Practical Session 4: Statistical analysis in RStudio

Welcome to your fourth and final practical R session. In this session we are going to explore the use of statistical tests in R to understand how you can question your data in several different ways to gain an understanding of trends and corelations. Below are four main sections that will be used to explore this area.

(1) Catch up with sessions 1, 2 & 3
(2) Import seal data, set up a project, check for normality.
(3) How can we test for corelations/associations?
(4) Comparing seal species population variation

SECTION 1 – Catch-up with sessions 1, 2 and 3

If you were able to attend the masterclass sessions, you will have been part of the introductory session to import data to R. Below is a short synopsis of the session to bring back memories of the session and then begin todays main aims.

1. Download the data1.xlsx from blackboard
2. Convert this into a cvs file.
3. Download R and Rstudio.
4. Start an R project called: "Practical_1".
5. Open a new R script to begin coding with.
6. Import your data by moving the data1.csv file to your project.
7. Import and name your data in RStudio [data1 <- read.csv("Data1.csv")]

If you have any trouble with the above process refer to your session 1 worksheet available from blackboard.

In session 2 we learnt about multiple data curation, plotting and assessment steps.

1. You learnt to use the # key to make notes.
2. You learnt to view your data and check it using str().
3. Use of the $ to specify data columns and convert data types.
4. How to make a data frame.
5. How to check for normality in your data using qqnorm() and shapiro.test().
6. Multiple ways to plot your data in RStudio and change some of the parameters.

If you have forgotten any of the above, please refer back to your practical worksheet for session 2. This has all of the tools and instruction to help you.

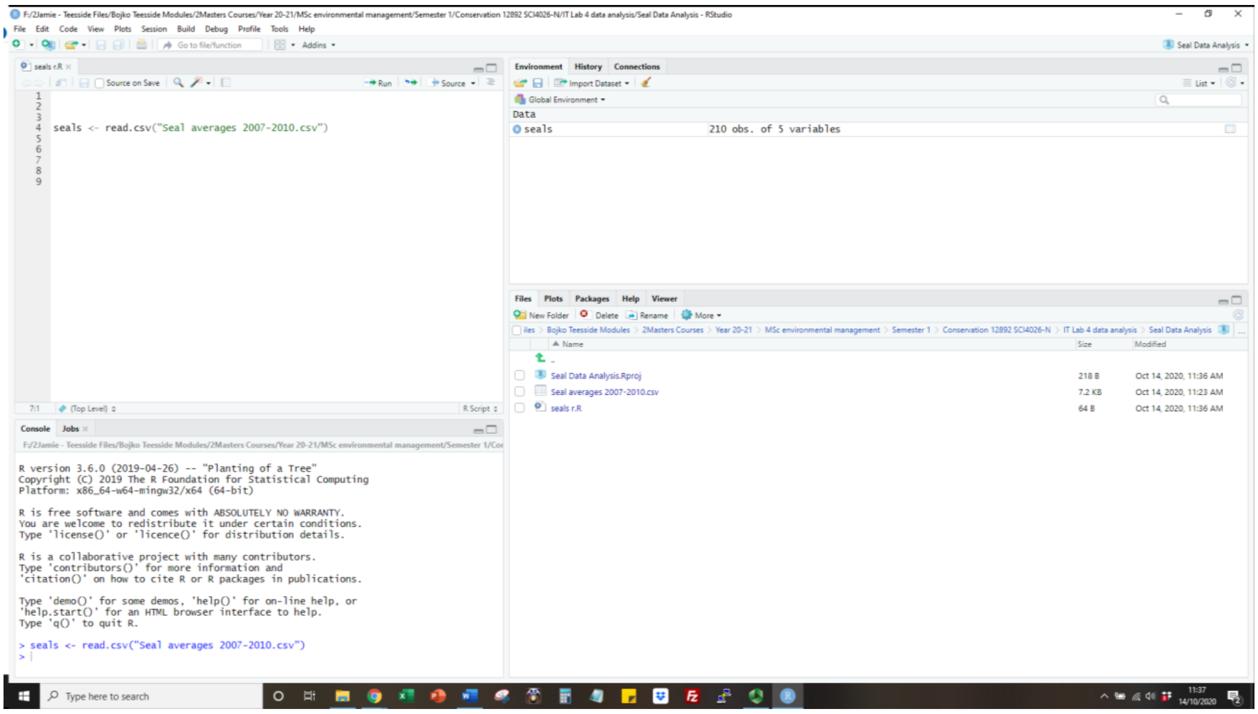
In session 3, we explored the use of ggplot2 to develop graphs of your data to increase our ability to interpret and understand it.

