is part of the `tidyverse`.) You can assume for this question that the package has already been installed (with `install.packages`) and loaded (with `library`). The data have been read into a data frame called `runners`. Give code (using `dplyr` tools) to accomplish the following tasks. You should need no more than three lines of code in each case, and each part *can* be done with two lines or fewer:

   (a) (2 marks) Display the runners, but not the "generally sedentary" people.

My answer: Here, and below, if you manage to obtain a solution that does *not* use `dplyr` tools, you can expect to get about half marks.

I need to read in the data first, and load `dplyr`, though of course you don't:

```
R> runners=read_tsv("runners.txt")
R> runners

Parsed with column specification:
cols(
  id = col_integer(),
  group = col_character(),
  sex = col_character(),
  beats = col_integer()
)
# A tibble: 800 x 4
      id   group    sex beats
   <int>   <chr>  <chr> <int>
 1     1 Control Female   159
 2     2 Control Female   183
 3     3 Control Female   140
 4     4 Control Female   140
 5     5 Control Female   125
 6     6 Control Female   155
 7     7 Control Female   148
 8     8 Control Female   132
 9     9 Control Female   158
10    10 Control Female   136
# ... with 790 more rows
```

Selecting individuals (rows) is done with `filter`, and the variable that distinguishes the runners from the rest is `group`. Here and below we only see the top 10 lines of the result; if you wanted to see them all, you could add a `print(n=Inf)` to the pipeline:

```
R> runners %>% filter(group=="Runners")

# A tibble: 400 x 4
      id   group    sex beats
   <int>   <chr>  <chr> <int>
```

```
 1   401 Runners Female   119
 2   402 Runners Female    84
 3   403 Runners Female    89
 4   404 Runners Female   119
 5   405 Runners Female   127
 6   406 Runners Female   111
 7   407 Runners Female   115
 8   408 Runners Female   109
 9   409 Runners Female   111
10   410 Runners Female   120
# ... with 390 more rows
```

(b) (3 marks) Display just the genders of the people that have `beats` less than 100.

My answer: This is `filter` again, but this time followed by a `select` to get the `sex`es:

```
R> runners %>% filter(beats<100) %>%
R>   select(sex)

# A tibble: 120 x 1
     sex
   <chr>
 1  Male
 2  Male
 3  Male
 4  Male
 5  Male
 6  Male
 7  Male
 8  Male
 9  Male
10  Male
# ... with 110 more rows
```

Having the `select` first would not have worked, since then there is no column `beats` left to select the values less than 100 from.

The obvious thing to do would have been to count these, but I already had a `summarize` question (later). Had I wanted to do so (and to demonstrate that some of the people with a low heart rate were actually female), I could have done this:

```
R> runners %>% filter(beats<100) %>%
R>   count(sex)

# A tibble: 2 x 2
     sex     n
   <chr> <int>
1 Female    30
2   Male    90
```

The `select` is no longer necessary, since none of the other columns got displayed anyway.

(c) (3 marks) Obtain the mean and standard deviation of `beats` for both the runners and the sedentary people.

My answer: Since we are doing this for both `group`s, we need a `group_by` first:

```
R> runners %>% group_by(group) %>%
R>   summarize(mean=mean(beats), sd=sd(beats))

# A tibble: 2 x 3
    group   mean       sd
    <chr>  <dbl>    <dbl>
1 Control 139.00 18.94961
2 Runners 109.98 15.53376
```

To no-one's great surprise, the heartbeats are typically quite a lot lower for the runners than the sedentary people.

9. Does logging (cutting down trees) in an area have an effect on the number of tree species present some years later? A study of logging in Borneo looked at 12 forest plots that had never been logged, and 9 (otherwise similar) plots that had been logged 8 years earlier. Some code and output is shown in Figures 15 through 17. 2017 note: this year we are not doing randomization tests, but you should think about what kind of test you would do instead.

   (a) (2 marks) Why, looking at the Figures, would you have doubts about doing a two-sample t-test to compare the tree species in the two types of plots? How would a randomization test be better? Explain briefly.

   My answer: Look at the boxplot in Figure 15. There are outliers: one high one among the unlogged plots and one low one among the logged plots. (This could exaggerate the difference between the two groups: it would increase the difference between the means, but would also increase the SDs within the groups.)

   The randomization test does not assume normality within the two groups, so its P-value can be trusted even when there are outliers.

   2017: Mood's median test does not assume normality within the two groups, so would be a reasonable test to do here as well. I did Mood's median test in the solution to the last part.

   (b) (3 marks) Describe briefly how the function in Figure 16 does something different from the first two lines of that Figure. (You can do this by describing what they each do, and then explaining how they are different.)

