Mathematical Skills 2- Recent advances - Statistics

30 April 2020

Street Crime data of Lewisham

The goal of the task is to show a graphical summary of the street crimes of the borough of Lewisham during 2019. The first command we use is to find the source location so that we can find where to store our files

```
getwd()
```

Downloading data on crimes in Lewisham from January 2019 to December 2019

The police service responsible for the Greater London area (excluding City of London) is the Metropolitan police service. Therefore for the Lewisham borough, we will be looking into downloading the data information on the Metropolitan police. *We firstly go to the link https://data.police.uk/data/ and click on the custom download tab.* We select a time range from January 2019 to December 2019 *We select the Metropolitan Police as that is the service used in Lewisham* Only the crime data data set is selected. *We then generate the file the press the* ***Download now*** *button.* Once we unzip the files, there should be 12 folders with a corresponding date such as 2019-01 for January 2019, or 2019-02 for February 2019 In each folder there should be a file such as `2019-01-metropolitan-street.csv` in 2019-01

Installing the packages

Before we start loading the street crime data into R using the read_csv(), we want to install the `readr`, `dplyr`, `devtools` and `ggmap` packages, and then load them into R. We can achieve this by using the install.packages() function such as below

```
install.packages('readr')
install.packages('dplyr')
if(!requireNamespace("devtools")) install.packages("devtools")
# installs devtools if not installed yet
devtools::install_github("dkahle/ggmap")
# installs ggmap from GitHub
```

To use `ggmap` in paticular, the `devtools` package is required so that we can download the GitHub version of the package. As we are on the latest version of R (3.6.2), tidyverse does not need to be installed to get ggplot2.

We next then load all the installed packages to Rstudio by using the following code:

```
library(readr)
library(dplyr)
library(devtools)
library(ggmap)
```

Loading the crime data onto RStudio

We then want to read the crime data onto RStudio. We do this by using the read_csv function.

```
police1 <- read_csv("2019-01-metropolitan-street.csv")
police2 <- read_csv("2019-02-metropolitan-street.csv")
police3 <- read_csv("2019-03-metropolitan-street.csv")
```

```
police4 <- read_csv("2019-04-metropolitan-street.csv")
police5 <- read_csv("2019-05-metropolitan-street.csv")
police6 <- read_csv("2019-06-metropolitan-street.csv")
police7 <- read_csv("2019-07-metropolitan-street.csv")
police8 <- read_csv("2019-08-metropolitan-street.csv")
police9 <- read_csv("2019-09-metropolitan-street.csv")
police10 <- read_csv("2019-10-metropolitan-street.csv")
police11 <- read_csv("2019-11-metropolitan-street.csv")
police12 <- read_csv("2019-12-metropolitan-street.csv")
```

We dont want to have to work with multiple tibbles, so we bind the data into one tibble so that all the information will be in one place

```
policex <- bind_rows(police1,police2,police3,police4,police5,police6,police7,police8,police9,police10,police11,police12)
```

Now we want to create another tibble but one that only includes the variables `Month`, `Longitude`, `Latitude` and `Crime type`. It is quite difficult to use variables with spaces in RStudio so we change the `Crime type` variable to `Crimetype` before selecting the variables to use for our now tibble.

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

```
## # A tibble: 1,108,042 x 4
##    Month   Longitude Latitude CrimeType
##    <chr>       <dbl>    <dbl> <chr>
##  1 2019-01    -0.710     50.8 Violence and sexual offences
##  2 2019-01     0.140     51.6 Anti-social behaviour
##  3 2019-01     0.140     51.6 Burglary
##  4 2019-01     0.141     51.6 Burglary
##  5 2019-01     0.141     51.6 Drugs
##  6 2019-01     0.136     51.6 Other theft
##  7 2019-01     0.137     51.6 Other theft
##  8 2019-01     0.140     51.6 Vehicle crime
##  9 2019-01     0.140     51.6 Vehicle crime
## 10 2019-01     0.141     51.6 Vehicle crime
## # ... with 1,108,032 more rows
```

Getting location coordinates with geocode()

We now want to find plot data for `Lewisham` and we can do this by using the `geocode()` from `ggmap` to get the longitude and latitude for the area.

