Math Skills 2 - Statistics Assessment

Part A

1: Packages

First I will look at Part A of the Statistics Assessment relating to Crime data.

The first task is to load the tidyverse package as well as the latest version of **ggmap**. I used the **library()** function to do this. These packages are essential as they contain functions that I will be using often.

```
library(tidyverse)
library(ggmap)
```

Now I need to check I have the latest version of **ggmap**.

```
packageVersion("ggmap")
```

```
## [1] '3.0.0.901'
```

This is indeed the correct version, so I can move onto the next step.

2: Downloading and formulating the data

I decided to pick Chester for my city of choice and all the data were provided online by the Cheshire county. The data were given in 12 separate folders each containing a month, from January to December, of 2019. Once these folders are unzipped they can be imported into R Studio using the **read_csv()** function.

```
crimejan <- read_csv("C:/Users/aels7/Documents/Q1Data/2019-01-cheshire-street.csv")
crimefeb <- read_csv("C:/Users/aels7/Documents/Q1Data/2019-02-cheshire-street.csv")
crimemar <- read_csv("C:/Users/aels7/Documents/Q1Data/2019-03-cheshire-street.csv")
crimeapr <- read_csv("C:/Users/aels7/Documents/Q1Data/2019-04-cheshire-street.csv")
crimemay <- read_csv("C:/Users/aels7/Documents/Q1Data/2019-05-cheshire-street.csv")
crimejun <- read_csv("C:/Users/aels7/Documents/Q1Data/2019-06-cheshire-street.csv")
crimejul <- read_csv("C:/Users/aels7/Documents/Q1Data/2019-07-cheshire-street.csv")
crimeaug <- read_csv("C:/Users/aels7/Documents/Q1Data/2019-08-cheshire-street.csv")
crimesep <- read_csv("C:/Users/aels7/Documents/Q1Data/2019-09-cheshire-street.csv")
crimeoct <- read_csv("C:/Users/aels7/Documents/Q1Data/2019-10-cheshire-street.csv")
crimenov <- read_csv("C:/Users/aels7/Documents/Q1Data/2019-11-cheshire-street.csv")
crimedec <- read_csv("C:/Users/aels7/Documents/Q1Data/2019-12-cheshire-street.csv")
```

Now all the data have been imported into R Studio. For convenience, I'm going to use the **bind_rows()** function to add all the data to one large dataset. This large dataset will be called *crimes* and it will contain all the data from the excel files.

```
crimes <- bind_rows(crimejan, crimefeb, crimemar, crimeapr, crimemay, crimejun,
                    crimejul, crimeaug, crimesep, crimeoct, crimenov, crimedec)
```

The data supplied by the Cheshire county contains lots of information that I don't need, like the Crime ID, and the Final Outcome. Since I don't need this, I can reduce the dataset, *crimes*, to the four relevant columns using the **select()** function. This is the Longitude, Latitude, Month and the Crime Type.

```
crimes %>%
  mutate(CrimeType = `Crime type`) %>%
  select(Longitude, Latitude, Month, CrimeType) -> crimes
```

3: Attaining the Map of Chester

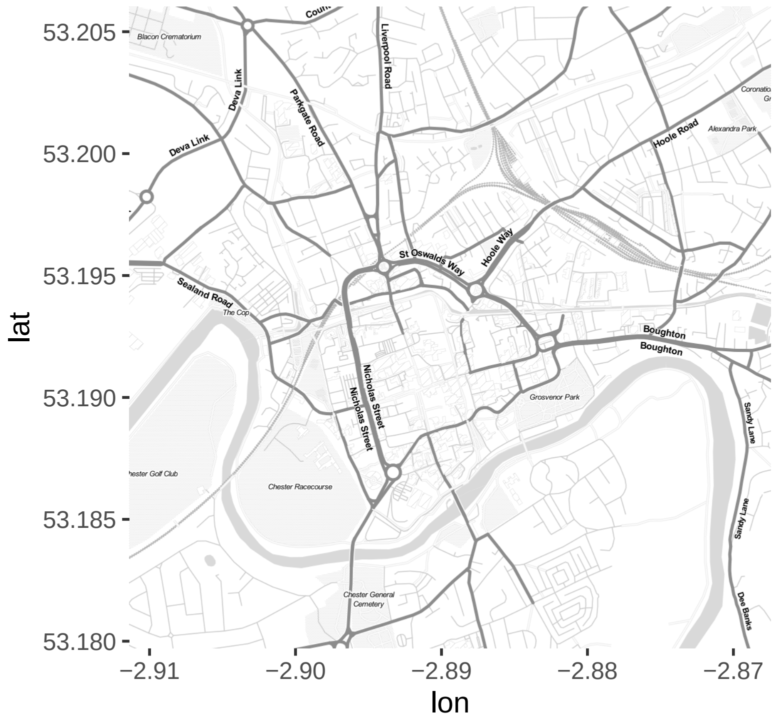
Now I can have a look at a map of Chester and plot all of the crimes on the map. First, for the map of Chester:

```
geocode("Chester", source="dsk")
```

```
## # A tibble: 1 x 2
##     lon   lat
##   <dbl> <dbl>
## 1 -2.58  53.2
```

