Final Project

STAT40620 Data Programming with R

Denis O'Riordan - 21201588

```r
knitr::opts_chunk$set(echo = TRUE)
```

Part 1: Analysis.

Choice of Data

According to the National Dairy Council, a semi-state body responsible for promoting milk and dairy products, Ireland exports almost 90% of its dairy output to 120 countries across the globe. Milk production and the value of it's exports have doubled since EU milk production quotas were removed in 2015, rising from €2 billion to over €4 billion currently. It is clear the importance of dairy production to the economy of Ireland. For this section, I am going to analyze data relevant to the above information and examine trends in the population of dairy cows in Ireland.

The data I have chosen is *AAA10 - Number of Cattle in June* data from the website of the Central Statistics Office (CSO). This dataset contains the data of each year from 2015 to 2021 and is split by county or region for the Republic Of Ireland. It contains the counts, in 000's, of different cattle type e.g Dairy cows, Bulls, Cattle: 2 years and over etc.. This data is freely available and compiled using information form the Department of Agriculture, Food and Marine (DAFM).

Reading in Data

The dataset is saved to my computer and loaded in its raw format into R.

```
data = read.csv(file=
                  'C:/Users/denis/Documents/STAT40620 Data Programming with R/Project/CattleFigures.csv',
                header=TRUE)
head(data)
```

```
##                ï..Statistic Year Type.of.Cattle    Region.and.County     UNIT
## 1 Number of Cattle in June 2015   Total cattle                State 000 Head
## 2 Number of Cattle in June 2015   Total cattle Northern and Western 000 Head
## 3 Number of Cattle in June 2015   Total cattle               Border 000 Head
## 4 Number of Cattle in June 2015   Total cattle              Leitrim 000 Head
## 5 Number of Cattle in June 2015   Total cattle                Sligo 000 Head
## 6 Number of Cattle in June 2015   Total cattle                Cavan 000 Head
##    VALUE
## 1 6963.5
## 2 1723.9
## 3  793.6
## 4   65.0
## 5  113.5
## 6  223.0
```

```
colnames(data)
```

```
## [1] "ï..Statistic"      "Year"              "Type.of.Cattle"
## [4] "Region.and.County" "UNIT"              "VALUE"
```

```
dim(data)
```

```
## [1] 4144    6
```

Description of Data

This dataset contains 4144 rows(observations) and six columns (variables). The columns are as follows:

- **ï..Statistic:** The statistic for each row, only takes the value 'Number of Cattle in June'
- **Year:** - Year of figures, ranging from 2015 to 2021
- **Type.of.Cattle:** A categorical variable of 16 different subsets (including total) of the Irish cattle population. Full list shown below
- **Region.and.County:** A categorical variable containing 26 counties of Rep. of Ireland and the NUT 2 and NUT 3 regions and an an overall 'state' value. Nomenclature of Territorial Units for Statistics (NUTS) were created by Eurostat in order to define territorial units for the production of regional statistics across the European Union. A full breakdown of the NUT regions and the counties in each is given in the link.
- **UNIT:** Unit of the value, only takes the value '000 Head' representing 1'000 cattle
- **VALUE** Number of cattle in 000's for that particular year, type and region or county.

```
str(data)
```

```
## 'data.frame':    4144 obs. of  6 variables:
##  $ ï..Statistic     : chr  "Number of Cattle in June" "Number of Cattle in June" "Number of Cattle in J
##  $ Year             : int  2015 2015 2015 2015 2015 2015 2015 2015 2015 2015 ...
##  $ Type.of.Cattle   : chr  "Total cattle" "Total cattle" "Total cattle" "Total cattle" ...
##  $ Region.and.County: chr  "State" "Northern and Western" "Border" "Leitrim" ...
##  $ UNIT             : chr  "000 Head" "000 Head" "000 Head" "000 Head" ...
##  $ VALUE            : num  6964 1724 794 65 114 ...
```

```
unique(data$Type.of.Cattle)
```

```
##  [1] "Total cattle"                    "Cows"
##  [3] "Dairy cows"                      "Other cows"
##  [5] "Bulls"                           "Cattle: 2 years and over"
##  [7] "Cattle male: 2 years and over"   "Cattle female: 2 years and over"
##  [9] "Cattle: 1-2 years"               "Cattle male: 1-2 years"
## [11] "Cattle female: 1-2 years"        "Cattle: under 1 year"
## [13] "Cattle male: under 1 year"       "Cattle female: under 1 year"
## [15] "Total cattle: male"              "Total cattle: female"
```

```
unique(data$Region)
```

```
##  [1] "State"                "Northern and Western" "Border"
##  [4] "Leitrim"              "Sligo"                "Cavan"
##  [7] "Donegal"              "Monaghan"             "West"
## [10] "Galway"               "Mayo"                 "Roscommon"
## [13] "Southern"             "Mid-West"             "Clare"
## [16] "Limerick"             "Tipperary"            "South-East"
## [19] "Carlow"               "Kilkenny"             "Wexford"
## [22] "Waterford"            "South-West"           "Cork"
## [25] "Kerry"                "Eastern and Midland"  "Dublin"
## [28] "Dublin and Mid-East"  "Kildare"              "Louth"
## [31] "Meath"                "Wicklow"              "Midland"
## [34] "Laois"                "Longford"             "Offaly"
## [37] "Westmeath"
```

Data adjustment and manipulation

Before any data analysis can be done, I must be satisfied the data being used is appropriately formatted for what I plan to analyse.

1. Convert to dataframe and remove needless columns

The data is converted into an R dataframe to aid with analysis as there are multiple data types.The columns *ï..Statistic* and *UNIT* are dropped as they provide no beneficial information and are constant throughout the data.

```
df = data.frame(data)
unique(df$ï..Statistic)
```

```
## [1] "Number of Cattle in June"
```

```
unique(df$UNIT)
```

```
## [1] "000 Head"
```

```
df = subset(df, select = -c(ï..Statistic,UNIT) )
colnames(df)
```

```
## [1] "Year"              "Type.of.Cattle"    "Region.and.County"
## [4] "VALUE"
```

2. Check and remove missing values

Missing data will provide a barrier to meaningful data analysis. It is best to check for missing data and to remove or adjust for these entries where possible.
In this dataset, the only missing data are of the `VALUE` variable when entries for `Year` = 2021 and `Region.and.County` is not 'state'. This is likely due to that county/regional data for 2021 was not available to the CSO when this data was published. I will drop all missing rows from my data and continue analysis on years 2015 - 2020 on a county and regional basis and can continue analysis on national trends from the full 2015 to 2021 data.

```
# what years contain missing data for value
unique(df[is.na(df$VALUE) == TRUE, "Year"])
```

```
## [1] 2021
```

```
# test if any of other columns contain missing data
any(is.na(c(df$Year, df$Type.of.Cattle,   df$Region.and.County)) == TRUE)
```

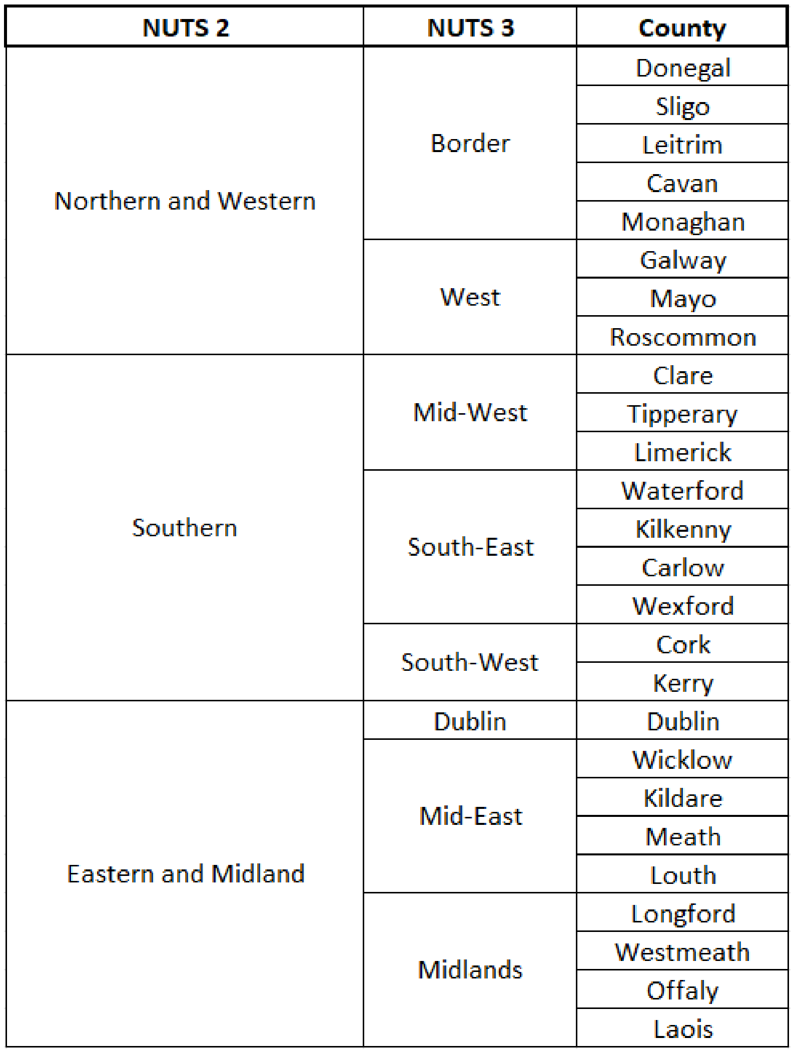
```
## [1] FALSE
```

```
#drop missing data
df = na.omit(df)
#check for missing data again
any(is.na(df) == TRUE)
```

```
## [1] FALSE
```

3. Separating Region and County

The counties of Ireland and their NUTS 2 and NUTS 3 regions are shown in the table below.