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

```
## Source : http://www.datasciencetoolkit.org/maps/api/geocode/json?address=lewisham
```

```
## # A tibble: 1 x 2
##      lon   lat
##    <dbl> <dbl>
## 1 -0.018  51.5
```

We should now have the latitude and longtitude information for Lewisham. We now want to get a bounding area for the map. I ended up using the below coordinates.

```
bbox <- c(-0.038, 51.45, 0.018, 51.47)
```

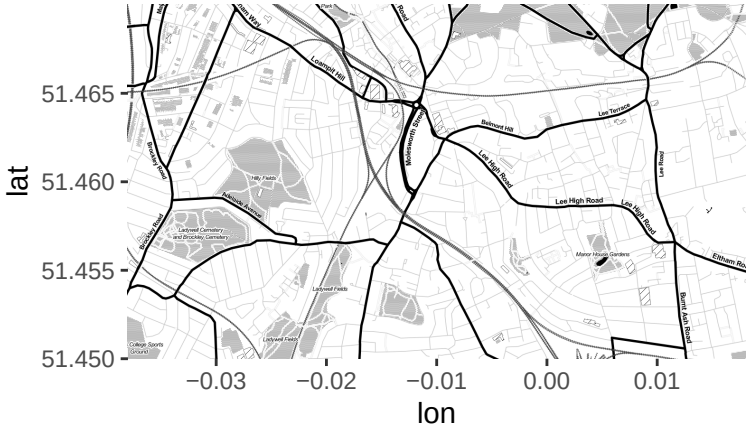
Downloading a map

We are able to use the function get_map() to get a raster object with the map from Stamen Maps

```
LewishamMap <- get_map(location = bbox, source = "stamen", maptype = "toner")
```

We can then display the map by using the `ggmap()` function:

```
ggmap(LewishamMap)
```



Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL

Putting the data onto the map

We now want to overlay the crime information onto the actual map.

Filtering the data

We only want to keep the rows with longitudes and latitudes that fall within the map, we can use the bounding box `bbox` to do this

```
crimeLewisham <- filter(policex, bbox[1] <= Longitude, Longitude <= bbox[3],
                  bbox[2] <= Latitude,  Latitude  <= bbox[4])
crimeLewisham
```

```
## # A tibble: 9,999 x 4
##    Month   Longitude Latitude CrimeType
##    <chr>       <dbl>    <dbl> <chr>
##  1 2019-01   0.0141      51.5 Burglary
##  2 2019-01   0.0131      51.5 Vehicle crime
##  3 2019-01   0.00959     51.5 Vehicle crime
##  4 2019-01   0.0145      51.5 Vehicle crime
##  5 2019-01   0.0137      51.5 Vehicle crime
##  6 2019-01   0.0176      51.5 Other crime
##  7 2019-01   0.0114      51.5 Anti-social behaviour
##  8 2019-01   0.0114      51.5 Anti-social behaviour
##  9 2019-01   0.0127      51.5 Bicycle theft
## 10 2019-01   0.0114      51.5 Bicycle theft
## # ... with 9,989 more rows
```

There is now 9999 observations which fall into the region selected, which is a lot less than 1108042. We now want to look at the frequency for each type of crime, which we do by using the following command:

```
table(crimeLewisham$CrimeType)
```

```
##
##        Anti-social behaviour                Bicycle theft
##                         1885                          162
```

```
##                     Burglary    Criminal damage and arson
##                          800                          572
##                        Drugs                  Other crime
##                          389                          108
##                  Other theft        Possession of weapons
##                          985                           85
##                Public order                      Robbery
##                          554                          377
##                  Shoplifting        Theft from the person
##                          452                          297
##               Vehicle crime Violence and sexual offences
##                         1048                         2285
```

We want to focus on the following crimes because they are the most commonly occuring crimes in the area chosen: *Anti-social behaviour* Other theft *Vehicle crime* Violence and sexual offences