I used the **geocode()** function to find to co-ordinates for Chester. The two co-ordinates above give the approximate centre of Chester. As I need a map I can create a variable called **bbox** that encompasses the co-ordinates given previously. After setting **bbox** I use **get_map()** to display a map based on the co-ordinates I set in **bbox**. I set the zoom and after some playing around with the co-ordinates in **bbox** I used **ggmap()** to display the map of Chester.

```
bbox <- c(-2.91, 53.2, -2.87, 53.18)
map <- get_map(location = bbox, source = "stamen", maptype = "toner-lite", zoom = 15,
               crop = FALSE)
ChesterMap <- ggmap(map)
ChesterMap
```



4: Specifying crimes

Next step is to overlay the crime I imported earlier on the map of Chester.

To do this I create a second **bbox** that is slightly larger that the original **bbox** to ensure that it encompasses the previous **bbox**. I will use this new **bbox** to reduce all the crime in Chester to just the crime that lies within the co-ordinates below.

```
bboxlarge<- c(-2.94, 53.23, -2.84, 53.15)
```

Now that I have our larger **bbox** I need to filter all the crimes to just the crimes that will be visible over the map using the **filter()** function.

```
crimeChester <- filter(crimes, bboxlarge[1] <= Longitude, Longitude <= bboxlarge[3],
                       bboxlarge[2] >= Latitude,  Latitude  >= bboxlarge[4])
```

Now I have greatly reduced the crimes so I can only see the crimes that will show up on the map. The next step is to see how much crime there is per crime type. I use the table function to show this.

```
table(crimeChester$CrimeType)
```

```
##
##       Anti-social behaviour                Bicycle theft
##                        2496                          138
##                    Burglary    Criminal damage and arson
##                         291                          750
##                       Drugs                  Other crime
##                         327                          120
##                 Other theft        Possession of weapons
##                         749                           53
##                Public order                      Robbery
##                        1601                           52
##                 Shoplifting        Theft from the person
##                         771                           84
##               Vehicle crime Violence and sexual offences
##                         180                         3242
```

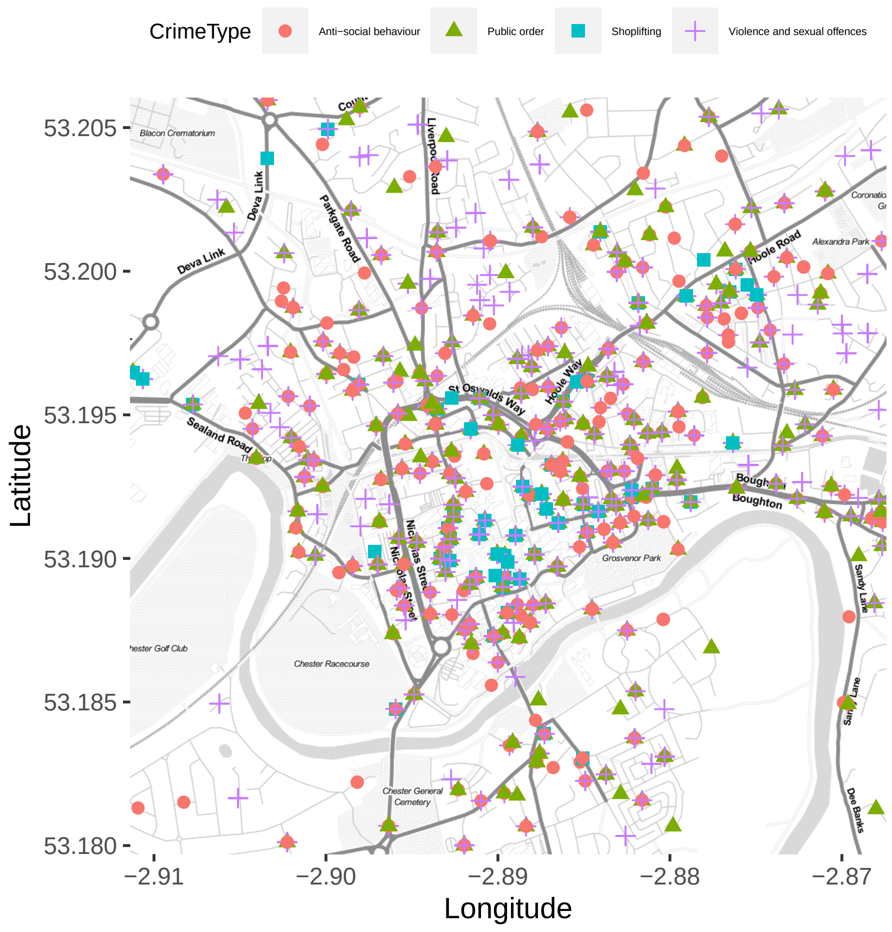
From this table I can see that *Anti-social behaviour*, *Violence and sexual offences*, *Public order*, and *Shoplifting* are the most frequent. So I will reduce all the crimes in our area to just the crimes that fall in those categories.

```
crimeChester %>%
  filter(CrimeType == "Anti-social behaviour" |
           CrimeType == "Violence and sexual offences" |
           CrimeType == "Public order" |
           CrimeType == "Shoplifting"
  ) -> crimeChester
```