1. We learnt to install tools and libraries.
2. We learnt about various functions ggplot2 provides.
3. Finally, we plotted your data in several ways.

If you need a refresher session on these techniques, please download and go over practical session 3's workbook available on blackboard.

SECTION 2 – Import seal data, set up a project, and check for normality.

Using your R skills, acquired through the previous sessions, create a project called "Seal Data Analysis". Start a new R script. Import the data file 'Seal averages 2007-2010.csv' (available from blackboard) and call it 'seals'. Your screen should look like the one below:



F:/2Jamie - Teesside Files/Bojko Teesside Modules/2Masters Courses/Year 20-21/MSc environmental management/Semester 1/Conservation 12892 SCI4026-N/IT Lab 4 data analysis/Seal Data Analysis - RStudio

```r
seals <- read.csv("Seal averages 2007-2010.csv")
```

```
Data
seals                    210 obs. of 5 variables
```

```
R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> seals <- read.csv("Seal averages 2007-2010.csv")
> 
```

Name	Size	Modified
Seal Data Analysis.Rproj	218 B	Oct 14, 2020, 11:36 AM
Seal averages 2007-2010.csv	7.2 KB	Oct 14, 2020, 11:23 AM
seals r.R	64 B	Oct 14, 2020, 11:36 AM

Once you have set yourself up, you're ready to go!

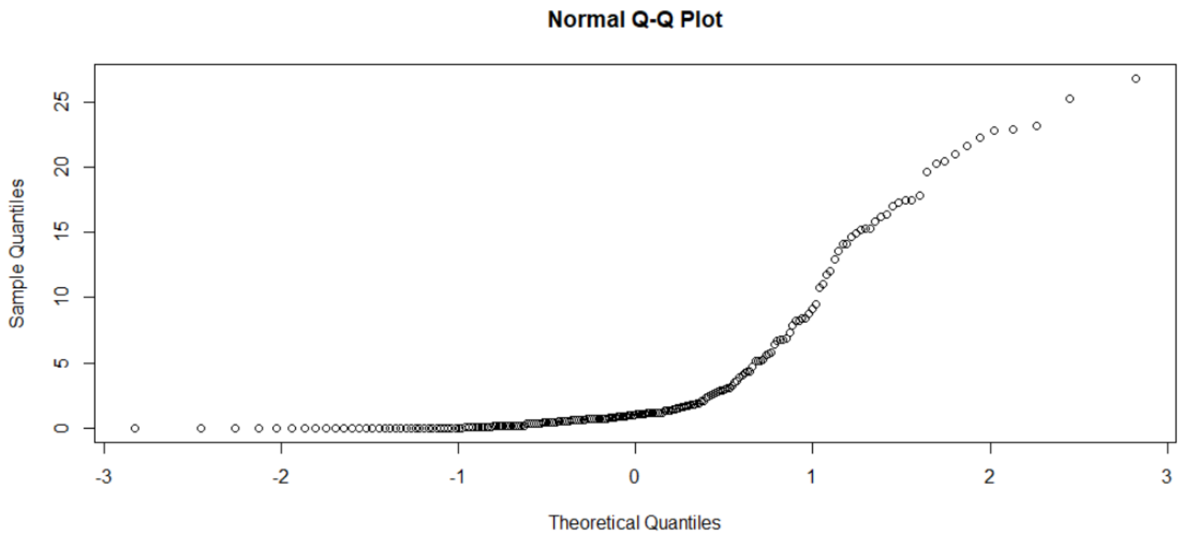
You will recall from previous analyses that we checked our data to see if it was normal or not. To do this, you need to apply two different methods. One is the 'qqnorm()' function and the other is the Shapiro test.

Check your different data columns to determine if your data are normal or not.

You will notice that we only have one data column with numerical information. This is the only column that we can test for normality.

Run qqnorm(seals$average.count)

Do you see the graph shown below?



You can see that this line is not linear, but instead is distinctly curved. This curvature suggests that the data are not normal. Now, let's check to see if the data are normal using the shapiro test.