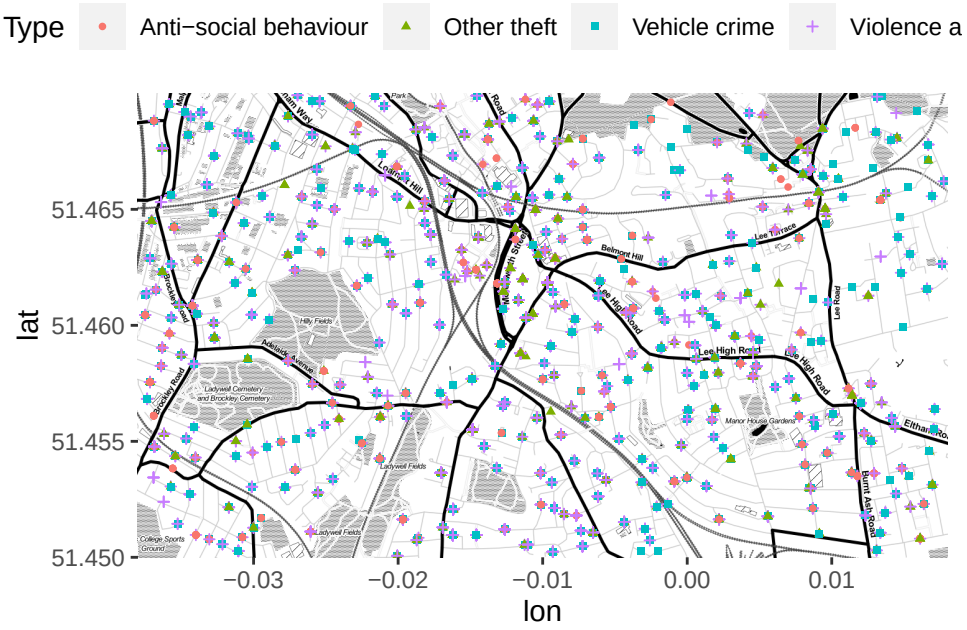
We use the `filter` function to restrict the table to only these four crime types:

```
crimeLewisham %>%
  filter(CrimeType %in% c("Anti-social behaviour", "Violence and sexual offences",
                          "Other theft", "Vehicle crime")
  ) -> crimeLewisham
table(crimeLewisham$CrimeType)
```

```
##
##        Anti-social behaviour                  Other theft
##                         1885                          985
##               Vehicle crime Violence and sexual offences
##                         1048                         2285
```

After this we can now put the locations of the selected crime types onto the map we are using.

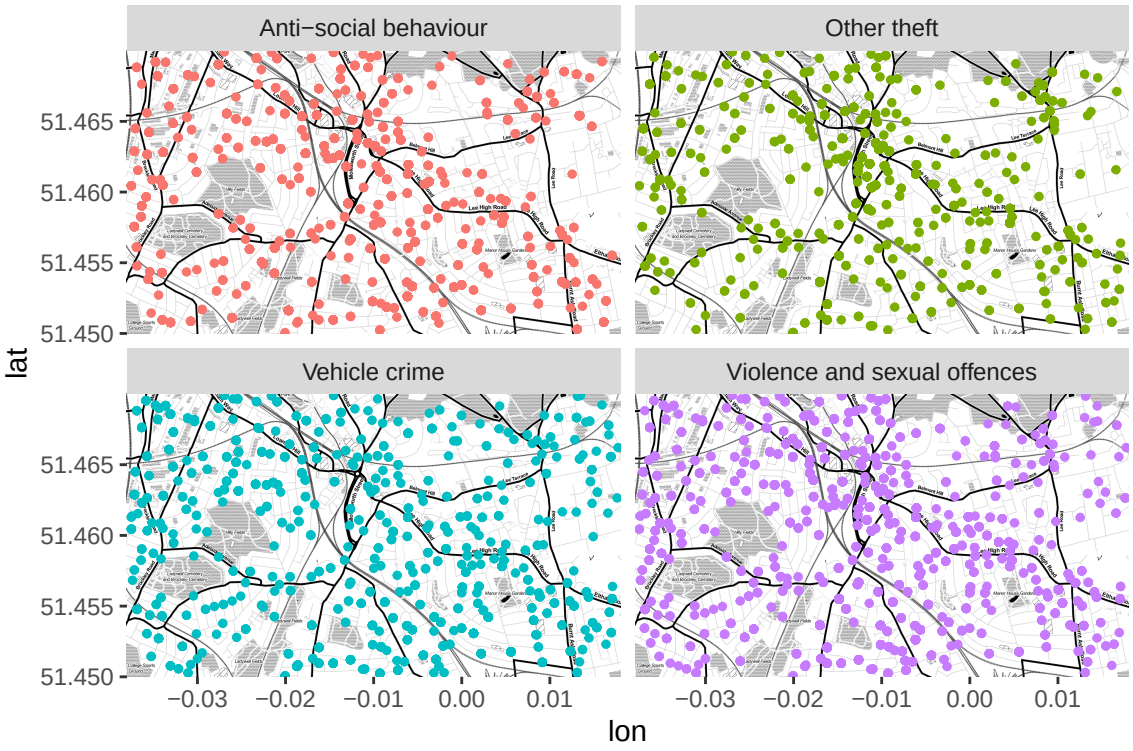
```
MapPoints <- ggmap(LewishamMap) +
  geom_point(aes(x = Longitude, y = Latitude, colour = CrimeType, shape=CrimeType),
             size=1, data = crimeLewisham) +
  theme(legend.position = "top")

MapPoints
```



Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL

We can also display the crime types in seperate maps so that the data can be displayed more clearly.

```
MapCrimes <- ggmap(LewishamMap) +
  geom_point(aes(x = Longitude, y = Latitude, colour = CrimeType),
             size=1, data = crimeLewisham) +
  theme(legend.position = "none") +
  facet_wrap(~ CrimeType)

MapCrimes
```



Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL

Fish Landings data

The goal of this task is to look through Fish Landings data given to us from the website https://www.gov.uk/government/statistical-data-sets/uk-and-foreign-vessels-landings-by-uk-port-and-uk-vessel-landings-abroad

Loading the data into RStudio

The .ods Excel file contains 9 sheets, but we are only interested in the Pubishled_dataset_2014 to Published_dataset_2019 files. *The numerical values of the last 3 columns are firstly displayed at their full precision so that we have reliable numerical information.* We then save each file seperately as a `csv` file so that we are able to load them into RStudio by using `read_csv()`

```
FishLanding2014 <- read_csv("Published_dataset_2014.csv")
FishLanding2015 <- read_csv("Published_dataset_2015.csv")
FishLanding2016 <- read_csv("Published_dataset_2016.csv")
FishLanding2017 <- read_csv("Published_dataset_2017.csv")
FishLanding2018 <- read_csv("Published_dataset_2018.csv")
FishLanding2019 <- read_csv("Published_dataset_2019.csv")
```

The above information shows the variables that will be loaded along with their data type.

Working with the fish data in RStudio

###Changing the 2019 variable names We firstly want to change the variable names of the 2019 file as some of them are different to the variable names in the 2014-2018 files.

```
FishLanding2019 <- rename(FishLanding2019, 'Live weight (tonnes)'='Sum of Live weight (tonnes)', 'Landed weight (tonnes)'='Sum of Landed weight (tonnes)', 'Value(£000s)'='Sum of Value(£)')
```

Binding the rows together

Before binding the rows together, we create an additional column to every tibble called `Test` which adds the corresponding year to each tibble so that we can create a new `Time` variable later on.

```
FishLanding2014 <-mutate(FishLanding2014, Test = 2014)
FishLanding2015 <-mutate(FishLanding2015, Test = 2015)
FishLanding2016 <-mutate(FishLanding2016, Test = 2016)
FishLanding2017 <-mutate(FishLanding2017, Test = 2017)
FishLanding2018 <-mutate(FishLanding2018, Test = 2018)
FishLanding2019 <-mutate(FishLanding2019, Test = 2019)
```

After this we now bind the tibbles together and add an additional column named year to link the dataframes back to their original tibble.

```
landings <- bind_rows(FishLanding2014,FishLanding2015,FishLanding2016,FishLanding2017,FishLanding2018,FishLanding2019, .id="Year")
```

We now add an additional column to the tibble `landings` called `Time` which has the class `Date`. We want to use dates such as `2014-01-15` so each observation has a middle of the month date corresponding to it. We also start removing spaces in many of the variables so that they are easier to work with.