5: Adding crimes to map

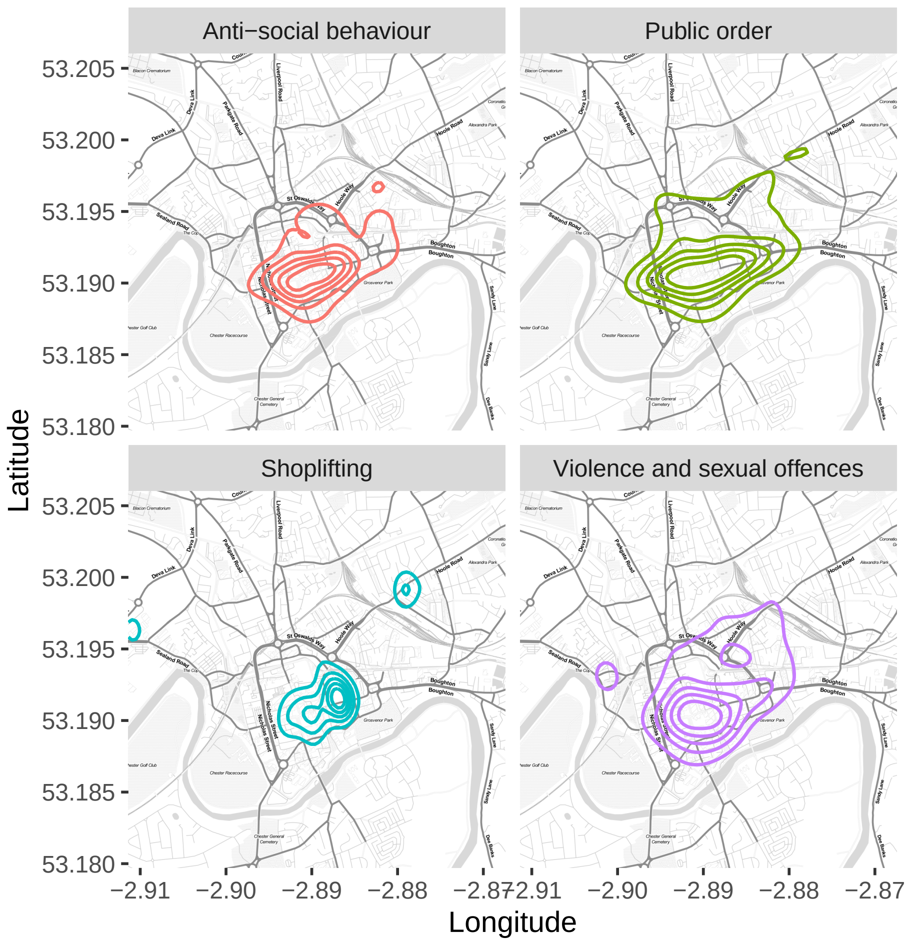
With only four types of crime, I can plot them on our map and categorise them based on crime type. To do this I used **geom_point()** to display the points on the map where each crime has taken place. In the second line, you can see the colour and shape of the points on the map vary depending on the type of crime. As there are several points to display I have chosen a small size, 2, for the points so the map remains clear. A legend is visible on top of the map to distinguish which points/colours are used for which crime type. I have also set the name of the axes in the last line for simplicity.

```
ChesterMap +
  geom_point(aes(x = Longitude, y = Latitude, colour = CrimeType, shape=CrimeType),
             size=2, data = crimeChester) +
  theme(legend.position = "top", legend.text = element_text(size=4),
        legend.title = element_text(size=8)) +
  labs(y="Latitude", x="Longitude")
```