What code will you use to use this test on your data? ______________________

What does your shapiro test result tell you? ______________________________

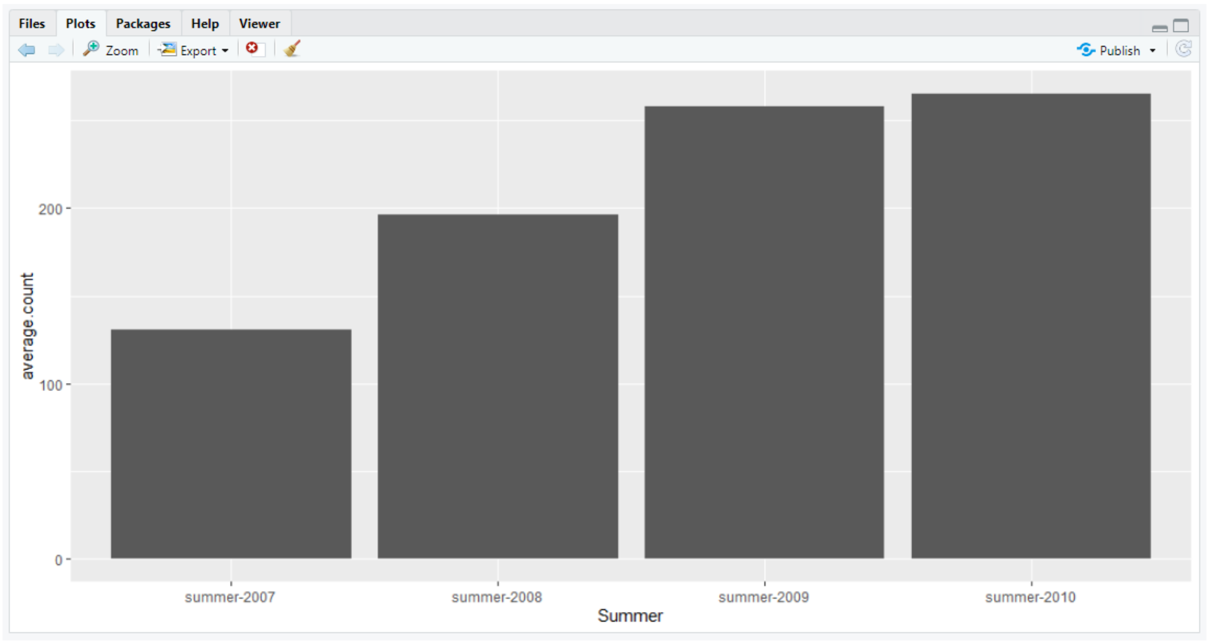
You should have received a W number and a p value. If your p value is <0.001, your data are not significant. If your data are not significant, you need to consider non-parametric statistical tests. Have a look on blackboard for some information relating to different non-parametric tests.

SECTION 3 – corelations and statistical associations

Now that you have your data imported to your new R project and you have tested your data for normality, you will know that your data are not normal. Due to last weeks R session, you will also be familiar with your data because you have plotted it in several ways. Your data should have 5 columns (Summer, Year.Month, Site, Species and average.count) All of these are factorial information other than the average.count columns, which is an integer or numerical.

You will be able to see that the Summer column includes four groups, or levels. These are summer-2007, summer-2008, summer-2009 and summer-2010.

Perhaps we want to know if the number of seals are significantly different for each year? In the past, we have seen a plot of these data (reminder below)



However, we do not know if these changes are significant or not.

To figure this out we can run some non-parametric tests. The first is a Kruskal-Wallis Rank Sum Test. This provides an overall significance for the data, providing a chi-squared interpretation and a p-value.

To run a Kruskal-wallis test, you need the function kruskal.test()

```
#This allows us to explore the overall difference between treatments.
kruskal.test(seals$average.count, seals$Summer)
```

What result do you get? ____

Does this suggest there is a significant difference? _________

Although the kruskal.test() gave us an answer for the overall data, we might want to have more detailed information. For this we need to use a pairwise wilcox test. This is displayed in R via the function: pairwise.wilcox.test().

pairwise.wilcox.test(seals$average.count, seals$Summer)

When you run this code, you will get the following output:

```
data:  seals$average.count and seals$Summer

            summer-2007 summer-2008 summer-2009
summer-2008 0.89        -           -
summer-2009 0.20        0.43        -
summer-2010 0.43        0.64        0.89
```

Further to this, you need to provide a p adjustment method, to avoid false positive results. We will select “BH” for this. See below. Run this yourself.