NUTS 2	NUTS 3	County
Northern and Western	Border	Donegal
		Sligo
		Leitrim
		Cavan
		Monaghan
	West	Galway
		Mayo
		Roscommon
Southern	Mid-West	Clare
		Tipperary
		Limerick
	South-East	Waterford
		Kilkenny
		Carlow
		Wexford
	South-West	Cork
		Kerry
Eastern and Midland	Dublin	Dublin
	Mid-East	Wicklow
		Kildare
		Meath
		Louth
	Midlands	Longford
		Westmeath
		Offaly
		Laois

These are also the possible values for *Region.and.County*. The only exception to this is Dublin and Mid-East is grouped as one region and the inclusion of a value for the entire country - ‘state’. For this analysis I want to look at county and regional trends separately. The NUTS 3 regions are too broad so I will be removing them. To perform this, I will create a new variable *Region* which will be the NUTS 2 region for a particular county in the data set and `NA` otherwise. Where *Region.and.County* = ‘state’ is given a *Region* of ‘state’. As *Region* is initialized with `NA`, all non-county rows will still contain `NA`, `na.omit()` can be used to remove these rows as we have already accounted for other missing values earlier. Now the data contains separate county and regional variables for analysis. *Region.and.County* column name is changed to *County* for clarity.

```r
# initialise region column
df$Region = NA

#loop over each row in the data
# the value for region.and.county is a county
# then the region column is assigned the appropirate NUTS2 region value
for (i in 1:nrow(df)) {
  # border counties
  if (df$Region.and.County[i] %in% c('Donegal','Sligo','Leitrim','Cavan','Monaghan')) {
    df$Region[i] = 'Border'
  }
  # west
  if (df$Region.and.County[i] %in% c('Galway', 'Mayo', 'Roscommon')) {
    df$Region[i] = 'West'
  }
  # mid-west
  if (df$Region.and.County[i] %in% c('Clare', 'Tipperary', 'Limerick')) {
    df$Region[i] = 'Mid-West'
  }
  # south-east
  if (df$Region.and.County[i] %in% c('Waterford', 'Kilkenny', 'Carlow', 'Wexford')) {
    df$Region[i] = 'South-East'
  }
  # south-west
  if (df$Region.and.County[i] %in% c('Cork','Kerry')) {
    df$Region[i] = 'South-West'
  }
  # dublin and mid-east
  if (df$Region.and.County[i] %in% c('Dublin','Wicklow','Kildare','Meath','Louth')) {
    df$Region[i] = 'Dublin and Mid-East'
  }
  # midlands
  if (df$Region.and.County[i] %in% c('Longford','Westmeath','Offaly','Laois')) {
    df$Region[i] = 'Midlands'
  }
  #state
  if (df$Region.and.County[i] %in% c('State')) {
    df$Region[i] = 'State'
  }
}
#rename columns
colnames(df) = c('Year', 'Type.of.Cattle', 'County' , 'VALUE', 'Region')
# drop rows with null values - rows where region is not populated
df = na.omit(df)
# check county and region variables are populated correctly and print dimension of new dataframe
dim(df)
```

```
## [1] 2608    5
```

```r
unique(df$County)
```

```
##  [1] "State"     "Leitrim"   "Sligo"     "Cavan"     "Donegal"   "Monaghan"
##  [7] "Galway"    "Mayo"      "Roscommon" "Clare"     "Limerick"  "Tipperary"
## [13] "Carlow"    "Kilkenny"  "Wexford"   "Waterford" "Cork"      "Kerry"
## [19] "Dublin"    "Kildare"   "Louth"     "Meath"     "Wicklow"   "Laois"
## [25] "Longford"  "Offaly"    "Westmeath"
```

```r
unique(df$Region)
```

```
## [1] "State"               "Border"              "West"
## [4] "Mid-West"            "South-East"          "South-West"
## [7] "Dublin and Mid-East" "Midlands"
```

The data frame is now reduced to 2608 rows.

4. Separating Type.of.Cattle and VALUE into separate variable

This piece of manipulation is to change the variables *Type.of.Cattle* and *VALUE* into separate variables storing the count of distinct cattle types.

Type.of.Cattle contains too many variables and the only two I am concerned with for this analysis is "Total cattle" and "Dairy cows". In effect, what I am doing is create two new variables *Total_Cattle* and *Dairy_Cows*, and for both, store the *VALUE* where *Type.of.Cattle* is 'Total cattle' and 'Dairy cows' respectively on one row for a given *Year* and *County*. *Type.of.Cattle* = "Total cattle" is used as a reference for the row to store both values. Similar to the previous step, I'll drop all rows with missing values as I only want rows with *Total_Cattle* and *Dairy_Cows* populated. Columns *Type.of.Cattle* and *VALUE* can also be dropped from the data once this is done. As there has been a lot of rows dropped from the data, I will reset the row index.

```r
# initialize column - Total_Cattle
df$Total_Cattle = NA
#loop over all rows
for (i in 1:nrow(df)) {
  # if type of cattle = "total cattle" assign the value to the total_cattle column
  if (df$Type.of.Cattle[i] == "Total cattle") {
    df$Total_Cattle[i] = df$VALUE[i]
  }
}

# initialize column - dairy_cows
df$Dairy_Cows = NA
#loop over all rows
for (i in 1:nrow(df)) {
  # check if type of cattle = "total cattle" to get reference row
  if (df$Type.of.Cattle[i] == "Total cattle") {
    # loop again to get the appropriate value for dairy cow population
    for (j in 1:nrow(df)) {
      if ((df$Year[i] == df$Year[j]) & (df$County[i] == df$County[j]) & (df$Type.of.Cattle[j] == 'Dairy co
        # assign the value to the dairy_cows column
        df$Dairy_Cows[i] = df$VALUE[j]
      }
    }
  }
}
# drop Type.of.Cattle, VALUE columns and rows containing NA
df = subset(df, select = -c(Type.of.Cattle, VALUE) )
df = na.omit(df)
#reset row index
row.names(df) <- NULL
```

5. Percentage Dairy column

Finally, I'll add one final column, the percentage of dairy cows of the total cattle for a given year and county. This requires no extra manipulation bar populating the column.

```r
# new column Percent_Dairy: percent of Dairy_Cows column of the Total_Cattle column
df$Percent_Dairy = NA
# loop over every row of the data
for (i in 1:nrow(df)) {
  # assign Percent_Dairy column value of Dairy_Cows as percent of Total_Cattle
  df$Percent_Dairy[i] = (df$Dairy_Cows[i]*100)/df$Total_Cattle[i]
}
```

Finalized Dataset

```
# structure, head and dimension of the data
str(df)
```

```
## 'data.frame':    163 obs. of  6 variables:
##  $ Year         : int  2015 2015 2015 2015 2015 2015 2015 2015 2015 2015 ...
##  $ County       : chr  "State" "Leitrim" "Sligo" "Cavan" ...
##  $ Region       : chr  "State" "Border" "Border" "Border" ...
##  $ Total_Cattle : num  6964 65 114 223 186 ...
##  $ Dairy_Cows   : num  1295.8 2.1 8.1 35.8 18.9 ...
##  $ Percent_Dairy: num  18.61 3.23 7.14 16.05 10.19 ...
##  - attr(*, "na.action")= 'omit' Named int [1:2445] 28 29 30 31 32 33 34 35 36 37 ...
##   ..- attr(*, "names")= chr [1:2445] "38" "41" "42" "43" ...
```

```
head(df)
```

```
##   Year   County Region Total_Cattle Dairy_Cows Percent_Dairy
## 1 2015    State  State       6963.5     1295.8     18.608458
## 2 2015  Leitrim Border         65.0        2.1      3.230769
## 3 2015    Sligo Border        113.5        8.1      7.136564
## 4 2015    Cavan Border        223.0       35.8     16.053812
## 5 2015  Donegal Border        185.5       18.9     10.188679
## 6 2015 Monaghan Border        206.7       34.4     16.642477
```

```
dim(df)
```

```
## [1] 163   6
```

I am now finished with initial data manipulation and adjustments. The result of this is a 163 row dataset with six columns.
There are three categorical columns - Year, County, Region.
There are three numeric columns - Total_Cattle, Dairy_Cows and Percent_Dairy. Total_Cattle, Dairy_Cows are the counts of total cattle and dairy cows for a given county and year. Percent_Dairy is the percentage of Total_Cattle that is made up of Dairy_Cows.