I can also show the crime based on density using much of the same code as before. All that has changed is I now have four separate maps for each crime type and used the **facet_wrap** for the density.

```
ChesterMap +
  geom_density2d(aes(x = Longitude, y = Latitude, colour = CrimeType),
                 size=0.6, bins=7, data = crimeChester) +
  theme(legend.position = "none") +
  facet_wrap(~ CrimeType) +
  labs(y="Latitude", x="Longitude")
```



Part B

1: Inputting all the data

All the packages and functions that I will be using in Part B have already been unpacked and loaded from Part A, so I do not need to load them again.

Like in Part A, I need to import all the data using the function, **read_csv()**. The data were given as one large excel file with six sheets representing each year from 2014 to 2019. In order to read these sheets into R, I saved each sheet separately as a **.csv** file and used the **read_csv()** function to import them into R Studio.

```r
fourdata <- read_csv("C:/Users/aels7/Documents/Q2Data/Published_dataset_2014.csv")
fivedata <- read_csv("C:/Users/aels7/Documents/Q2Data/Published_dataset_2015.csv")
sixdata <- read_csv("C:/Users/aels7/Documents/Q2Data/Published_dataset_2016.csv")
sevendata <- read_csv("C:/Users/aels7/Documents/Q2Data/Published_dataset_2017.csv")
eightdata <- read_csv("C:/Users/aels7/Documents/Q2Data/Published_dataset_2018.csv")
ninedata <- read_csv("C:/Users/aels7/Documents/Q2Data/Published_dataset_2019.csv")
```

I used **bind_rows()** to combine all the datasets in to one large dataset called *landings*. I also used **.id** in the final line to link each dataset to its original year. This adds a column called Year that contains the year of each row.

```
landings <- bind_rows("2014" = fourdata,
                      "2015" = fivedata,
                      "2016" = sixdata,
                      "2017" = sevendata,
                      "2018" = eightdata,
                      "2019" = ninedata,
                      .id = ("Year"))
```

To add the Time column, I will use the **paste()** function as follows. This creates a new column called Time which retrieves the Year from the Year column, the Month from the Month column, and adds "-15" to create the date in a yyyy-mm-dd format. As the **paste()** function automatically adds a space between each input, the final expression on the first line removes that space. The second line uses **mutate()** to classify the column, Time as a Date, and not a string of characters.

```
landings$Time <- paste(landings$Year,"-", landings$`Month Landed`, "-15", sep="")
landings <- mutate(landings, Time = Time %>% as.Date())
```

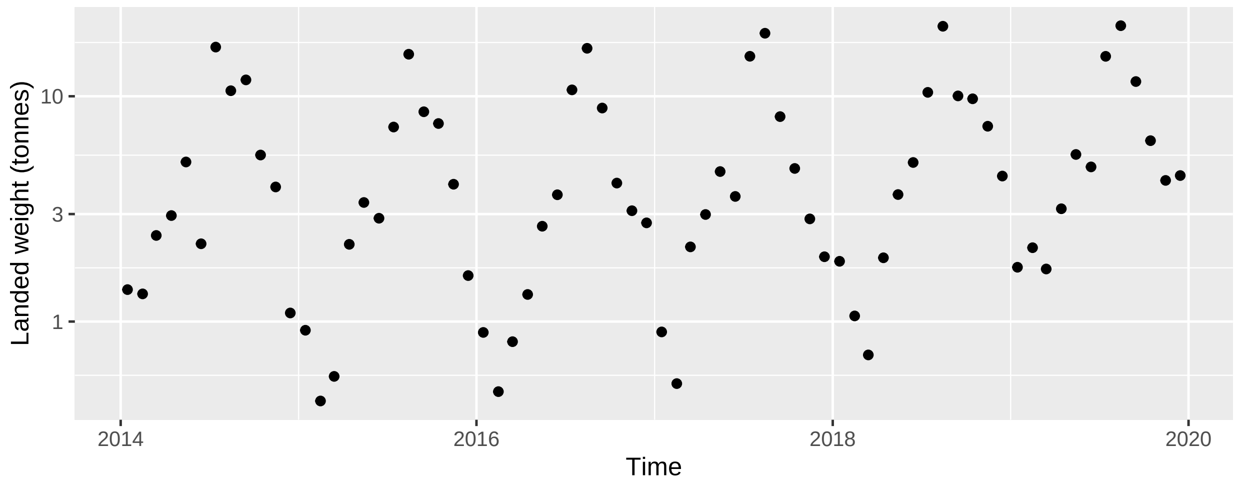
2: Creating subgroups, and plotting graphs

