MA4605: Lab 2

Getting started

The example dataset contains information for $n = 32$ students. There were 6 questions asked and we have identified in the lecture the data type of the responses for each question.

We begin by loading the data set into the R workspace and saving it in a dataframe called `dat`.

```
dat = read_csv(file="../data/ExampleData.csv")
```

```
## Parsed with column specification:
## cols(
##   LC.points = col_double(),
##   Exercise = col_character(),
##   Num.times.exercise = col_double(),
##   Facebook = col_character(),
##   Stats.ability = col_double(),
##   Stats.interest = col_double()
## )
```

Each row in the dataframe represents an individual student. Each column represents a *variable*.

To view the names of the variables, type the command

```
names(dat)
```

```
## [1] "LC.points"          "Exercise"           "Num.times.exercise"
## [4] "Facebook"           "Stats.ability"      "Stats.interest"
```

A list of the variables and what they represent is given below:

- `LC.points`: Leaving Cert points
- `Exercise`: exercise or not (Yes/No)
- `Num.times.exercise`: number of times per week of exercise
- `Facebook`: have a Facebook account (Yes/No)
- `Stats.ability`: rate your stats ability (1: Very poor, 2: poor, 3: good, 4: very good, 5: excellent)
- `Stats.interest`: statistics is interesting (1: Strongly disagree, 2: disagree, 3: neutral, 4: agree, 5: strongly agree)

We can have a look at the first few entries (rows) of our data with the command

```
head(dat)
```

```
## # A tibble: 6 x 6
##   LC.points Exercise Num.times.exercise Facebook Stats.ability Stats.interest
##       <dbl> <chr>                 <dbl> <chr>            <dbl>          <dbl>
## 1        NA Yes                       3 No                   4              4
## 2       498 Yes                       5 Yes                  3              5
## 3       509 Yes                       4 Yes                  4              5
## 4       509 Yes                       7 No                   2              2
## 5       502 Yes                       3 No                   5              5
## 6       500 Yes                       5 No                   3              4
```

and similarly we can look at the last few by typing

```r
tail(dat)
```

```
## # A tibble: 6 x 6
##   LC.points Exercise Num.times.exercise Facebook Stats.ability Stats.interest
##       <dbl> <chr>                 <dbl> <chr>            <dbl>          <dbl>
## 1       554 Yes                       2 Yes                  3              3
## 2       456 Yes                       2 Yes                  3              3
## 3       566 Yes                       7 Yes                  4              4
## 4       589 Yes                       4 No                   3              3
## 5       543 Yes                       5 Yes                  3              3
## 6       487 Yes                       4 Yes                  3              5
```

You could also look at *all* of the data frame at once by typing its name into the console.

Recoding variables

The `Stats.ability` and `Stats.interest` variables are recorded as numbers which represent each category of the Likert scales. We need to recode these variables from numbers to category labels. Let's start with the `Stats.ability` variable. The `recode_factor` command creates a labelled factor from the corresponding numbers.

```r
dat$Stats.ability = recode_factor(dat$Stats.ability, `1` = "very poor", `2` = "poor", `3` = "good", `4`
```

Exercise 1:

Recode the `Stats.interest` variable using the labels given in the lab sheet.

```r
# Write your R code for Exercise 1 here.
```

Summarise and plot categorial variables

Let's start by creating a frequency distribution for the `Stats.ability` variable. This is achieved using the following command

```r
dat %>%
  count(Stats.ability) %>%
  mutate(prop = prop.table(n), pct = prop.table(n)*100) %>%
  kable() %>%
  kable_styling()
```

Stats.ability	n	prop	pct
very poor	2	0.06250	6.250
poor	1	0.03125	3.125
good	18	0.56250	56.250
very good	10	0.31250	31.250
excellent	1	0.03125	3.125

The `count()` command creates counts/frequencies in each category. The `mutate()` command creates a new variable called `prop` which calculates the proportion/relative frequency in each category. The `kable()` and `kable_styling()` commands outputs this table in a nice format.

Comment: 56% of students rate their ability in statistics as good. A further 31% rate their ability as very good. 9% of students rate their statistics ability as poor (1 student) or very poor (2 students). Only one student (3%) rated their ability as excellent.

Exercise 2:

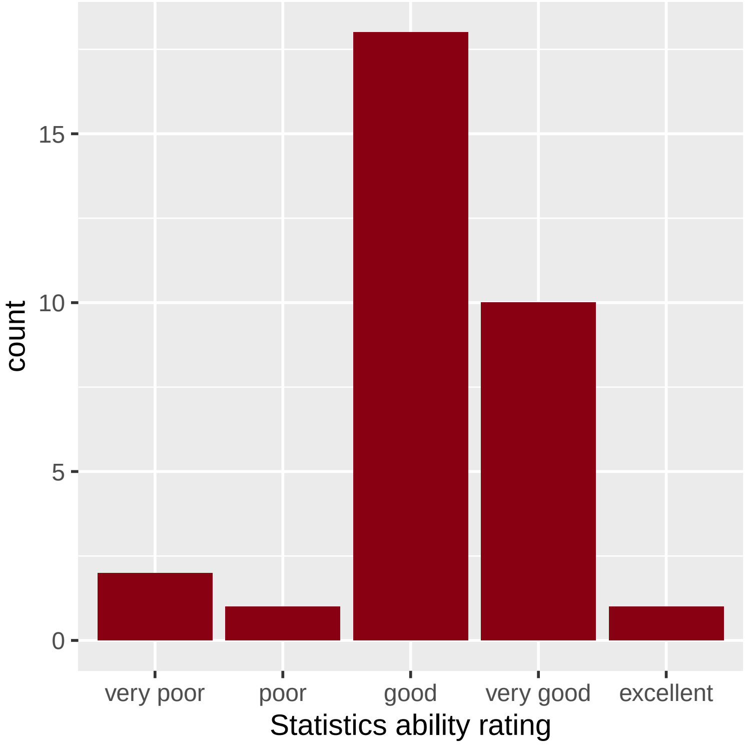
Create a frequency distribution for the `Stats.interest` variable and comment.

```
# Write your R code for Exercise 2 here.
```

[Write your comment for Exercise 2 here.]

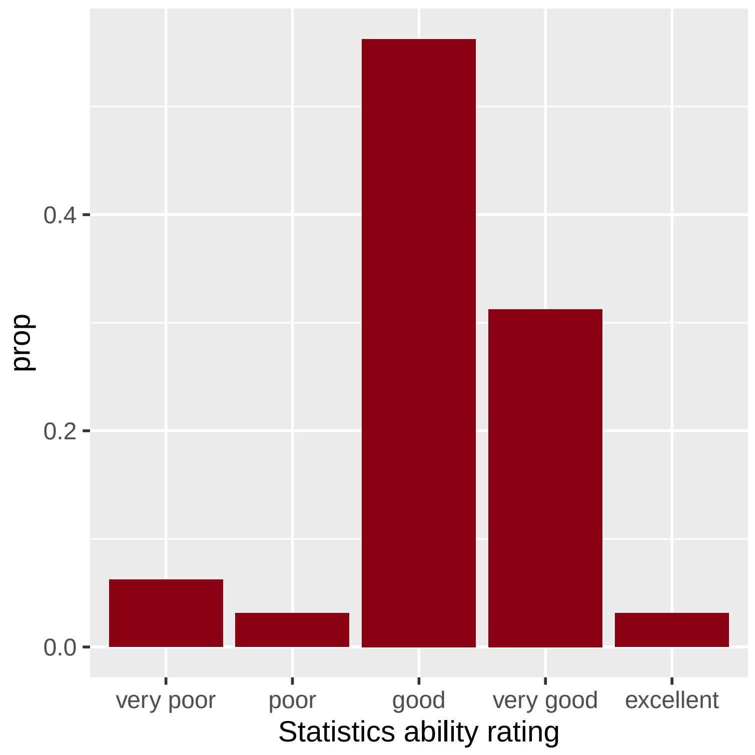
To create a barchart plotting the frequencies on the y-axis, use `ggplot` with the `geom_bar` command. The `fill="#880011"` changes the colours of the bars.

```
  ggplot(aes(x=Stats.ability), data=dat) + geom_bar(fill="#880011") + xlab("Statistics ability rating")
```



To create a barchart with the relative frequencies on the y-axis, we need to first calculate the relative frequencies using the `mutate` command from previously. Then we need to create a barchart.

```
dat %>%
  count(Stats.ability) %>%
  mutate(prop = prop.table(n)) %>%
  ggplot(aes(x=Stats.ability, y=prop)) + geom_bar(stat="identity", fill="#880011") +
  xlab("Statistics ability rating")
```



Exercise 3:

Create a barchart for the `Stats.interest` variable. You can choose whether to plot the frequency or relative frequency on the y-axis. Play around with the colour of the bars.

```
# Write your R code for Exercise 3 here.
```

Summarise and plot numeric variables

To create summary statistics for the variable `LC.points` use the following code:

```
dat %>%
  summarise(min = min(LC.points,na.rm=TRUE), max = max(LC.points,na.rm=TRUE),
            mean = mean(LC.points,na.rm=TRUE), med = median(LC.points,na.rm=TRUE),
            sd = sd(LC.points,na.rm=TRUE),
            q1 = quantile(LC.points, probs = 0.25,na.rm=TRUE),
            q3 = quantile(LC.points, probs = 0.75,na.rm=TRUE), IQR = IQR(LC.points,na.rm=TRUE))
```

```
## # A tibble: 1 x 8
##     min   max  mean   med    sd    q1    q3   IQR
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1   405   601  525.  526.  45.2   498  558.  59.8
```

The `na.rm=TRUE` tells R to ignore missing (NA) values in the calculations. Again, using the `kable()` and `kable_styling()` commands outputs this table in a nicer format.

```
dat %>%
  summarise(min = min(LC.points,na.rm=TRUE), max = max(LC.points,na.rm=TRUE),
            mean = mean(LC.points,na.rm=TRUE), med = median(LC.points,na.rm=TRUE),
            sd = sd(LC.points,na.rm=TRUE),
            q1 = quantile(LC.points, probs = 0.25,na.rm=TRUE),
            q3 = quantile(LC.points, probs = 0.75,na.rm=TRUE), IQR = IQR(LC.points,na.rm=TRUE)) %>%
  kable() %>%
  kable_styling()
```