Exploratory Data analysis

For this analysis, I will be splitting the data into two distinct subsets. State figures will distort any statistics generated as it is the sum of figures of counties for that particular year. It is retained in a separate dataset as an easy way to look at national trends. The counties-only set will then be used to look at local/regional trends in the data.

```
# create dataset with only national data
df_state = df[df$County == 'State' ,]
#reset row index
row.names(df_state) <- NULL
# create dataset without national data
df_county = df[df$County != 'State' ,]
#reset row index
row.names(df_county) <- NULL
```

Section 1: State Trends

When looking at state trends in cattle population I am going to add three extra columns to the data. One for each numeric column and containing the difference in value between the current year and the previous year, it will be zero for 2015 as 2014 data isn't in the scope of the project.

```
# initialise columns
df_state$diff_T = 0
df_state$diff_D = 0
df_state$diff_P = 0

#populate columns with appropriate values
# loop over years
#calculate difference in value from the previous year
for (i in 2:7){
  df_state$diff_T[i] = df_state$Total_Cattle[i] - df_state$Total_Cattle[i-1]
  df_state$diff_D[i] = df_state$Dairy_Cows[i] - df_state$Dairy_Cows[i-1]
  df_state$diff_P[i] = df_state$Percent_Dairy[i] - df_state$Percent_Dairy[i-1]
}
```

National trends in total cattle population in Ireland 2015 - 2021

```
#summary of the total cattle population figures from 2015 to 2021
summary(df_state$Total_Cattle)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6964    7215    7314    7254    7354    7364
```

```
# data ordered by total cattle population from 2015 to 2021 in descending order
df_state[order(df_state$Total_Cattle, decreasing = TRUE),c('Year', 'Total_Cattle', 'diff_T')]
```

```
##   Year Total_Cattle diff_T
## 3 2017       7363.5  142.3
## 7 2021       7358.9   44.5
## 4 2018       7348.5  -15.0
## 6 2020       7314.4  105.8
## 2 2016       7221.2  257.7
## 5 2019       7208.6 -139.9
## 1 2015       6963.5    0.0
```

National trends in dairy cow population in Ireland 2015 - 2021

```
#summary of the dairy population figures from 2015 to 2021
summary(df_state$Dairy_Cows)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1296    1415    1481    1469    1536    1604
```

```
# data ordered by dairy cow population from 2015 to 2021 in descending order
df_state[order(df_state$Dairy_Cows, decreasing = TRUE),c('Year', 'Dairy_Cows', 'diff_D')]
```

```
##   Year Dairy_Cows diff_D
## 7 2021     1604.5   36.8
## 6 2020     1567.7   62.9
## 5 2019     1504.8   23.9
## 4 2018     1480.9   48.2
## 3 2017     1432.7   34.8
## 2 2016     1397.9  102.1
## 1 2015     1295.8    0.0
```

The above tables provide useful information on bovine demographics in Ireland.
In total population numbers, the lowest was recorded in the first year of the data, 2015 and the highest was recorded in the most recent year 2021. The largest change is an increase of 257'000 recorded between 2015 and 2016. From 2016 to 2021, however, the population seems relatively stable, varying between 7.2 and 7.4 million.
In contrast to this, the population of dairy cows has been increasing constantly by at least 20'000 per year. Ranging from under 1.3 million in 2015 to over 1.6 million in 2021.

```
#correlation between year and total number of cattle in Ireland
cor(df_state$Year, df_state$Total_Cattle)
```

```
## [1] 0.6558922
```

```
#correlation between year and total number dairy cows in Ireland
cor(df_state$Year, df_state$Dairy_Cows)
```

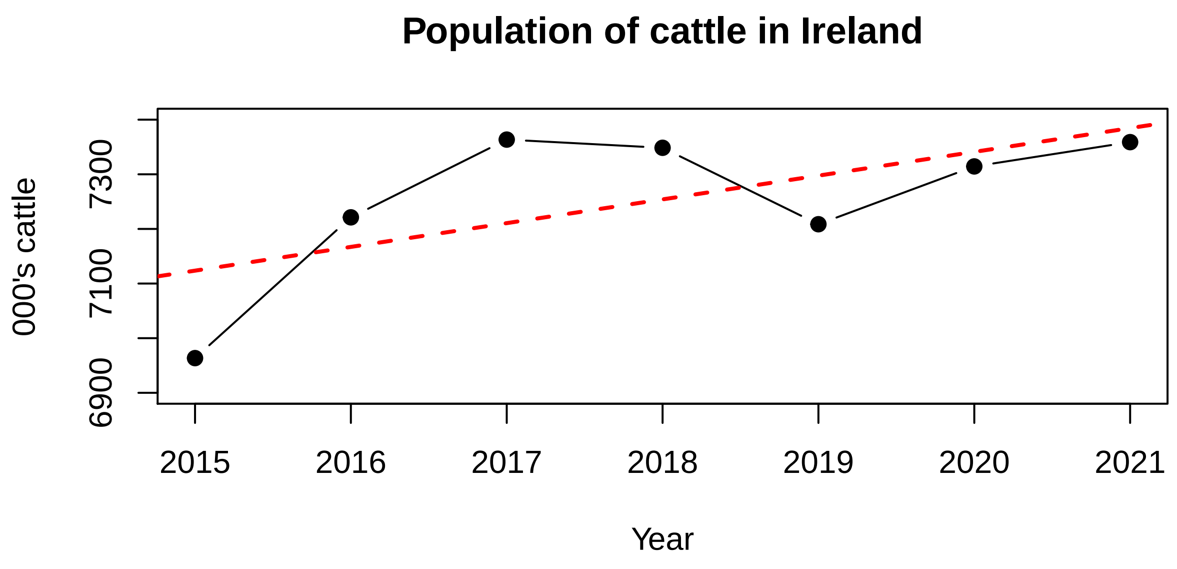
```
## [1] 0.9848778
```

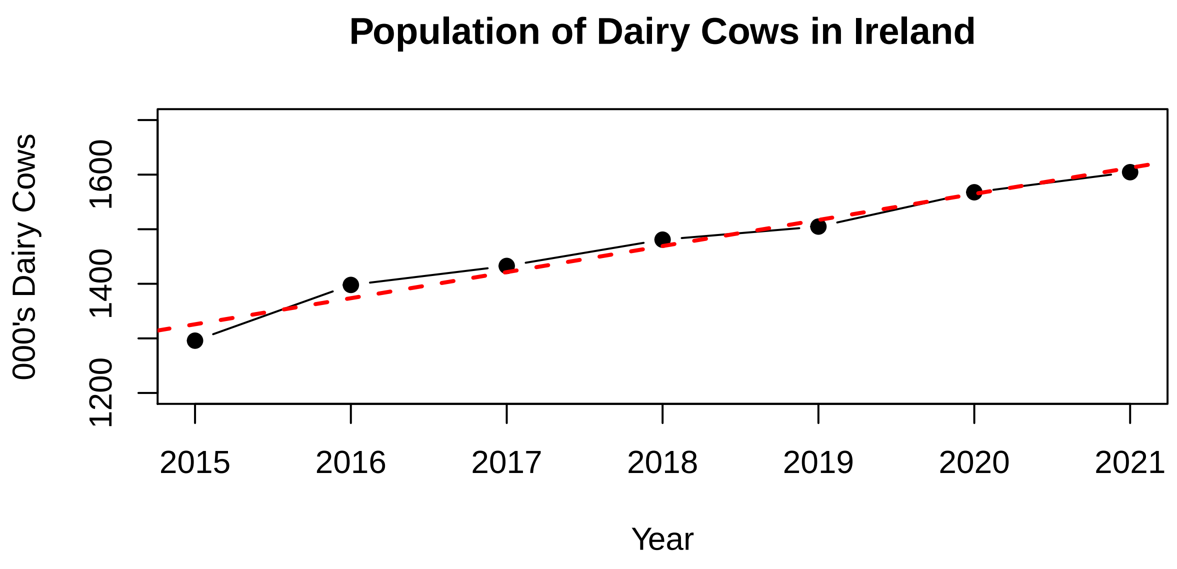
Correlation provides a measure of the strength of the linear relationship between the two variables i.e. as one variable changes in value, the other variable tends to change in a specific direction. The correlation coefficient between year and the population of dairy cows is 0.98 which indicated a strong positive relationship between the year and the population of dairy cows. The correlation coefficient between year and the total cattle population is 0.66 shows a moderate positive relationship between the total cattle population and the year.

As there is such disparity in the numbers of cattle between total and dairy cows. It is not appropriate to plot them together but to instead plot them separately. To aid with comparison I will plot both with identical range of Y-axis of 500 (i.e. 500’000 cattle).

```
# plot layout
par(mfrow = c(2,1))
# plot year vs total number of cattle in Ireland
plot(df_state$Year, df_state$Total_Cattle, type = "b",  pch = 19, main="Population of cattle in Ireland",
     xlab = "Year", ylab = "000's cattle", ylim = c(6900, 7400))
# line of the linear model of total number of cattle vs year
abline(lm(Total_Cattle ~ Year, data = df_state), col= 'red', lty= 2, lwd = 2)

# plot year vs number of dairy cows in Ireland
plot(df_state$Year, df_state$Dairy_Cows, type = "b",  pch = 19,main="Population of Dairy Cows in Ireland",
     xlab = "Year", ylab = "000's Dairy Cows", ylim = c(1200, 1700))
# line of the linear model number of dairy cows vs year
abline(lm(Dairy_Cows ~ Year, data = df_state), col= 'red', lty= 2, lwd = 2)
```





The plots strongly echo the numeric results. After an initial jump between 2015 and 2016, the population of cattle in Ireland level off and oscillating between 7.2 million and 7.4 million. There is a constant linear increase in the number of dairy cows each year. On both plots I have included a line representing the linear model between the two variables. For the number of dairy cows, the line is very close to the true data but is is not as good of a fit for the total population of cattle.