Now I want to create a new variable called *WhitbyU10mLobs*, for all the Lobsters that landed at Whitby and are at a size of 10m and under. The **filter()** function works well for this.

```
WhitbyU10mLobs <- filter(landings, `Port of Landing` == "Whitby" & Species == "Lobsters"
                         & `Length Group` == "10m&Under")
```

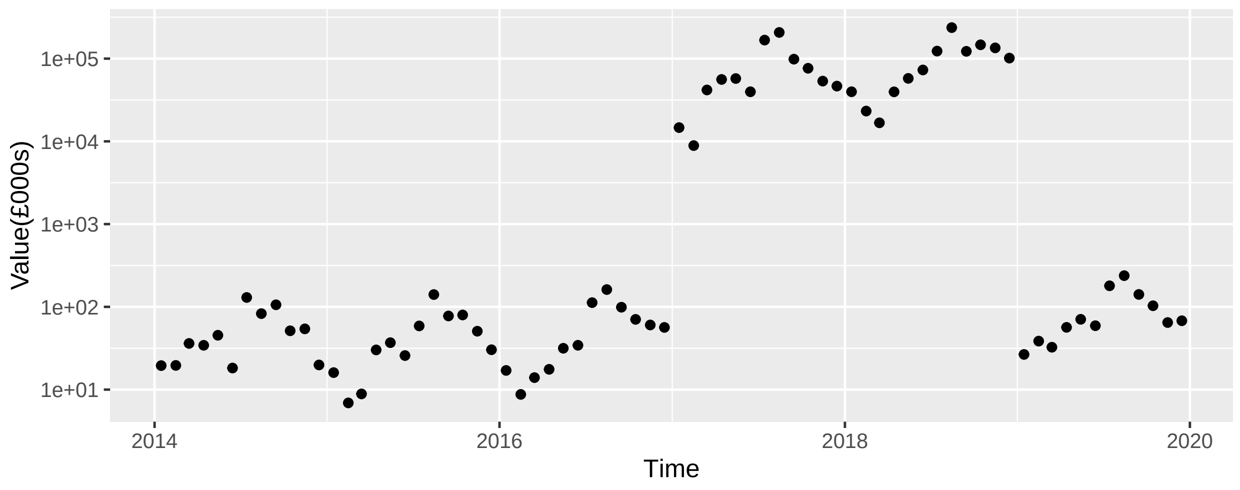
Now I can plot a graph using **ggplot()** with Time on the x-axis, and Landed Weight on the y-axis. I have used **scale_y_log10()** to display a log scale on the y-axis and **geom_point()** to show the points on the graph.

```
ggplot(WhitbyU10mLobs, aes(x=`Time`, y=`Landed weight (tonnes)`)) + geom_point() +
  scale_y_log10()
```



This second graph is very similar to the last. In this graph I have placed Time on the x-axis and Value on the y-axis.

```
ggplot(WhitbyU10mLobs, mapping = aes(x = Time, y = `Value(£000s)`)) + geom_point() +
  scale_y_log10()
```



From the second graph I can see there is a large discrepancy for years 2017 and 2018. This needs to be investigated and fixed.

3: Investigating and fixing errors

I can see on the second graph that there is a large discrepancy for years 2017 and 2018. To confirm this I can create a new tibble with the sums of the Value for each year.

```
by_year <- group_by(landings, Year)
sum_per_year <- summarise(by_year, "Total Value(£000s)" = sum(`Value(£000s)`, na.rm = TRUE))
```

```
sum_per_year
```

```
## # A tibble: 6 x 2
##   Year  `Total Value(£000s)`
##   <chr>                <dbl>
## 1 2014               946571.
## 2 2015               841704.
## 3 2016              1024996.
## 4 2017           1049923943.
## 5 2018           1054021169.
## 6 2019               977732.
```