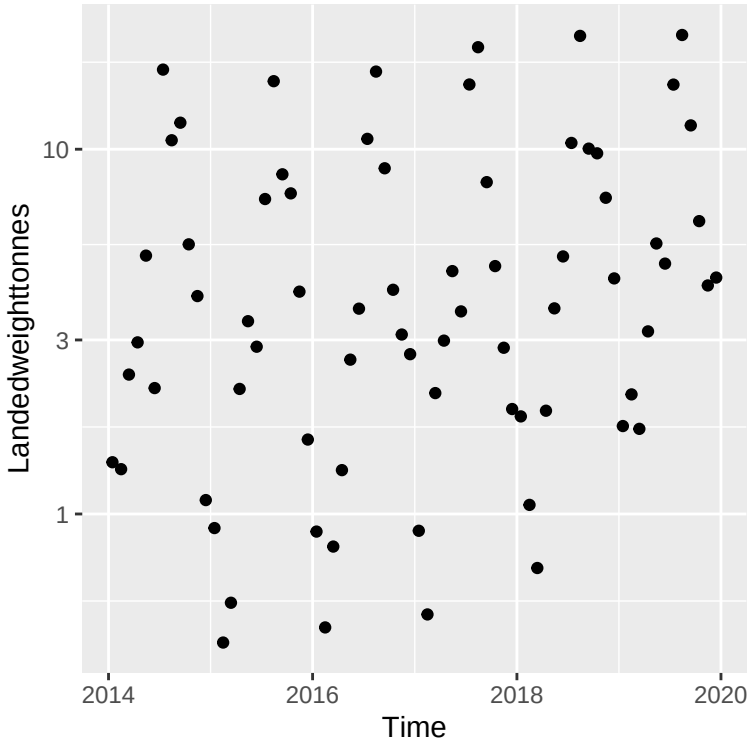
Data on Whitby: Lobsters caught by vessels 10m and under

We now want to filter out data in the tibbles of Lobstsers caught in Whitby by vessels 10m and lower. To do this we firstly use the `filter()` function to filter out the data on landings

```
WhitbyU10mLobs <- filter(landings, PortofLanding == "Whitby", LengthGroup =="10m&Under", Species == "Lobsters")
```

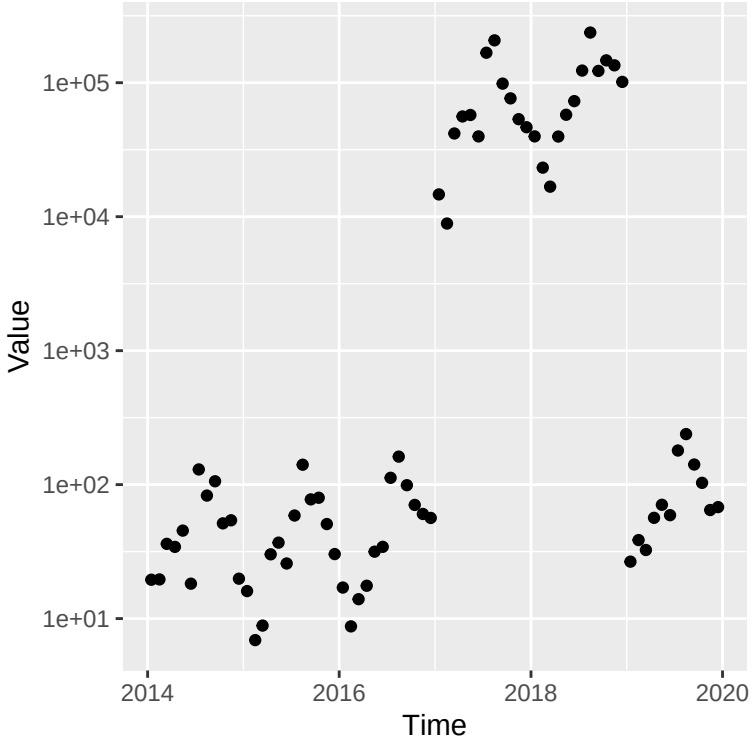
We now want to create a plot of `Landed weight` against `Time` and `Value` against `Time`. We do this using the `ggplot()` and `geom_point()` functions firstly for `Landed weight`:

```
LandingPlot1 <-ggplot(data= WhitbyU10mLobs) + geom_point(mapping = aes(x = Time, y = Landedweighttonnes)) + scale_y_log10()
LandingPlot1
```



And then `Value`

```
LandingPlot2 <-ggplot(data= WhitbyU10mLobs) + geom_point(mapping = aes(x = Time, y = Value)) + scale_y_
LandingPlot2
```



Correcting erros

As seen in the Value table, there is an inconsistency with the units for Value. We need to check this by checking the mean of the values by using the `summarise()` and `group_by()` functions:

```
Correction <-WhitbyU10mLobs %>%
  group_by(Year) %>%
  summarise(Value = mean(Value))

Correction
```