National trends in dairy cows as percentage of total cattle population in Ireland 2015 - 2021

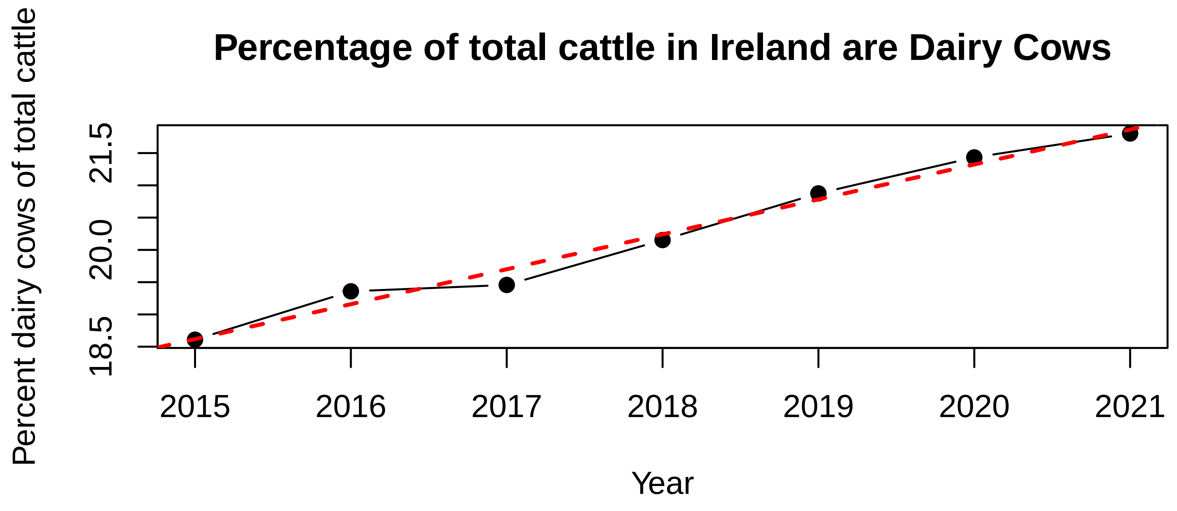
```
#summary of the dairy population figures from 2015 to 2021
summary(df_state$Percent_Dairy)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.61   19.41   20.15   20.24   21.15   21.80
```

```
# data ordered by dairy cow population from 2015 to 2021 in descending order
df_state[order(df_state$Percent_Dairy, decreasing = TRUE),c('Year', 'Percent_Dairy', 'diff_P')]
```

```
##   Year Percent_Dairy     diff_P
## 7 2021      21.80353 0.37046687
## 6 2020      21.43306 0.55799765
## 5 2019      20.87507 0.72265384
## 4 2018      20.15241 0.69563199
## 3 2017      19.45678 0.09850166
## 2 2016      19.35828 0.74982001
## 1 2015      18.60846 0.00000000
```

```
# plot year vs percentage of dairy cows of total cattle in Ireland
plot(df_state$Year, df_state$Percent_Dairy, type = "b", pch = 19, main="Percentage of total cattle in Irel
     xlab = "Year", ylab = "Percent dairy cows of total cattle")
# line of the best fit
abline(lm(Percent_Dairy ~ Year, data = df_state), col= 'red', lty= 2, lwd = 2)
```



The percentage of dairy cows of the total cattle population also increases steadily each year from 2015 to 2021. The cumulative percentage increase over this period is over 3%.

Section 2: County/Regional Trends

County/Regional trends involves analyzing two more categorical variables, county and region. For this portion of analysis, instead of base R I will use *tidyverse* and *ggplot*. Though *tidyverse* is less efficient than base R it produces neater results and similar with *ggplot*.

As there are two new variables added, I will not be looking at the variable *Year* as closely as I did for state trends. Instead I will only be looking at data of the years 2015 and 2020 and will be looking at overall movements in cattle and dairy cow population over this period.

```r
library(tidyverse)
library(dplyr)
#converting county-only dataframe into a tibble
df.tbl = as_tibble(df_county)

# creating two subsets using filter(). one subset for 2015 data and one subset for 2020 data
tbl.2015 = df.tbl  %>%
  filter(Year==2015 )
tbl.2020 = df.tbl  %>%
  filter(Year==2020 )
```

Regional trends in cattle population in Ireland 2015 - 2020

```r
#summary table of 2020 data
#grouped by region, it shows the sum of Total_Cattle i.e. the total cattle figure for each region
#ordered in descending order
TC_2020 = tbl.2020 %>%
        group_by(Region) %>%
        summarize(Total_Cattle = sum(Total_Cattle, na.rm = TRUE)) %>%
        arrange((1/Total_Cattle), desc(Region))

#same as above but for 2020 data
TC_2015 = tbl.2015 %>%
        group_by(Region) %>%
        summarize(Total_Cattle = sum(Total_Cattle, na.rm =TRUE)) %>%
        arrange((1/Total_Cattle), desc(Region))

#same as above but for Dairy_Cow figures
DC_2020 = tbl.2020 %>%
        group_by(Region) %>%
        summarize(Dairy_Cows = sum(Dairy_Cows, na.rm = TRUE)) %>%
        arrange((1/Dairy_Cows), desc(Region))
DC_2015 = tbl.2015 %>%
        group_by(Region) %>%
        summarize(Dairy_Cows = sum(Dairy_Cows, na.rm = TRUE)) %>%
        arrange((1/Dairy_Cows), desc(Region))

#same as above but for percent_dairy figures
PD_2015 <- tbl.2015 %>%
        group_by(Region) %>%
        summarize(Percent_Dairy = mean(Percent_Dairy, na.rm = TRUE)) %>%
        arrange((1/Percent_Dairy), desc(Region))
PD_2020 <- tbl.2020 %>%
        group_by(Region) %>%
        summarize(Percent_Dairy = mean(Percent_Dairy, na.rm = TRUE)) %>%
        arrange((1/Percent_Dairy), desc(Region))
```

Table 1: 2015 figures (left) and 2020 figures (right) for Total Cattle in Ireland in 000's

Region	Total_Cattle	Region	Total_Cattle
South-West	1399.2	South-West	1462.7
Mid-West	1358.4	Mid-West	1458.8
South-East	992.7	South-East	1052.5
West	930.3	West	947.4
Midlands	811.1	Midlands	868.0
Border	793.7	Border	814.3
Dublin and Mid-East	678.4	Dublin and Mid-East	710.8

Table 2: 2015 figures (left) and 2020 figures (right) for dairy cows in Ireland in 000's

Region	Dairy_Cows	Region	Dairy_Cows
South-West	437.4	South-West	494.4
Mid-West	286.7	Mid-West	341.8
South-East	219.5	South-East	291.2
Dublin and Mid-East	103.9	Dublin and Mid-East	134.7
Border	99.3	Midlands	128.5
Midlands	93.3	Border	109.6
West	55.8	West	67.2

```r
#show tables
knitr::kable(list(TC_2015, TC_2020), align = "lc",
             caption = "2015 figures (left) and 2020 figures (right) for Total Cattle in Ireland in 000's"
```

```r
knitr::kable(list(DC_2015, DC_2020), align = "lc",
             caption = "2015 figures (left) and 2020 figures (right) for dairy cows in Ireland in 000's")
```

The South West of Ireland is dominant in terms of cattle and dairy cows in both 2015 and 2020. The Mid-West and South_East ranked second and third respectively in both metrics in both years. Dublin and the Mid_East was the region with lowest number of total cattle in both years but the west had the lowest number of Dairy Cows.

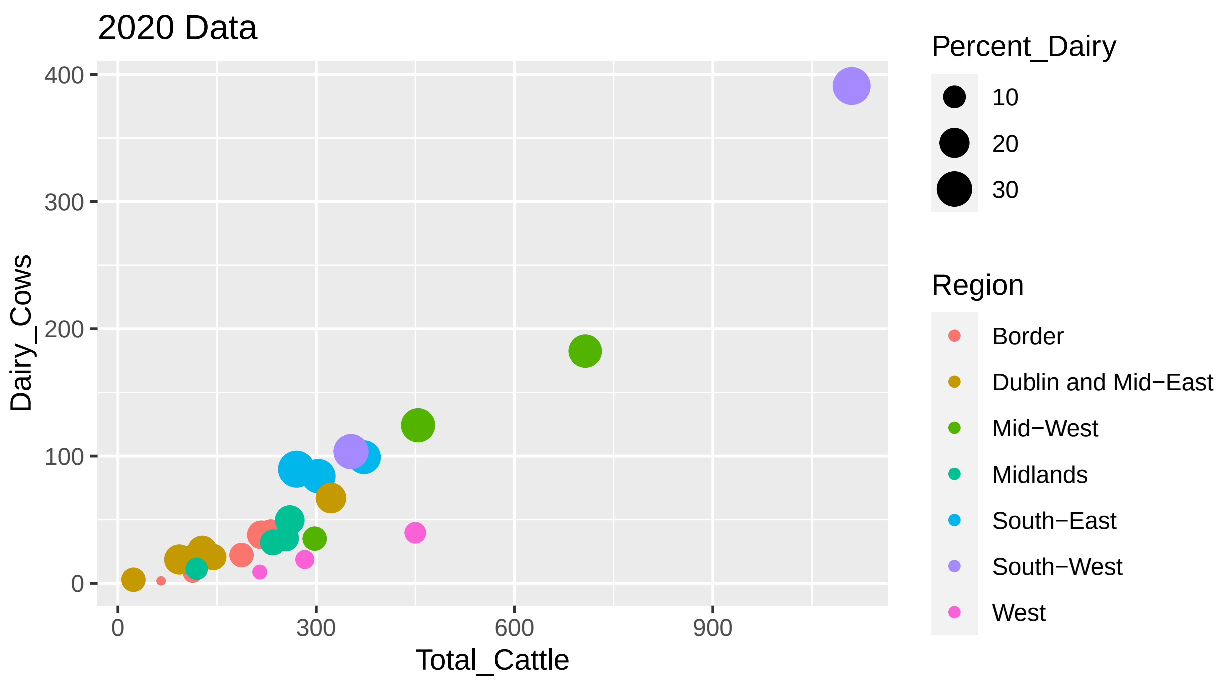
Table 3: 2015 figures (left) and 2020 figures (right) for percentage of dairy cows of total cattle population