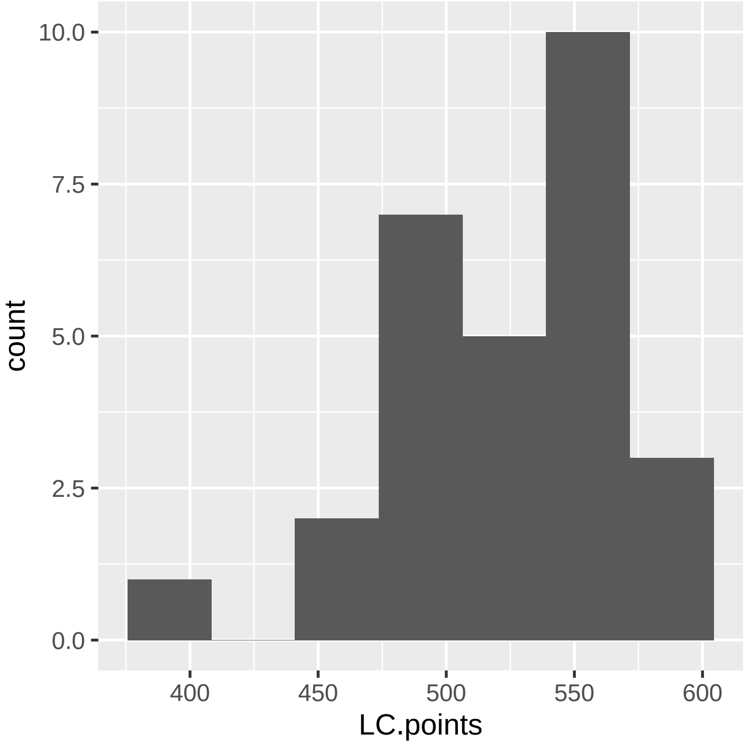
min	max	mean	med	sd	q1	q3	IQR
405	601	525.0357	525.5	45.2364	498	557.75	59.75

The two plots that are appropriate for numeric variables are a histogram and boxplot.

To create a histogram with 7 bins:

```
ggplot(aes(x=LC.points), data=dat) + geom_histogram(bins=7)
```

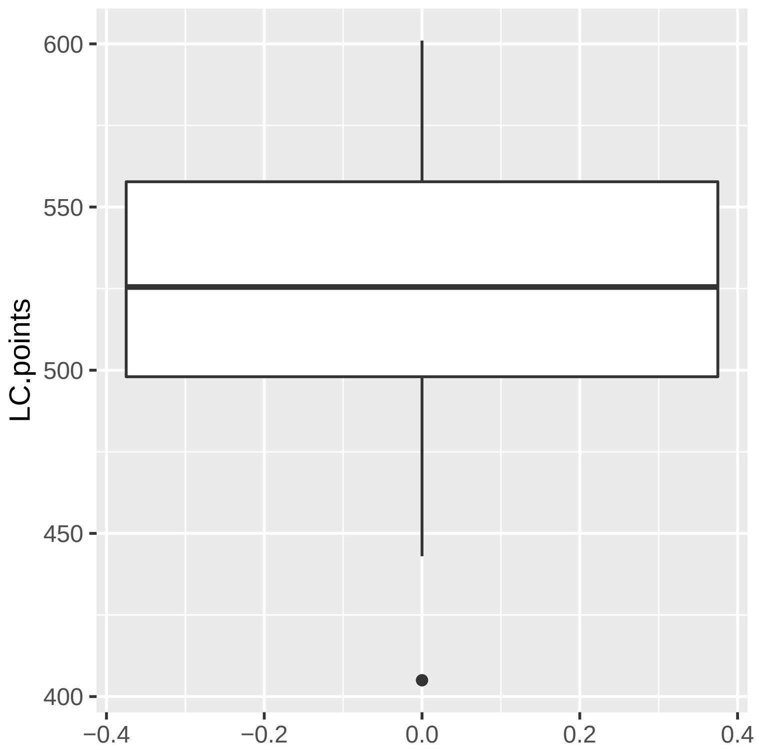
```
## Warning: Removed 4 rows containing non-finite values (stat_bin).
```



To create a boxplot:

```
ggplot(aes(y=LC.points), data=dat) + geom_boxplot()
```

```
## Warning: Removed 4 rows containing non-finite values (stat_boxplot).
```



Comment: The histogram is slightly negatively skewed with one potential outlier at 400 points. However the shape of the boxplot suggest that the Leaving Cert points data can be approximated by a Normal distributions. (The median is in the middle of the box, the tails are approximately symmetric.) There is one obvious outlier that should be noted. The mean number of Leaving Cert points is 525 (sd = 45.24).

Exercise 4:

Create a summary statistics, a histogram and a boxplot for the `Num.times.exercise` variable and comment on the results.

```
# Write your R code for Exercise 4 here.
```

[Write your comment for Exercise 4 here.]