I can now see that indeed the sums for 2017 and 2018 are out by a factor of 1000. To fix this, I need to divide all the values in the 'Value(£000s)' column by 1000 and reinstate the landings dataset. The data before:

```
head(eightdata$`Value(£000s)`)
```

```
## [1]   211.86   152.70  5102.96   134.76 12187.63   949.05
```

Change the values for both years…

```
sevendata$`Value(£000s)` <- sevendata$`Value(£000s)`/1000
eightdata$`Value(£000s)` <- eightdata$`Value(£000s)`/1000
```

The data after:

```
head(eightdata$`Value(£000s)`)
```

```
## [1]  0.21186  0.15270  5.10296  0.13476 12.18763  0.94905
```

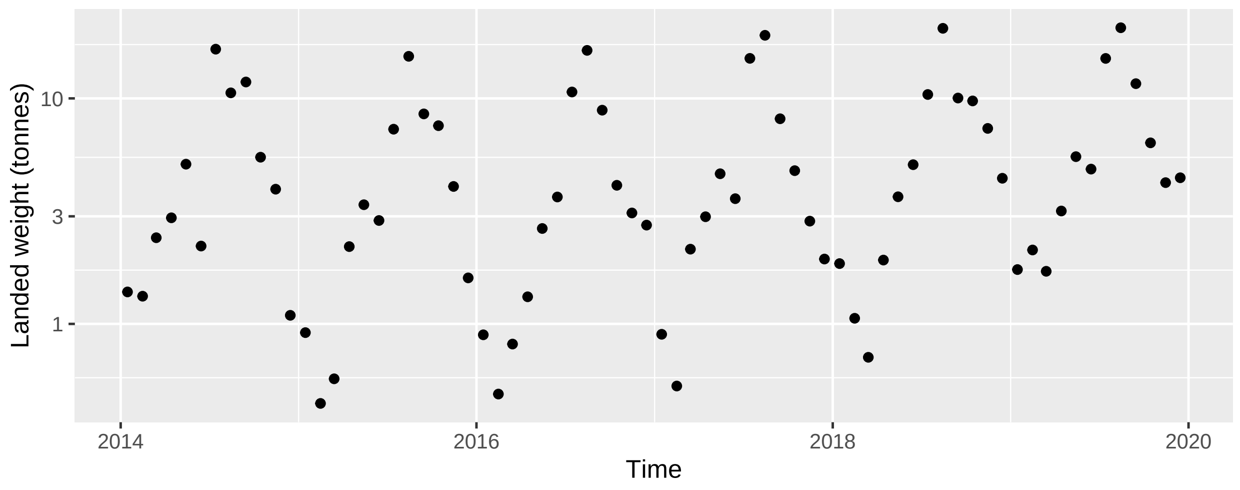
The data has now been corrected so I can recompile it, and reinstate our previous variables exactly as I did before.

```
landings <- bind_rows("2014" = fourdata, "2015" = fivedata,
                      "2016" = sixdata, "2017" = sevendata,
                      "2018" = eightdata, "2019" = ninedata,
                      .id = ("Year"))

landings$Time <- paste(landings$Year,"-", landings$`Month Landed`, "-15",sep="")
landings <- mutate(landings, Time = Time %>% as.Date())

WhitbyU10mLobs <- filter(landings, `Port of Landing` == "Whitby" & Species == "Lobsters"
                         & `Length Group` == "10m&Under")
```