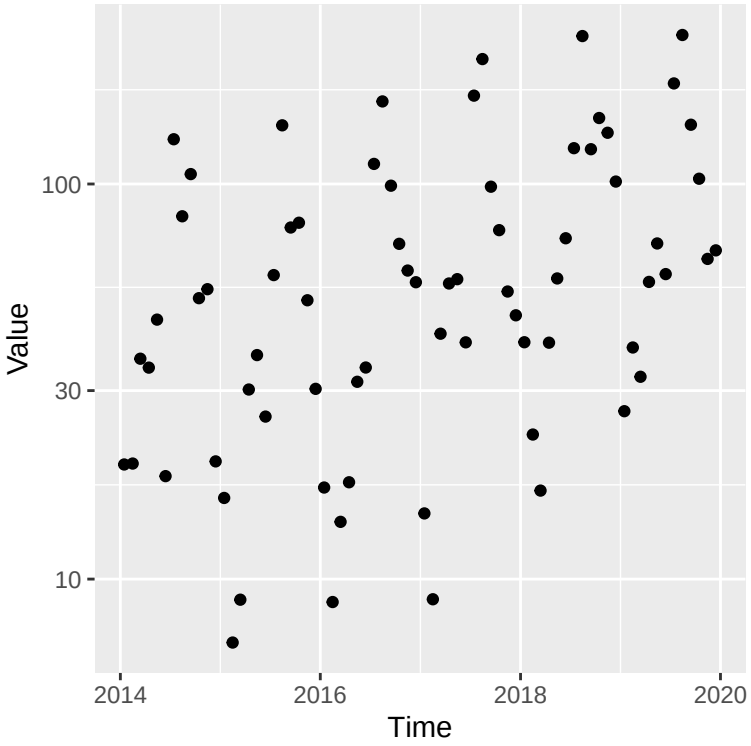
```
## # A tibble: 6 x 2
##   Year    Value
##   <chr>   <dbl>
## 1 1        51.4
## 2 2        46.9
## 3 3        57.0
## 4 4     72323.
## 5 5     92975.
## 6 6        89.9
```

As we can see, the means for the 2017 and 2018 values are much larger than the other year values. To correct this we divide all the values in the `Value` column by 1000 in the 2017 and 2018 tibbles to get these values to normal.

```
FishLanding2014 <- read_csv("Published_dataset_2014.csv")
FishLanding2015 <- read_csv("Published_dataset_2015.csv")
FishLanding2016 <- read_csv("Published_dataset_2016.csv")
FishLanding2017 <- read_csv("Published_dataset_2017.csv")
FishLanding2018 <- read_csv("Published_dataset_2018.csv")
FishLanding2019 <- read_csv("Published_dataset_2019.csv")

FishLanding2019 <- rename(FishLanding2019, 'Live weight (tonnes)'='Sum of Live weight (tonnes)', 'Landed weight (tonnes)'='Sum of Landed weight (tonnes)')
FishLanding2019 <- rename(FishLanding2019, 'Value(£000s)'='Sum of Value(£)')

FishLanding2014 <-mutate(FishLanding2014, Test = 2014)
FishLanding2015 <-mutate(FishLanding2015, Test = 2015)
FishLanding2016 <-mutate(FishLanding2016, Test = 2016)
FishLanding2017 <-mutate(FishLanding2017, Test = 2017)
FishLanding2018 <-mutate(FishLanding2018, Test = 2018)
FishLanding2019 <-mutate(FishLanding2019, Test = 2019)

FishLanding2017 <-rename(FishLanding2017, "Value"="Value(£000s)")
FishLanding2018 <-rename(FishLanding2018, "Value"="Value(£000s)")

FishLanding2017 <-mutate(FishLanding2017, Value = Value/1000)
FishLanding2018 <-mutate(FishLanding2018, Value = Value/1000)

FishLanding2017 <-rename(FishLanding2017, "Value(£000s)"="Value")
FishLanding2018 <-rename(FishLanding2018, "Value(£000s)"="Value")

landings <- bind_rows(FishLanding2014,FishLanding2015,FishLanding2016,FishLanding2017,FishLanding2018,FishLanding2019, .id="Year")
landings <-rename(landings, 'MonthLanded'='Month Landed')
landings <-mutate(landings, Time = as.Date(paste(Test, MonthLanded, 15, sep = "-")))
landings <-rename(landings, 'PortofLanding'='Port of Landing')
landings <-rename(landings, "LengthGroup"="Length Group")
landings <-rename(landings, "Landedweighttonnes"="Landed weight (tonnes)")
landings <-rename(landings, "Value"="Value(£000s)")
landings <-rename(landings, "SpeciesGroup"="Species Group")
WhitbyU10mLobs <- filter(landings, PortofLanding == "Whitby", LengthGroup =="10m&Under", Species == "Lobsters")
```