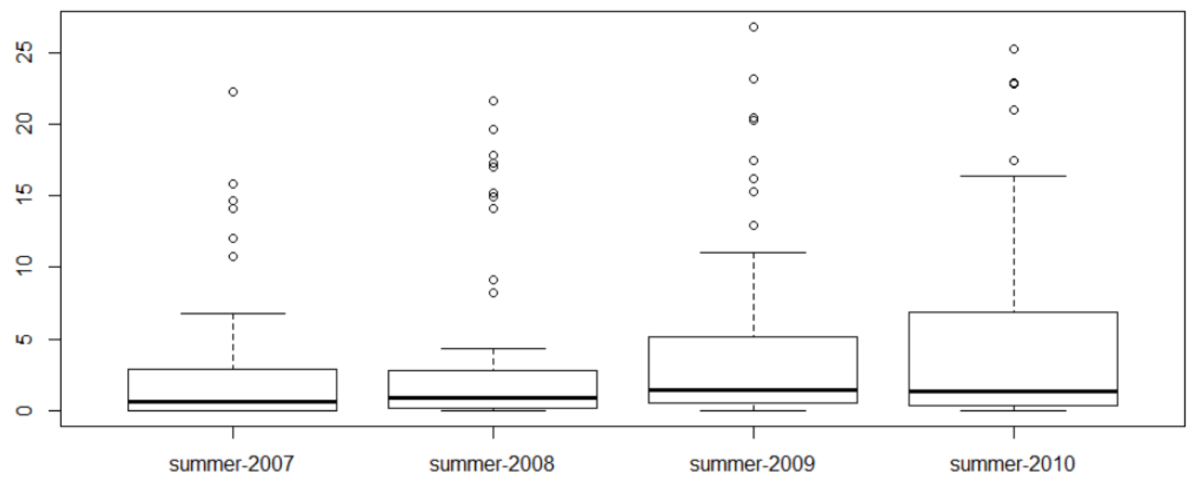
```
#This allows us to see the deatils of the kruskal test using a wilcox test
pairwise.wilcox.test(seals$average.count, seals$Summer, p.adjust.method = "BH")
```

How does your output change? Do the values remain the same?

The numbers in the table refer to p-values. These suggest whether the data are significant or not. Are any of your comparisons significant? To be significant the, p-value must be below 0.05, often represented as <0.05.

However, you may think this is odd. Why is there not a level of significance between the different years? To explore this a little, let’s change the way we look at our data.

The bar graph above suggested that there was a big difference between years 2007 and 2010, but we have seen this is not a significant difference using appropriate statistical tests. This is because we have not plotted any error bars onto this graph, so we miss out the underlying data. In reality this graph is better represented as a box plot – a plotting method that allows a better understanding of variance within your data. “plot(seals$Summer, seals$average.count)”



The y axis is the average count in this figure.

You can see in this figure, that although a greater average number of seals was counted in 2010, there were some high counts (represented by circles) in the summer of 2007. These levels of variance in the data are not well explored in a bar graph and need this information to be better represented in the box plot format to understood better.

Next, we might actually be interested in any differences from month to month, perhaps in 2007, to see which are the main months of seal presence and absence. To do this, type in the following code:

```
#explore differences between months in 2007
kruskal.test(seals$average.count[seals$Summer=="summer-2007"],
             seals$Year.Month[seals$Summer=="summer-2007"])
```

This code allows you to specify a year to work in. Here we specify 2007 using [seals$Summer=="summer-2007"].

Our kruskal.test() result is as follows:

```
data:  seals$average.count[seals$Summer == "summer-2007"] and seals$Year.Month[seals$Summer == "summer-2007"]
Kruskal-Wallis chi-squared = 1.3113, df = 2, p-value = 0.5191
```

When you run the pairwise test, you get the following:

```
#explore differences within these data
pairwise.wilcox.test(seals$average.count[seals$Summer=="summer-2007"],
                     seals$Year.Month[seals$Summer=="summer-2007"],
                     p.adjust.method = "BH")
```

.

```
         2007.Aug 2007.Jul
2007.Jul 0.63     -       
2007.Jun 0.63     0.63    
```

This suggests that there is no significance between the different months in 2007.