Region	Percent_Dairy	Region	Percent_Dairy
South-West	30.420585	South-West	32.280634
South-East	20.815640	South-East	26.158603
Mid-West	19.869052	Mid-West	21.651498
Dublin and Mid-East	14.462734	Dublin and Mid-East	17.420893
Midlands	10.987866	Midlands	14.061632
Border	10.650460	Border	11.299717
West	5.557361	West	6.511791

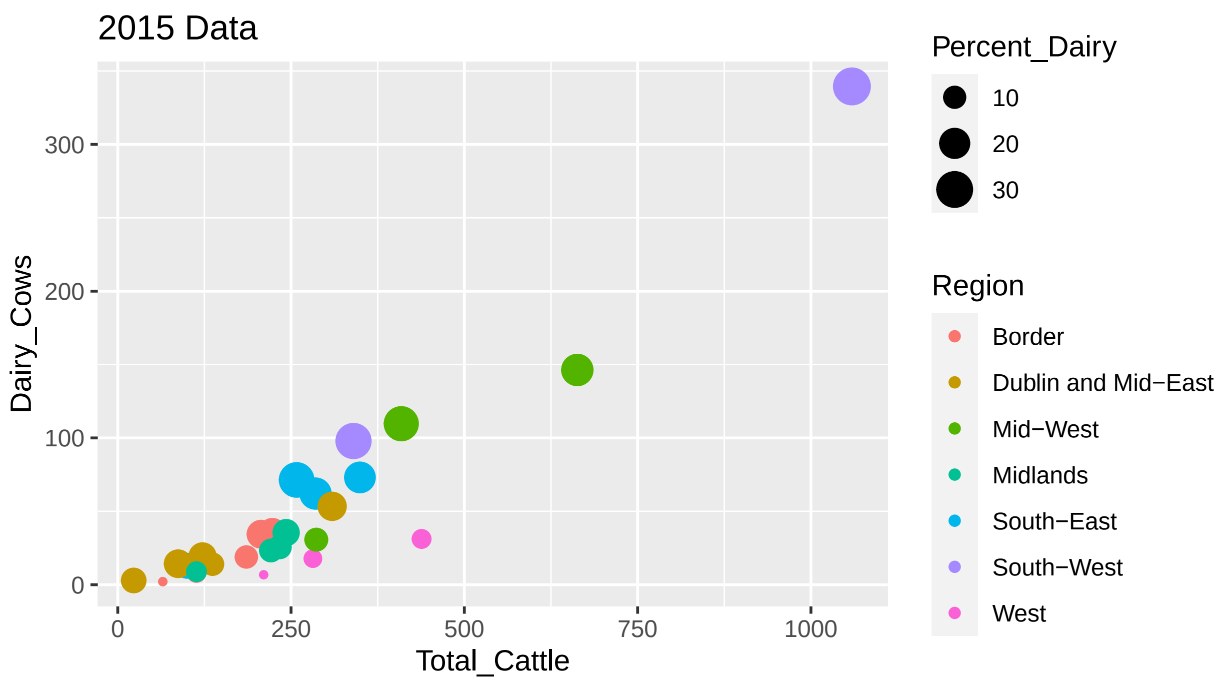
```r
knitr::kable(list(PD_2015, PD_2020), align = "lc",
             caption = "2015 figures (left) and 2020 figures (right) for percentage of dairy cows of total
```

Percentage of dairy cows exactly mirrors the number of Dairy cows table above. The South West has the highest percentage where diary cows accounted for over 30% of the total cattle population in both years. In the West of Ireland, less than 7% of cattle were dairy cows in both in both 2015 and 2020. The region with the highest growth of percentage of dairy cows was the South East, recording a 6% growth. All regions recorded a growth over the period but the Border region had the smallest growth, growing by less than 1%.

```r
# scatterplot of 2020 data, number of dairy cows vs total number of cattle
# coloured by region and size of point as percentage of dairy cows of total cattle
ggplot(tbl.2020, aes(y = Dairy_Cows, x = Total_Cattle, color = Region, size = Percent_Dairy))+
  geom_point()  +
  ggtitle("2020 Data")
```



```r
# scatterplot of 2015 data, number of dairy cows vs total number of cattle
# coloured by region and size of point as percentage of dairy cows of total cattle
ggplot(tbl.2015, aes(y = Dairy_Cows, x = Total_Cattle, color = Region, size = Percent_Dairy))+
  geom_point() +
  ggtitle("2015 Data")
```



The overall shape of the plots looks similar for 2015 and 2020 data however the scales on both axes have increased. From a maximum of 350('000) in 2015 too 400('000) in 2020 for the number of dairy cows and from 1'000('000) to 1'050('000) over the same period for total cattle.
One South-West county has by-far the most cattle and dairy cows of all the counties in Ireland. The second and third most popular counties for dairy cows are in the Mid_west and at which point the data becomes very concentrated. The points for counties with a total cattle figure of less than 300'000 seem to be smaller i.e. have a lower percentage of dairy cows than counties with over 300'000 cattle

```
# top 3 highest values by number of dairy cows in 2020
tbl.2020 %>%
  # order by number of dairy cows
  arrange(desc(Dairy_Cows)) %>%
  # take top 3 values
  slice(1:3)
```

```
## # A tibble: 3 x 6
##    Year County    Region     Total_Cattle Dairy_Cows Percent_Dairy
##   <int> <chr>     <chr>             <dbl>      <dbl>         <dbl>
## 1  2020 Cork      South-West        1110        391.          35.2
## 2  2020 Tipperary Mid-West           707        182.          25.8
## 3  2020 Limerick  Mid-West           454.       124.          27.4
```

```
# top 3 highest values by number of dairy cows in 2015
tbl.2015 %>%
  # order by number of dairy cows
  arrange(desc(Dairy_Cows)) %>%
  # take top 3 values
  slice(1:3)
```

```
## # A tibble: 3 x 6
##    Year County    Region     Total_Cattle Dairy_Cows Percent_Dairy
##   <int> <chr>     <chr>             <dbl>      <dbl>         <dbl>
## 1  2015 Cork      South-West        1059.       340.          32.1
## 2  2015 Tipperary Mid-West           663        146.          22.1
## 3  2015 Limerick  Mid-West           409        110.          26.8
```

Cork, Tipperary and Limerick are the three counties identified in the plots above and had the three highest populations of dairy cows in both 2015 and 2021.

Section 3: Linear Model

The final section of my analysis is to fit a linear model to predict the number of dairy cattle using the year and region as the predictor variables. Year will be used as a numeric variable in order to predict subsequent years and region will be a categorical variable, taking the value of one of the seven regions in the data. The model will be fitted over the years when regional data is available - 2015 to 2020.

I will use this model to then predict the dairy cow numbers for each region for 2021. As the national figures for dairy cows is available in the data I can test how well the model predicts the data. If the model is a good fit for 2021 data I will then make further predictions for the 2022 population of dairy cows in Ireland

```
# create a new table containing the sum of dairy cows, split by region and year
tbl.fit = df.tbl %>%
  group_by(Region, Year) %>%
  summarize(DairyCows = sum(Dairy_Cows)) %>%
  arrange(Region, desc(Region))
```

```
## `summarise()` has grouped output by 'Region'. You can override using the `.groups` argument.
```

```
# fit linear regression model that models number of dairy cows using region and year as the predictor vari
lm.fit <- lm(DairyCows ~ Year + Region, data = tbl.fit)

# create test data i.e. 2021 figures
tbl.test = as_tibble(tbl.fit)
tbl.test = tbl.test %>% add_row(
  Region = c("Border", "West", "Mid-West","South-East", "South-West", "Dublin and Mid-East", "Midlands"),
  Year = 2021, DairyCows = 0)
tbl.test = tbl.test %>%
  filter(Year==2021 ) %>%
  arrange(Region, desc(Region))

#predict the value of dairy cow population for 2021 using the test data on the linear model using a 95% co
tbl.predict2021 = predict(lm.fit, newdata = tbl.test, interval =  "prediction" ,level = 0.95)
# predicted vlaues for each region anf the upper and lower bound of the prediction interval
tbl.predict2021
```

```
##         fit       lwr      upr
## 1 129.50000 110.21852 148.7815
## 2 145.83333 126.55185 165.1148
## 3 341.96667 322.68519 361.2481
## 4 136.18333 116.90185 155.4648
## 5 283.58333 264.30185 302.8648
## 6 495.66667 476.38519 514.9481
## 7  86.36667  67.08519 105.6481
```

```
# sum regions to generate national prediction for dairy cow population
sum(tbl.predict2021[,1])
```

```
## [1] 1619.1
```

```
# actual 2021 figure for dairy cow population in 2021
df_state[df_state$Year == 2021, "Dairy_Cows"]
```

```
## [1] 1604.5
```

The prediction for 2021 was impressively accurate, it predicted an extra than 14'900 cattle but that is less than 1% of the actual figure.
Lets use this model to predict what the 2022 figure for number of dairy cows in Ireland will be.