Now when we check the Correction tables and the second group, the values should be normal in comparison to the others

```
LandingPlot2 <-ggplot(data= WhitbyU10mLobs) + geom_point(mapping = aes(x = Time, y = Value))
LandingPlot2 <- LandingPlot2 + scale_y_log10()
LandingPlot2
```




```
Correction <-WhitbyU1OmLobs %>%
  group_by(Year) %>%
  summarise(Value = mean(Value))

Correction
```

```
## # A tibble: 6 x 2
##   Year  Value
##   <chr> <dbl>
## 1 1      51.4
## 2 2      46.9
## 3 3      57.0
## 4 4      72.3
## 5 5      93.0
## 6 6      89.9
```

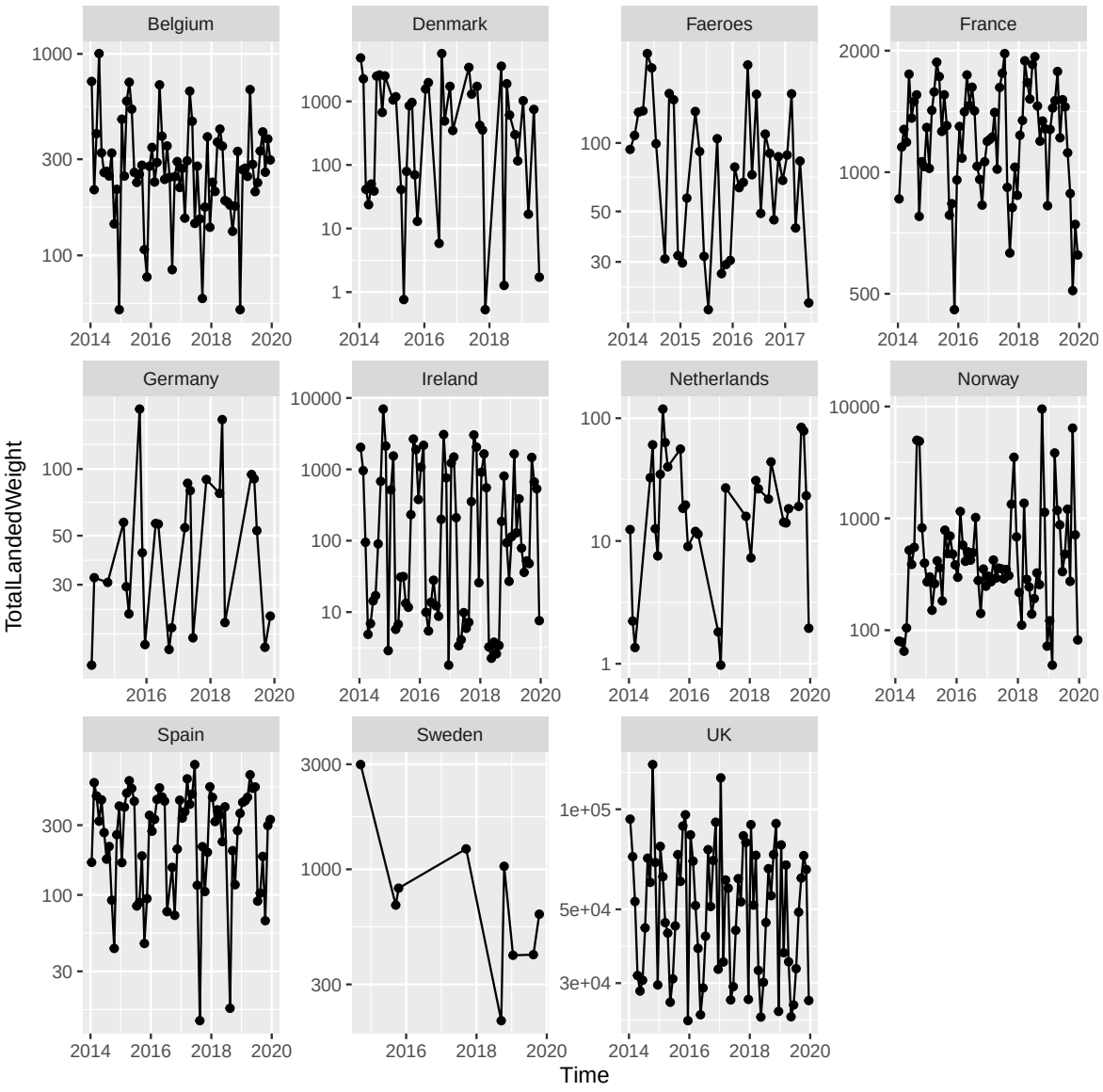
Landed weight by Vessel Nationality

We now want to create a new tibble called `LandWeiByVessNat` Which has 3 variables: Time, Vessel Nationality and TotalLandedWeight, where the last variable shows the total landed weight of each nationality during each month. We firstly rename the Vessel Nationality column to not have a space before proceeding:

```
landings<- rename(landings, 'VesselNationality'='Vessel Nationality')

LandWeiByVessNat <-select(landings, Time, VesselNationality)
LandWeiByVessNat <-landings %>%
  group_by(Time,VesselNationality) %>%
  summarise(TotalLandedWeight = sum(Landedweighttonnes))
```

After this we plot the `Time` variable against `TotalLandedWeight`

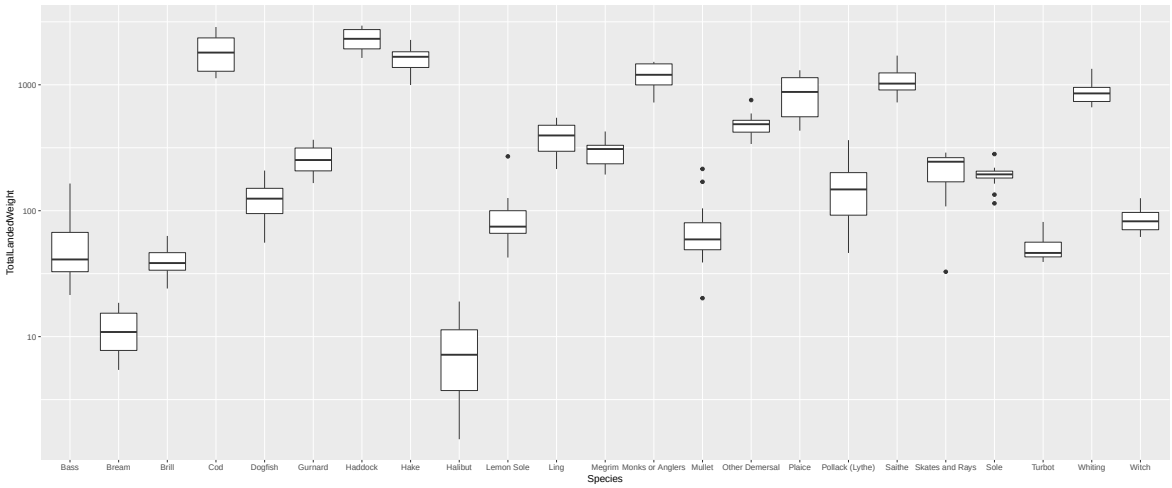
```