SESSION 4 – Comparing seal species abundance

Now that we have looked at how the whole number of seals are changing in population size over time, which appears non-significant, we are now going to look at the differences between seal species. To do this, we will be using the same techniques as described in the last section.

To explore the differences run a Kruskal.text() on your seal species and avaergae data as shown below.

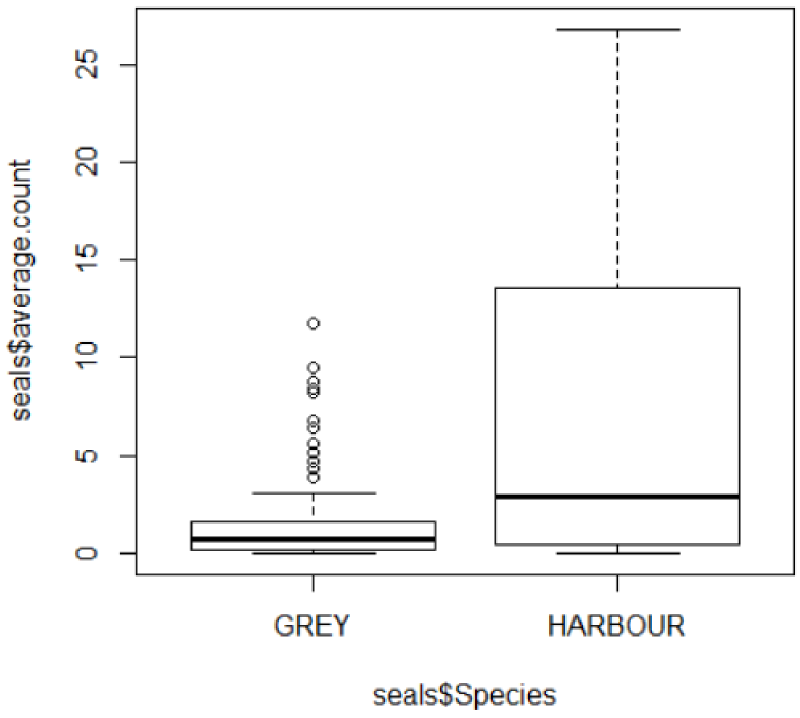
```
#Seal species comparison
kruskal.test(seals$average.count, seals$Species)
```

You should receive this output:

```
data:  seals$average.count and seals$Species
Kruskal-Wallis chi-squared = 18.66, df = 1, p-value = 1.562e-05
```

This means that there is a significant difference in the average counts of the two seal species over the four year data set! One seal is more abundant than the other. If you plot this data, you will see which.

```
plot(seals$average.count ~ seals$Species)
```



The Harbour seal is more common! It is also significantly more common than the grey species.

Can you find out if this is true for all of the years? Include the P-value.

Is it true for 2007?_________________

Is it true for 2008?_________________

Is it true for 2009?_________________

Is it true for 2010?_________________

In which years were the populations significantly different and in which years were the populations not significantly different?____

____

Hint: Use the code above to specify the year, which is in square brackets.