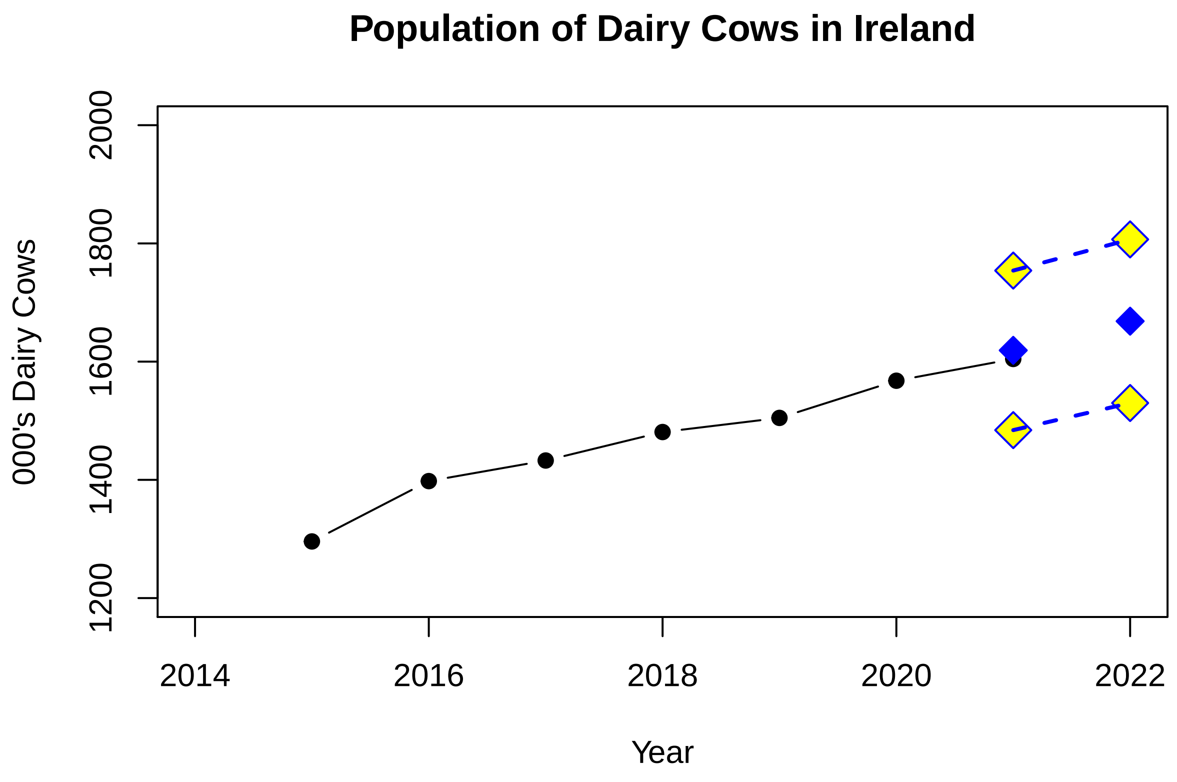
```
# new test data for 2022
tbl.test = tbl.test %>%
 add_row( Region = c("Border", "West", "Mid-West","South-East", "South-West", "Dublin and Mid-East", "Midl
 Year = 2022, DairyCows = 0) %>%
 filter(Year==2022 ) %>%
 arrange(Region, desc(Region))

#predict 2022 data using the fitted model and calculate sum over all regions
tbl.predict2022 = predict(lm.fit, newdata = tbl.test, interval =  "prediction" ,level = 0.95)
sum(tbl.predict2022[,1])
```

```
## [1] 1668.4
```

The model predicts there will be 1’668’400 dairy cows in Ireland in June 2022.
I’ll plot this value, the prediction error along with the recorded values up to 2021 and see how the estimate looks.

```
plot(df_state$Year, df_state$Dairy_Cows, type = "b",  pch = 19,main="Population of Dairy Cows in Ireland",
     xlab = "Year", ylab = "000's Dairy Cows", ylim = c(1200, 2000), xlim = c(2014, 2022))
points(x = c(2021,2022) , y = c(sum(tbl.predict2021[,1]), sum(tbl.predict2022[,1])), pch = 23,col ='blue',
points(x = c(2021,2021,2022,2022) , y = c(sum(tbl.predict2021[,2]),sum(tbl.predict2021[,3]), sum(tbl.predi
lines(x = c(2021, 2022), y = c(sum(tbl.predict2021[,2]), sum(tbl.predict2022[,2])), lwd = 2, lty = 2, col=
lines(x = c(2021, 2022), y = c(sum(tbl.predict2021[,3]), sum(tbl.predict2022[,3])), lwd = 2, lty = 2, col=
```



The prediction of the model for 2022 seems very much in line with the general trend of the data. The 95% prediction interval has a very wide range of about 300’000 cows but this is likely due to the small sample size.

Conclusion

There are very clear conclusions from the results of my analysis, the number of dairy cows in Ireland is increasing year-on-year both in numbers and as a proportion of the entire cattle population.
Dairy cows are most prevalent in the South of the county where more than 1 in 5 cattle are dairy cows, Cork is the county with the most dairy cows in Ireland with over 390’000 in 2021. Dairy cows are least prevalent in the West and Border Regions of Ireland where 11% and 6% of the total cattle population are dairy cows.
Using region and year alone can be used to effectively model the population of dairy cows in Ireland with the population likely to exceed 1’650’000 by June of 2022.
It’s clear Irish farmers are depending more on dairy farming since milk quotas were abolished in 2015.

Part 2: R Package

Introduction

One of the main advantages of programming with R is the ability to produce informative plots and visualizations of all data types. Plotting in R is suitable in so many different scenarios for all abilities. R combines functionality that allow the user to create simple intuitive plots with minimal input with the ability to make highly detailed plots with each feature being customizable to maximize the impact of a plot. Another benefit of using R is the collection of extra packages and libraries that can be downloaded. Many of these packages enhance the graphical environment of R such as *ggplot2* and *plotly*. While base R's plotting capabilities is impressive and can extend to creating 3-dimensional plots such as `persp()` and `contour()`, these graphs are not interactive and therefore the viewer only has one fixed view of the data when the plot is generated. This can lead to the viewer not experiencing the full depth of a 3-dimensional object, leading to the viewer missing important trends of the data being plotted.

The package I have chosen for this part of the project is `rgl`. `rgl` is described by it's helpfile as a "3D real-time rendering system". What this means is that `rgl` is used to produce interactive 3-D plots. The commands of `rgl` are modelled loosely after the commands of classic R graphics, making it an easy and intuitive transition for those with experience of base R. When an `rgl` plot is generated and outputted, it is not loaded in the usual location in R/RStudio but is in a special `RGL device` window that is loaded outside of R. The user can then intuitively use their mouse to move the plot. By clicking and dragging in a certain direction, the user is able to rotate the plot in any given direction and giving the viewer a new perspective on the plot and therefore on the data. The scroll wheel can also be used to zoom in and out of the plot. Because of these plots dynamic nature they cannot be stored in working order in Word/PDF files. They do, however, have functionality to create an image of a plot created with `rgl` capturing the exact image that is on the screen which can then be easily and accurately stored.

Creating 3-dimensional objects can have negative impacts also, or at least negligible positive impacts. It is harder to read values off the axes for 3-d plots. If the orientation of the plot is not suitable, it is very easy to incorrectly interpret values and for the plot to become obscured. There are many circumstances where a 2-dimensional would be more suitable than a 3-dimensional plot. 3-dimensional plots should only be used when there is added value from the extra information displayed. Scatter plots and surface plots are the best examples of this, creating a 3-dimensional render of a histogram, barplot or a piechart will add no value and should be avoided.

Functionality with examples

Below I will be showing the main uses of the `rgl` package including full examples.

Installation/Load

`rgl` needs to be installed and loaded into the users R environment before use. This is straightforward and is done as follows.

```r
install.packages('rgl')
library(rgl)
```

3-dimensional Scatter Plot using plot3d()

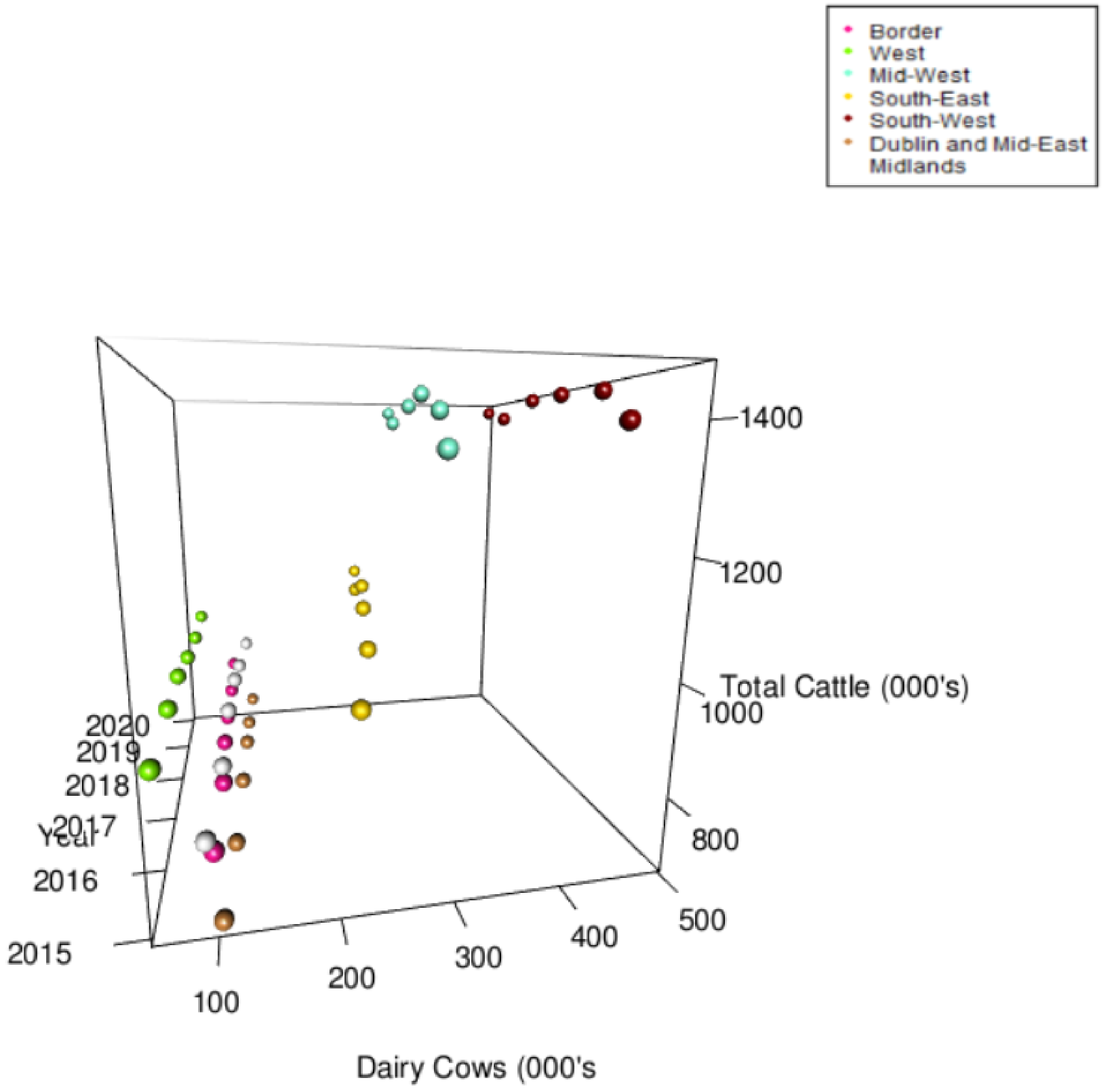
Using the county level data of from Part 1 of the project - population of dairy cows and total cattle in Ireland from 2015 to 2020 to construct a 3-dimensional plot. To add an extra information to plot, a column `Colour` is added to give a distinct colour to each region.