   My answer: The first two lines calculate the difference in group means for the observed data, and the function calculates the difference in group means for the *randomly shuffled* data. It's the same calculation, but on different data.

   (c) (2 marks) What does the top line of Figure 17 calculate, and how does it do it (in general terms, not in detail?)

   My answer: It obtains the randomization distribution of the difference in means. It does this by calculating the difference in means between the shuffled groups (from Figure 16) and repeating this 1000 times.

   (d) (2 marks) The research hypothesis was that a logged plot would have *fewer* tree species present 8 years later than an unlogged plot. Explain how the histogram in Figure 18 supports this hypothesis.

   My answer: The observed mean difference was calculated as logged minus unlogged (this is alphabetical order in `status`). The observed mean difference is negative, which is what you would expect if the research hypothesis is true. Not only that, it is *more* negative than most of the randomization distribution (which was calculated under the hypothesis that logging had no effect.)

   (e) (2 marks) Obtain a suitable P-value for the randomization test from Figure 17. What do you conclude in terms of the effect of logging?

   My answer: The test is one-sided (logging can only decrease the number of tree species, not increase it), so the P-value is simply $18/1000 = 0.018$.

   On this evidence, logging *does* decrease the mean number of tree species present in a plot. (The P-value is less than 0.05, so we reject the null hypothesis that logging has no effect in favour of the alternative that logging decreases the mean number of tree species.)

2017: we would do a Mood's median test, which in R can be done with the `smmr` package (or by piecing it together yourself):

```
R> borneo=read_delim("borneo.txt"," ")
R> borneo
R> library(smmr)
R> median_test(borneo,species,status)
```

```
Parsed with column specification:
cols(
  status = col_character(),
  species = col_integer()
)
# A tibble: 21 x 2
     status species
      <chr>   <int>
 1 unlogged      32
 2 unlogged      22
 3 unlogged      15
 4 unlogged      13
 5 unlogged      19
 6 unlogged      19
 7 unlogged      18
 8 unlogged      20
 9 unlogged      21
10 unlogged      13
# ... with 11 more rows
$table
          above
group      above below
  logged       3     4
  unlogged     7     3

$test
       what     value
1 statistic 1.2524490
2        df 1.0000000
3   P-value 0.2630853
```

Most of the unlogged plots have an above-average number of species, whereas for the logged plots the number of species could be above or below the overall median. (In case you are wondering how that is possible: there are 21 plots altogether, but only 17 in the `table`, which means that 4 of the plots must have had a number of species exactly equal to the overall median and were thus discarded from the analysis.)

This test should be done two-sided, since we are looking for "an effect" without saying anything about which way it goes. The `table` says the effect is quite small, since the table doesn't look all that unbalanced, and the two-sided P-value is 0.2631 says that we are nowhere near rejecting a null hypothesis that the median number of species in the two types of plot is the same.

This is a substantial difference from the randomization test, which was based on the means; if you look at the boxplot, you'll see that each group has one outlier, but they are in *opposite* directions, so there is the potential for the difference in means to be either very small or very large. With the data as they are, the difference in means (logged minus unlogged) will be at the negative end of the distribution of possible values, whereas there might not be anything interesting in the comparison of medians.

Let's see whether the Mood's median test calculation makes any sense in the light of the boxplot, Figure 15:

```
R> borneo %>% summarize(med=median(species))

# A tibble: 1 x 1
    med
  <int>
1    15
```

The overall median is 15. This is also the median of the `logged` values, so about half of them should be above and half below the overall median. This is what the Mood's median test table says. For the `unlogged` values, the overall median of 15 is down near the first quartile, so not quite 75% of the values in that group should be above and a little more than 25% should be below. This is also supported by the table.

Final thoughts: what about those values equal to the overall median? Let's include them in our summary:

```
R> borneo %>%
R>   mutate(where=case_when(
R>     species<15  ~ "below",
R>     species==15 ~ "equal",
R>     species>15  ~ "above")) %>%
R>   group_by(status,where) %>%
R>   count()

# A tibble: 6 x 3
# Groups:   status, where [6]
    status where     n
     <chr> <chr> <int>
1   logged above     3
2   logged below     4
3   logged equal     2
4 unlogged above     7
5 unlogged below     3
6 unlogged equal     2
```

Each group had two values exactly equal to the overall median, which meant that out of the original 21 values, only 17 remained for the test (seven in the `logged` group, and 10 in the `unlogged` group). This also explains how one group could be about 50–50 above and below the overall median and the other group could be mostly above: overall, there are 10 observations above the overall median, only 7 below, but *four exactly equal*, and because of this, 15 is the best value for the overall median.