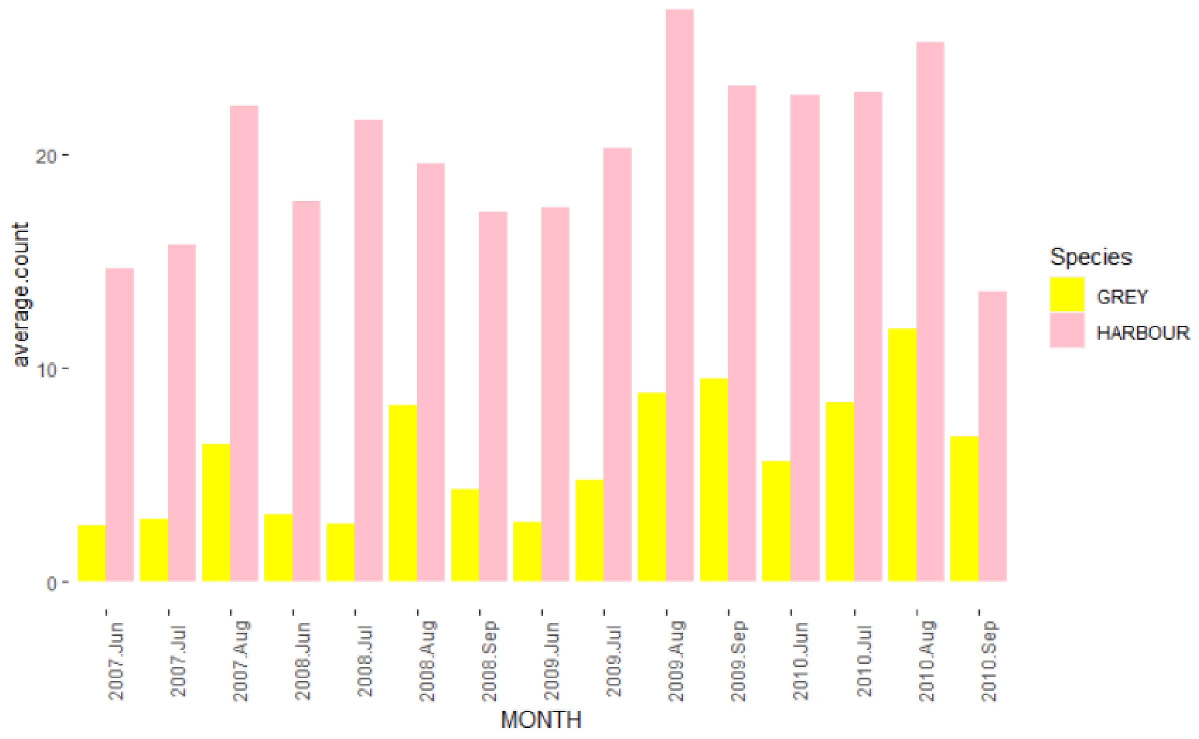
Can you plot each of these years as graphs to see what the data look like?

The next thing we will do is look at adding error bars. For this, we will draw upon your ggplot2 skills. Import and install the ggplot library and Rmic library as well as install both packages.

```
install.packages("Rmisc")
library(Rmisc)
install.packages("ggplot2")
library(ggplot2)
```

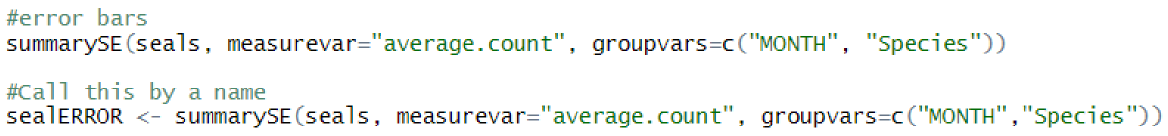
Take R code from your previous session (Session 3) to help you throughout this next test. To begin, we will use the code below, which was used to make the yellow and pink graph in the last session.

```
ggplot(seals, aes(fill = Species, x=MONTH, y=average.count)) +
  geom_bar(position="dodge", stat = "identity", width = 0.9) +
  scale_fill_manual(values=c("yellow", "pink")) +
  theme(panel.background = element_blank(), axis.text.x=element_text(angle=90))
```



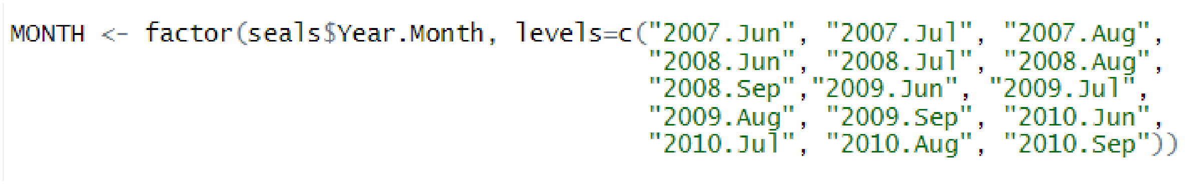
To develop the error bars, we will use the ggplot2 function, summarySE(). Later we will use the error bar plotting function geom_errorbar().