`rgl.open()` is used to open an rgl device. This will be blank until a plotting function is called.
`rgl.bg()` is used to set the background could which by default is grey The `plot3d()` is used to the `rgl` equivalent of `plot()` in base R. It follows similar format but an extra argument needs for be passed for the value of the z axis (third axis). Colour, type, xlab, xlim etc. are all adjustable arguments of `plot3d()`.
Using `legend3d()` a legend can be added to the rgl device to extra display on the plot. The legend is fixed in the device. `rgl.snapshot` is used to save a snapshot of the current view in the rgl device to the working directory. The filename for the image must be specified as an argument and the default format is png (Portable Network Graphics).

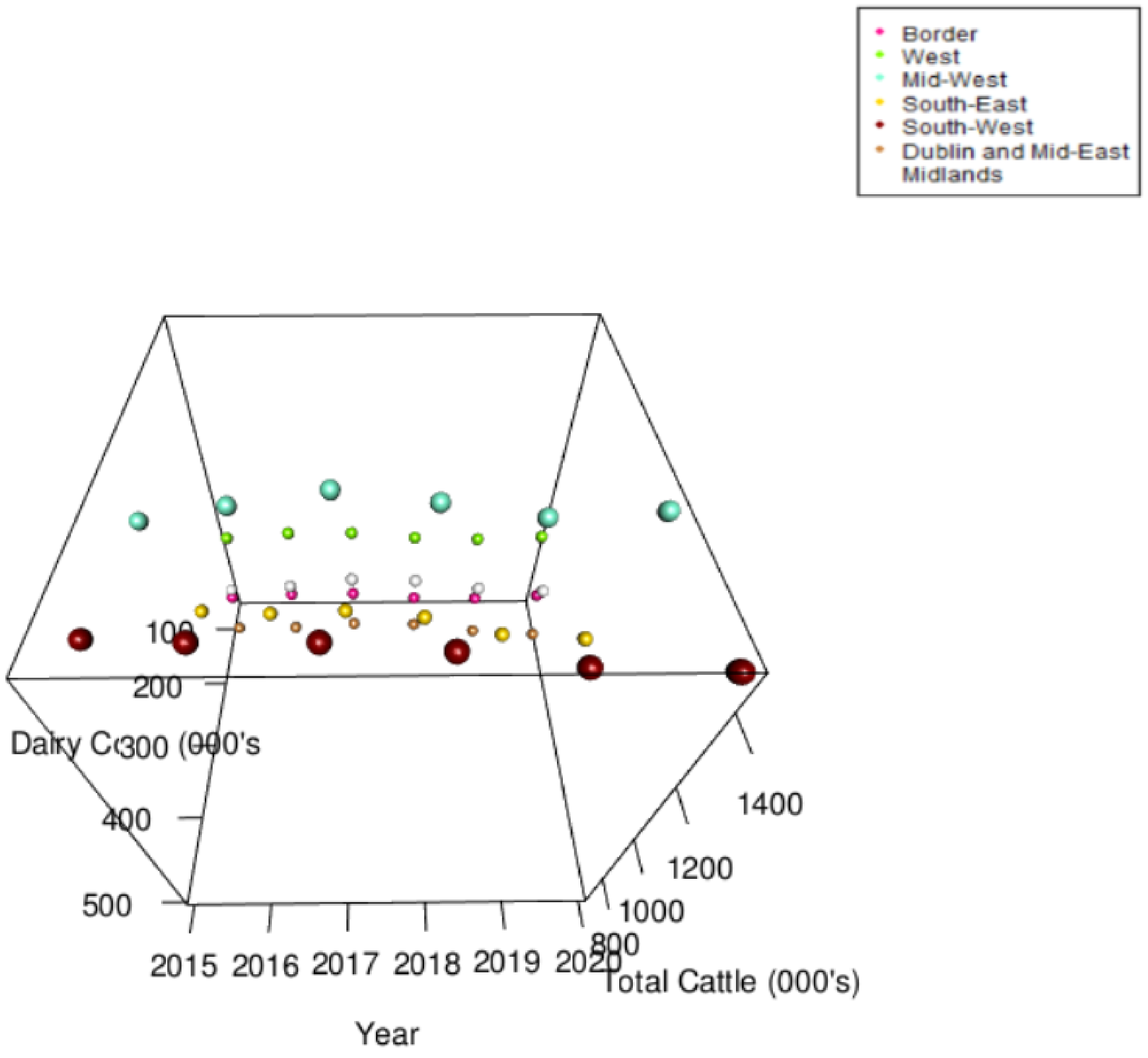
```r
# create aggregated data by region and year of population of dairy cows and total cattle
df.tbl = as_tibble(df_county)
df_regional.tbl = df.tbl %>%
  group_by(Region, Year) %>%
  summarize(Total_Cattle = sum(Total_Cattle), DairyCows = sum(Dairy_Cows))
df_regional = as.data.frame(df_regional.tbl)
# create colour column of the cattle population dataframe by region
df_regional$Colour = NA
for (i in 1:nrow(df_regional)) {
  if (df_regional$Region[i] == 'Border') {
    df_regional$Colour[i] = 'deeppink'
  } if (df_regional$Region[i] == 'West') {
    df_regional$Colour[i] = 'chartreuse'
  } if (df_regional$Region[i] == 'Mid-West') {
    df_regional$Colour[i] = 'aquamarine'
  } if (df_regional$Region[i] == 'South-East') {
    df_regional$Colour[i] = 'gold'
  } if (df_regional$Region[i] == 'South-West') {
    df_regional$Colour[i] = 'darkred'
  } if (df_regional$Region[i] == 'Dublin and Mid-East') {
    df_regional$Colour[i] = 'tan3'
  } if (df_regional$Region[i] == 'Midlands') {
    df_regional$Colour[i] = 'white'
  }
}

#open rgl device
rgl.open()
# set background colour
rgl.bg(color = 'white')
```

```r
# 3d scatterplot of year vs number of cattle vs number of dairy cows with colour as the region
plot3d(df_county$Year, df_county$Total_Cattle, df_county$DairyCows, radius = 10,type='s',
       col=df_county$Colour, xlab = "Year", ylab = "Total Cattle (000's)" , zlab = "Dairy Cows (000's")
# legend to match colour to region
legend3d("topright", legend = paste( c('Border', 'West', 'Mid-West', 'South-East', 'South-West', "Dublin a
         col = c('deeppink', 'chartreuse', 'aquamarine', 'gold','darkred', "tan3", 'white'),
         cex=1, inset=c(0.025))
#capture current view of the rgl device and save as image
rgl.snapshot(filename = "example_scatter.png")
# used to close current rgl window
rgl.close()
```



The above image is just one view of the scatterplot generated using the above code. A number of trends from the analysis from Part 1 are visible in the plot such as that the South-West and Mid-West are the regions with the highest populations of cattle and dairy cows, while the West has a relatively high number of cattle but low number of dairy cows. The change over each year is difficult to determine from this perspective and due to the scale of the axes relative to the changes.