LandingPlot3 <-ggplot(data= LandWeiByVessNat) + geom_point(mapping = aes(x = Time, y = TotalLandedWeight))
LandingPlot3 <-LandingPlot3 + scale_y_log10() +geom_line(mapping = aes(x = Time, y = TotalLandedWeight)) +facet_wrap(~ VesselNationality, scales = 'free')
LandingPlot3
```



Landing data of Demersals in December and January dates

We want to plot in a boxplot the landing data of the `Species Group` Demersal in December and January months. We want to use the `summarize()`, `filter()` and `group_by()` functions as seen in previous sections

```
Demersal1 <- select(landings, Time, MonthLanded, Species, SpeciesGroup,Landedweighttonnes)
Demersal1 <- filter(Demersal1,SpeciesGroup =="Demersal", MonthLanded ==12 | MonthLanded==1)
Demersal2 <-Demersal1 %>%
  group_by(Time,Species) %>%
  summarise(TotalLandedWeight = sum(Landedweighttonnes))

LandingPlot4 <-ggplot(Demersal2, aes(x=Species, y=TotalLandedWeight)) + geom_boxplot()
LandingPlot4 <-LandingPlot4 + scale_y_log10()
LandingPlot4
```



This shows data on all the Species within the Species Group Demersal that were caught in the months January and December only from 2014-2019.

References

Home Office (2020). Data.Police.uk website.

Government UK: UK and foreign vessels landings but UK port and vessel landings abroad: 2014 to 2018 and 2019 (2020). Gov.uk website.

Github; help with wrapping long code (2020) github.com website.