To begin with, we need to tell R what we think the error should reflect and which groups will see the variation. We do this by using the internal functions measurevar and groupvars. Once written in, you need to call this a new name for the development of the graph. See below:



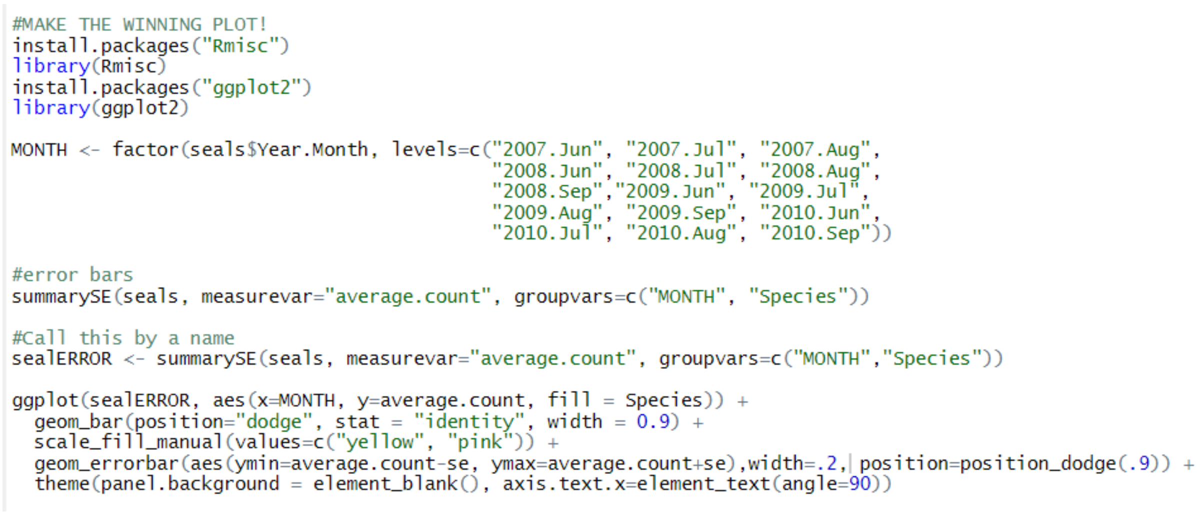
```
#error bars
summarySE(seals, measurevar="average.count", groupvars=c("MONTH", "Species"))

#Call this by a name
sealERROR <- summarySE(seals, measurevar="average.count", groupvars=c("MONTH","Species"))
```

Do you recall that we made a ‘MONTH’ allocation, to make sure our data were presented in the right order? Be sure to copy and paste this. A reminder is below:



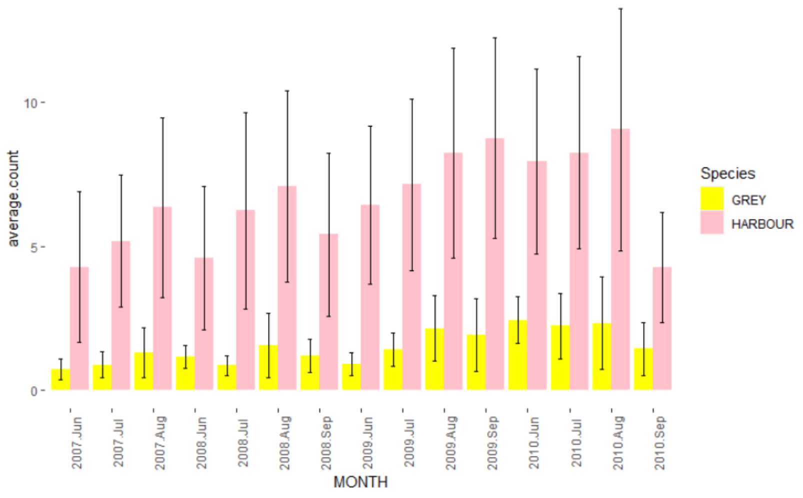
```
MONTH <- factor(seals$Year.Month, levels=c("2007.Jun", "2007.Jul", "2007.Aug",
                                            "2008.Jun", "2008.Jul", "2008.Aug",
                                            "2008.Sep","2009.Jun", "2009.Jul",
                                            "2009.Aug", "2009.Sep", "2010.Jun",
                                            "2010.Jul", "2010.Aug", "2010.Sep"))
```

Using these pieces of code, follow the example below to develop a graph with error bars.