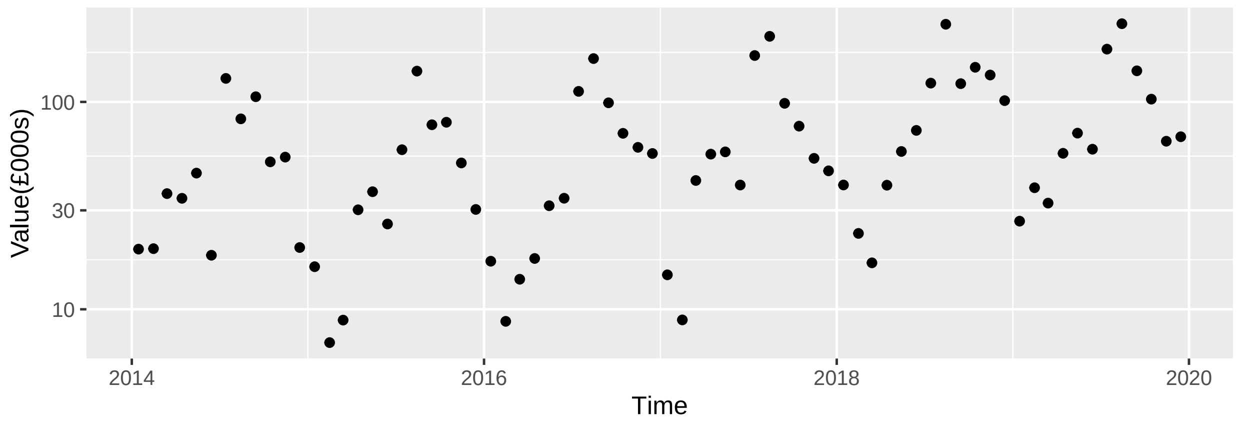
4: Re-plotting the graph

I used the same code as before but now with the corrected data.

```
ggplot(WhitbyU10mLobs, aes(x=`Time`, y=`Landed weight (tonnes)`)) + geom_point() +
  scale_y_log10()
```



```
ggplot(WhitbyU10mLobs, aes(x =`Time`, y =`Value(£000s)`)) + geom_point() + scale_y_log10()
```



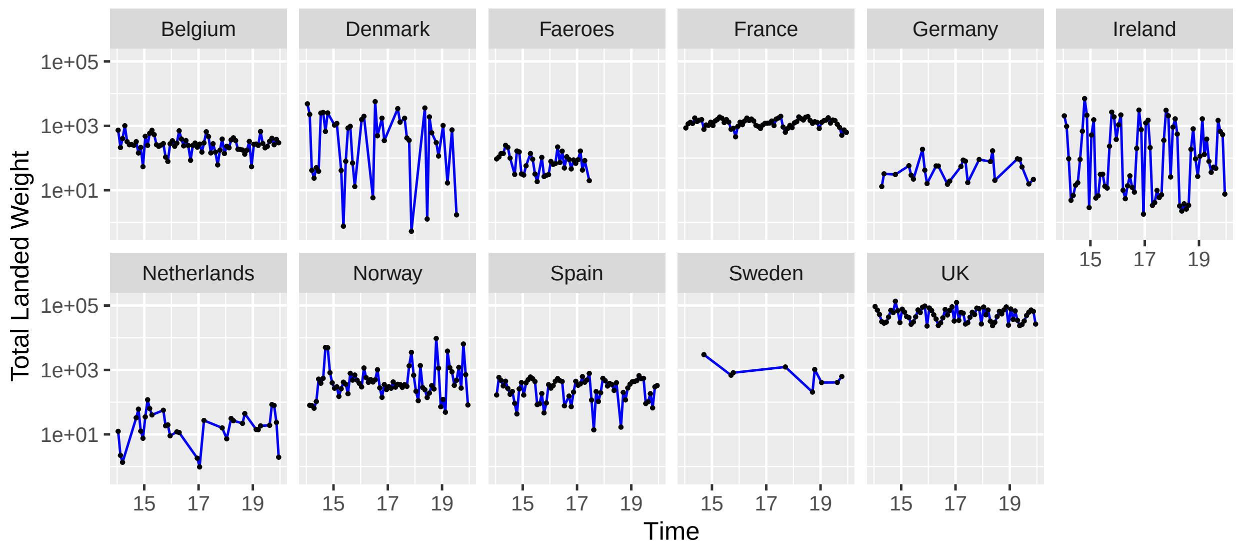
I can see, from these two graphs, that the middle of the year tends to have the largest value as well as the largest landed Weight as opposed to the start/end of the year. The discrepancy has been solved.

Now I want to create a new variable called *LandWeiByVessNat* where I will use the **group_by()** and **summarise()** functions to produce a dataset with Time, Vessel Nationality and Total Landed Weight. This new dataset, *LandWeiByVessNat* represents the total landed weight of fish from each nation per month.

```
by_Month <- group_by(landings, `Vessel Nationality`, Time)
LandWeiByVessNat <- summarise(by_Month, "Total Landed Weight"=sum(`Landed weight (tonnes)`))
```

Once this variable has been assigned, I can plot multiple graphs displaying the total landed weight of fish imported from each country from 2014-2019.

```
ggplot(data = LandWeiByVessNat, mapping = aes(x = Time, y = `Total Landed Weight`)) +
  scale_y_log10() + geom_line(color = "blue") + geom_point(size = 0.4) +
  facet_wrap( ~ `Vessel Nationality`, ncol=6) + scale_x_date(date_labels = "%y",
                                                          date_breaks = "2 years")
```