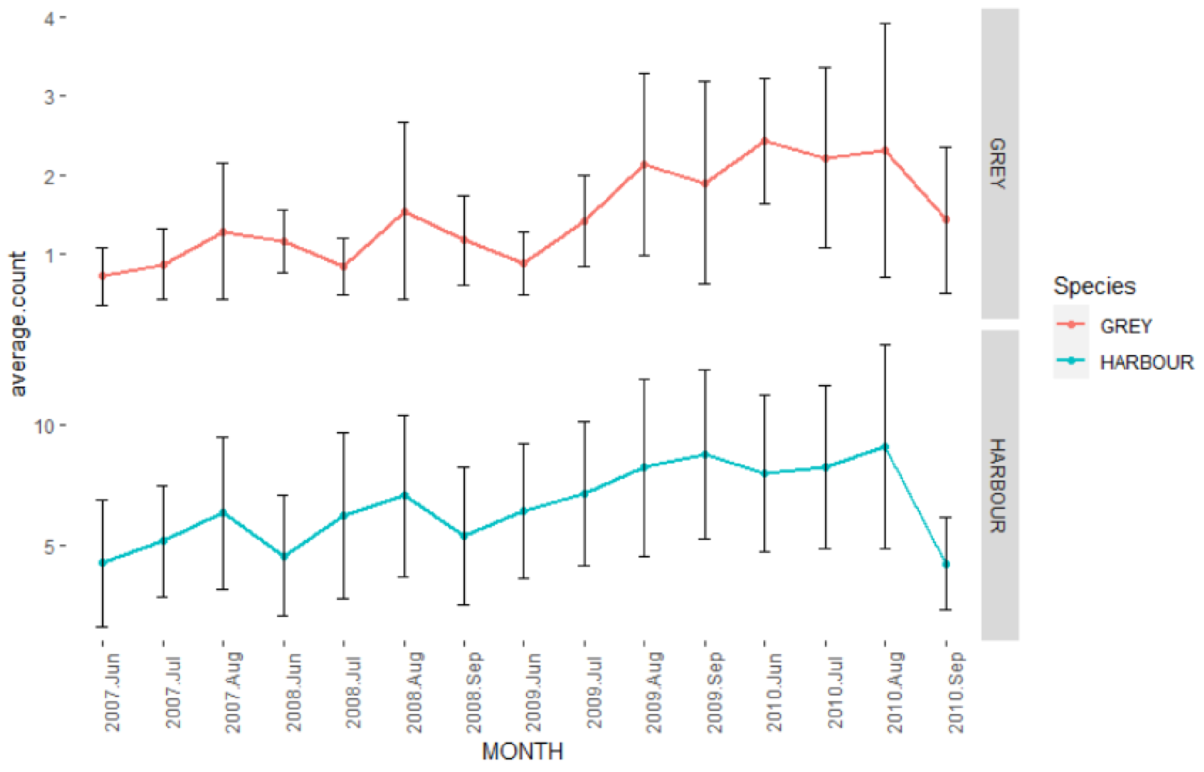
```
#MAKE THE WINNING PLOT!
install.packages("Rmisc")
library(Rmisc)
install.packages("ggplot2")
library(ggplot2)

MONTH <- factor(seals$Year.Month, levels=c("2007.Jun", "2007.Jul", "2007.Aug",
                                            "2008.Jun", "2008.Jul", "2008.Aug",
                                            "2008.Sep","2009.Jun", "2009.Jul",
                                            "2009.Aug", "2009.Sep", "2010.Jun",
                                            "2010.Jul", "2010.Aug", "2010.Sep"))

#error bars
summarySE(seals, measurevar="average.count", groupvars=c("MONTH", "Species"))

#Call this by a name
sealERROR <- summarySE(seals, measurevar="average.count", groupvars=c("MONTH","Species"))

ggplot(sealERROR, aes(x=MONTH, y=average.count, fill = Species)) +
  geom_bar(position="dodge", stat = "identity", width = 0.9) +
  scale_fill_manual(values=c("yellow", "pink")) +
  geom_errorbar(aes(ymin=average.count-se, ymax=average.count+se),width=.2, position=position_dodge(.9)) +
  theme(panel.background = element_blank(), axis.text.x=element_text(angle=90))
```

If you have copied this over correctly, you should get the following figure:



This graph provides us with good insight into the data, by providing error bars that show us how variable the data are using standard error.

Your final task is to apply all of your new R abilities to develop the graphic below! Use your ggplot2 skills and your seal data to do this. Get as close as you can and play with the R code to make a graph you like!



Happy plotting! Contact me at: J.Bojko@tees.ac.uk if you want the code to make this!