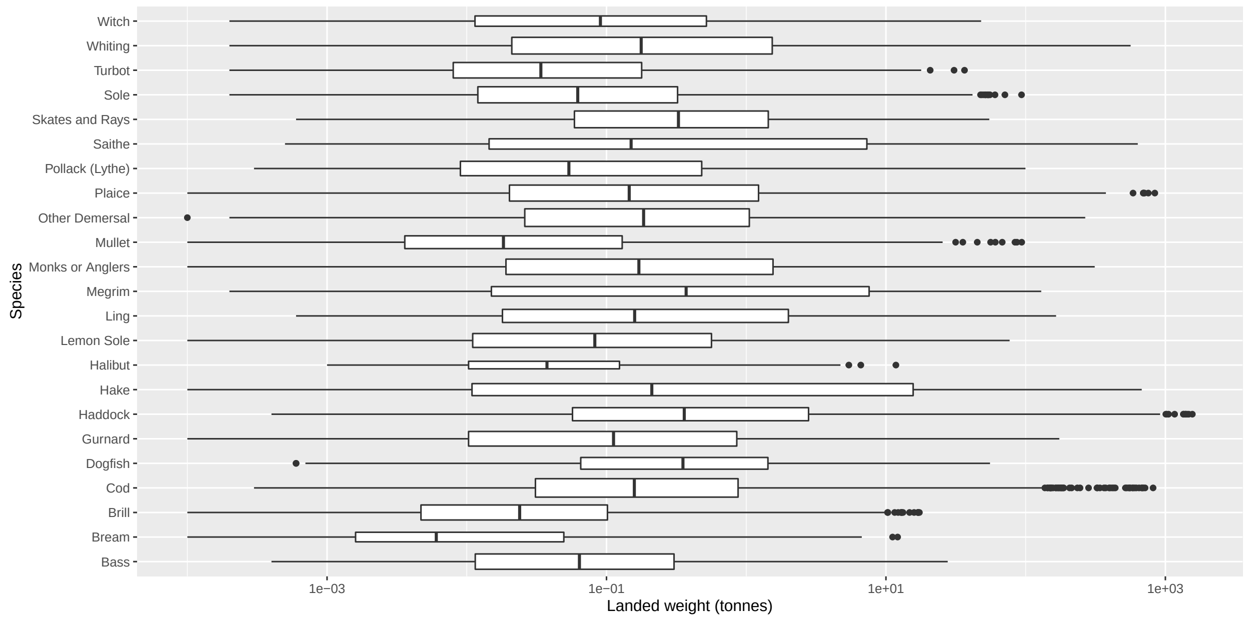


Now, I will create a final subgroup of landings where the Species group, Demersal, arrived in either January, or February and use the **group_by()** function to group the landed weight for each Species based on their Land Weight.

```r
demersalgroup <- filter(landings, landings$`Month Landed` %in% c("1", "12") &
                          landings$`Species Group` == "Demersal")
LandWeiBySpec <- group_by(demersalgroup, Species)
```

And now I can plot the final graph using boxplots for each Species. Note that I have once again used a log scale on the y-axis and have also used **coord_flip()** to move the species names to the y-axis so they can be read without any overlapping. Also note the width of the box plots represents the variance for each species.

```r
ggplot(data = LandWeiBySpec) +
  geom_boxplot(mapping = aes(x = Species, y = `Landed weight (tonnes)`), varwidth=TRUE) +
  scale_y_log10() + coord_flip()
```



As shown in the graph, the landed weights of the different fish species overlap considerably with Bream showing the lowest median weight and the highest weights record for haddock.

References:

1. https://data.police.uk/data/. Data for Part A.
2. https://www.gov.uk/government/statistical-data-sets/uk-and-foreign-vessels-landings-by-uk-port-and-uk-vessel-landings-abroad. Data for Part B.
3. https://rdrr.io/cran/dplyr/man/bind.html. This source was used to help with the .id argument in **bind_rows()**. Specifically line 46 on webpage.
4. https://stackoverflow.com/questions/20338813/how-do-i-multiply-square-divide-etc-a-single-column-in-r. Used to divide values of a column on lines 230-231 in .Rmd file.
5. http://environmentalcomputing.net/plotting-with-ggplot-adding-titles-and-axis-names/. I used this reference to change the names of the axes on my graphs.
6. https://stat.ethz.ch/pipermail/r-help/2006-January/086807.html. I used this reference for **sep=""** in the **paste()** function that removes the default spaces for the date column of *landings*.

Packages Uses: *tidyverse*, *ggmap*