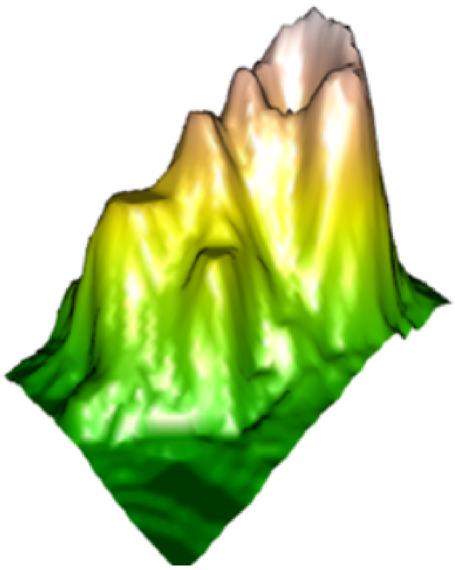
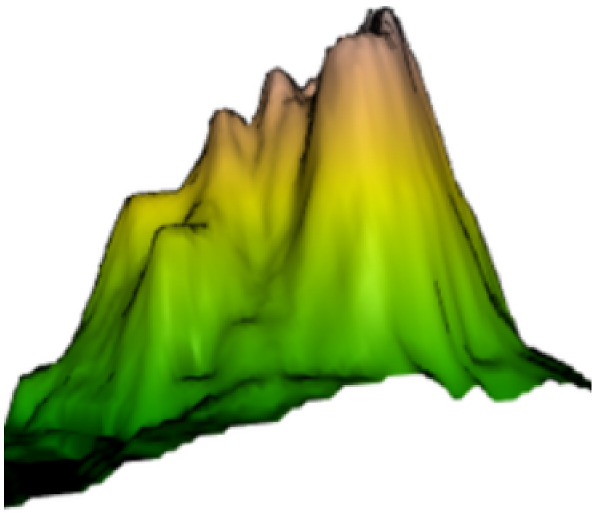
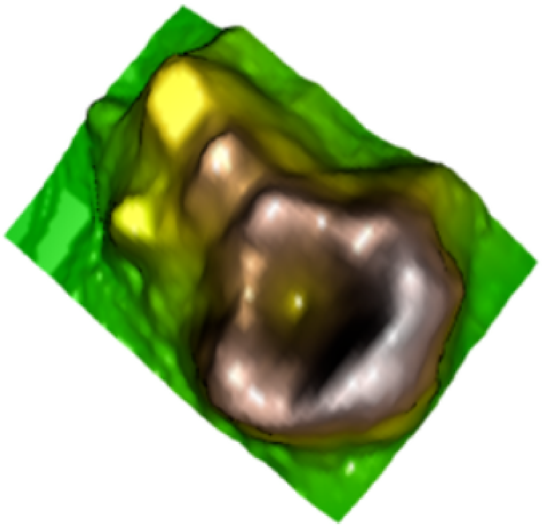
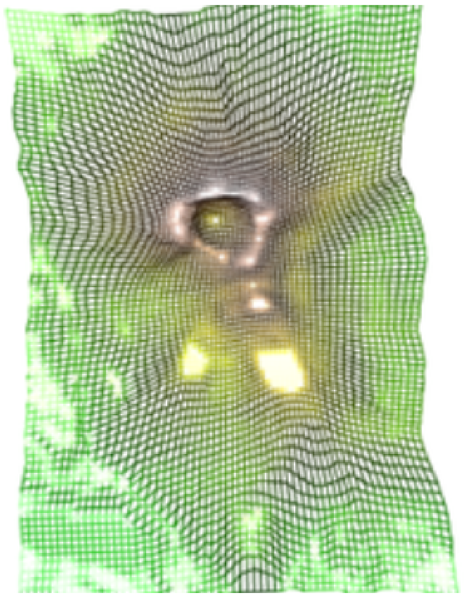
This image of the same scatterplot as above but from a different perspective and is an example of how three dimensional plots can become obscured depending on the perspective used. There is liitle that can be inferred from this view of the plot.

3-dimensional Surface Plot of a geographical feature using surface3d()

The `volcano` dataset that's included in base R is ideal for creating a surface plot.The data is a matrix containing topographical information of the Maunga Whau Volcano. It's matrix with 87 rows and 61 columns, rows corresponding to grid lines running east to west and columns to grid lines running south to north with spacing of 10m between each line.

```r
# set values of z-dimension to volcano values
# multiplied by 7.5 to exaggerate the relief
z <- 7.5 * volcano
# set x to 10m spacing over the number of rows as per gridlines
x <- 10 * (1:nrow(z))
# set x to 10m spacing over the number of columns as per gridlines
y <- 10 * (1:ncol(z))
# use range of heights of z and terrain.colours to create colour range for the gradient
zlim <- range(z)
zlen <- zlim[2] - zlim[1] + 1
```

```r
colorlut <- terrain.colors(zlen)
# assign colors to heights for each point
col <- colorlut[ z - zlim[1] + 1 ]
#open rgl device - open3d can be used as well as rgl.open
open3d()
# set background colour
bg3d("white")
# add surfec plot of x,y,z using the terrain colour scheme
surface3d(x, y, z, color = col, back = "lines")
# save current view of the rgl device as png
rgl.snapshot(filename = "example_volcano.png")
```

Above is four separate screen captures of the plot generated using the code above. It is clear the benefit `RGL` has in displaying many different perspectives of such geographical data. **Note** the image on the bottom right is of the underside of the plot

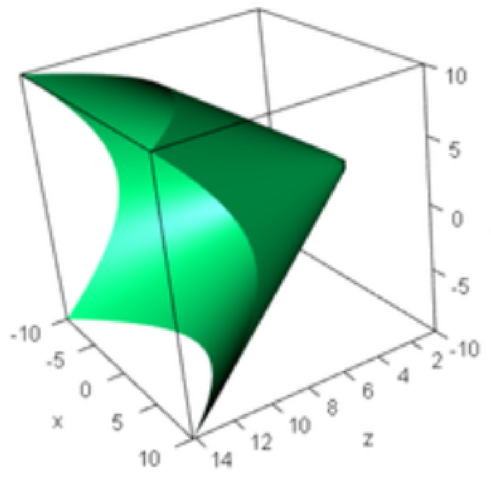
Display multivariate functions/distributions using using persp3d()

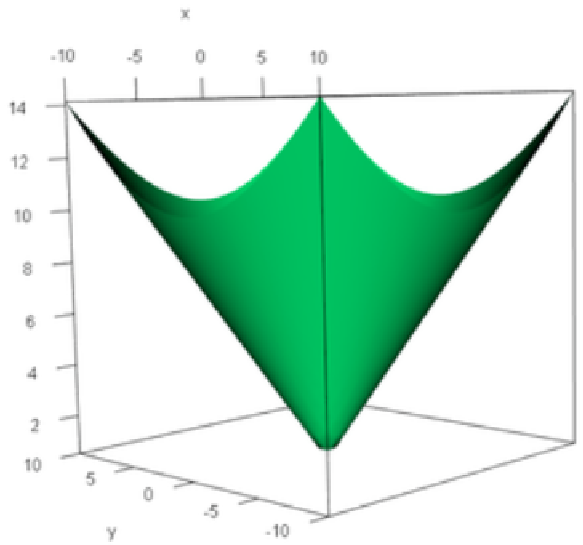
Another of `rgl` key uses is to plot multivariate functions and distributions in three dimensions. This is a very useful feature which allows complex three dimensional shapes to be visualized and aids with understanding and interpretation.

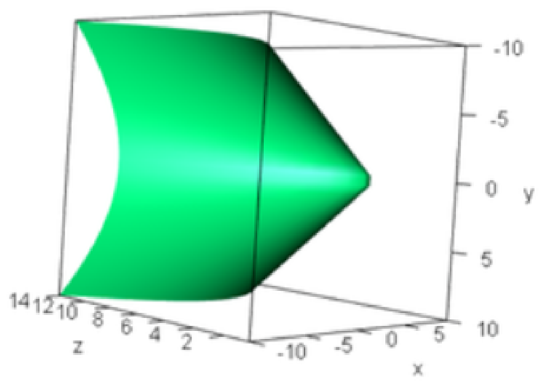
Firstly, model a simple three-dimensional cone shape which can be given as

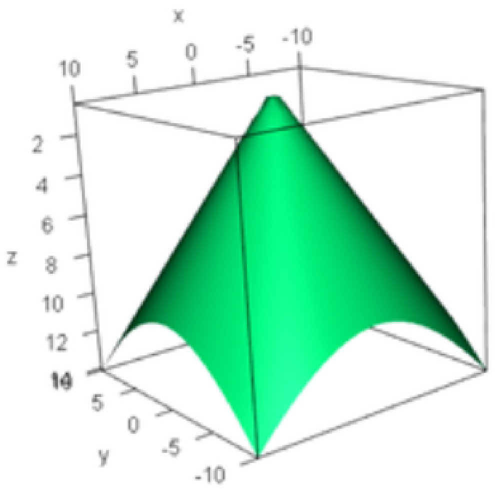
$$Z = \sqrt{X^2 + Y^2}$$

```r
#define cone function - root of x square and y squared
cone <- function(x, y){
  sqrt(x^2+y^2)
}
# define x as incremennts over the interval -1 and 1
x = y = seq(-10, 10, length= 30)
# use outer() to get values of z (cone function) using x and y
z = outer(x, y, cone)
# open rgl device
open3d()
# set background colour
bg3d("white")
# plot x,y,z, rgl also has a shade arguement to add shade to plots
persp3d(x,y,z, col = "springgreen", shade = 0.5)
# save current view of the rgl device as png
rgl.snapshot(filename = "example_cone.png")
```









Above are four screen captures of the plot generated using the code above. The relationship between X, Y and Z is clear and well visualised using `rgl`.

`rgl` is capable with more complicated functions, The sinc function of the cone is modelled below. Sinc of x is defined as the

$$\frac{\sin(x)}{x}$$

```r
# define x and y
x = y = seq(-10, 10, length = 30)
#define sinc function using previously defined cone function above
sinc <- function(x, y) {
  z = cone(x,y) ; 10 * sin(z)/z
}
# use outer() to get values of z (sinc function) using x and y
z <- outer(x, y, sinc)
# set any null/undefined values to 1
z[is.na(z)] <- 1
# open rgl device
```

```r
rgl.open()
# set background colour
rgl.bg(color = 'white')
# plot x,y and sinc(z)
persp3d(x, y, z,  col = "lightblue", xlab = "X", ylab = "Y", zlab = "Sinc(z)")
# save current view of the rgl device as png
rgl.snapshot(filename = "example_sinc.png")
```

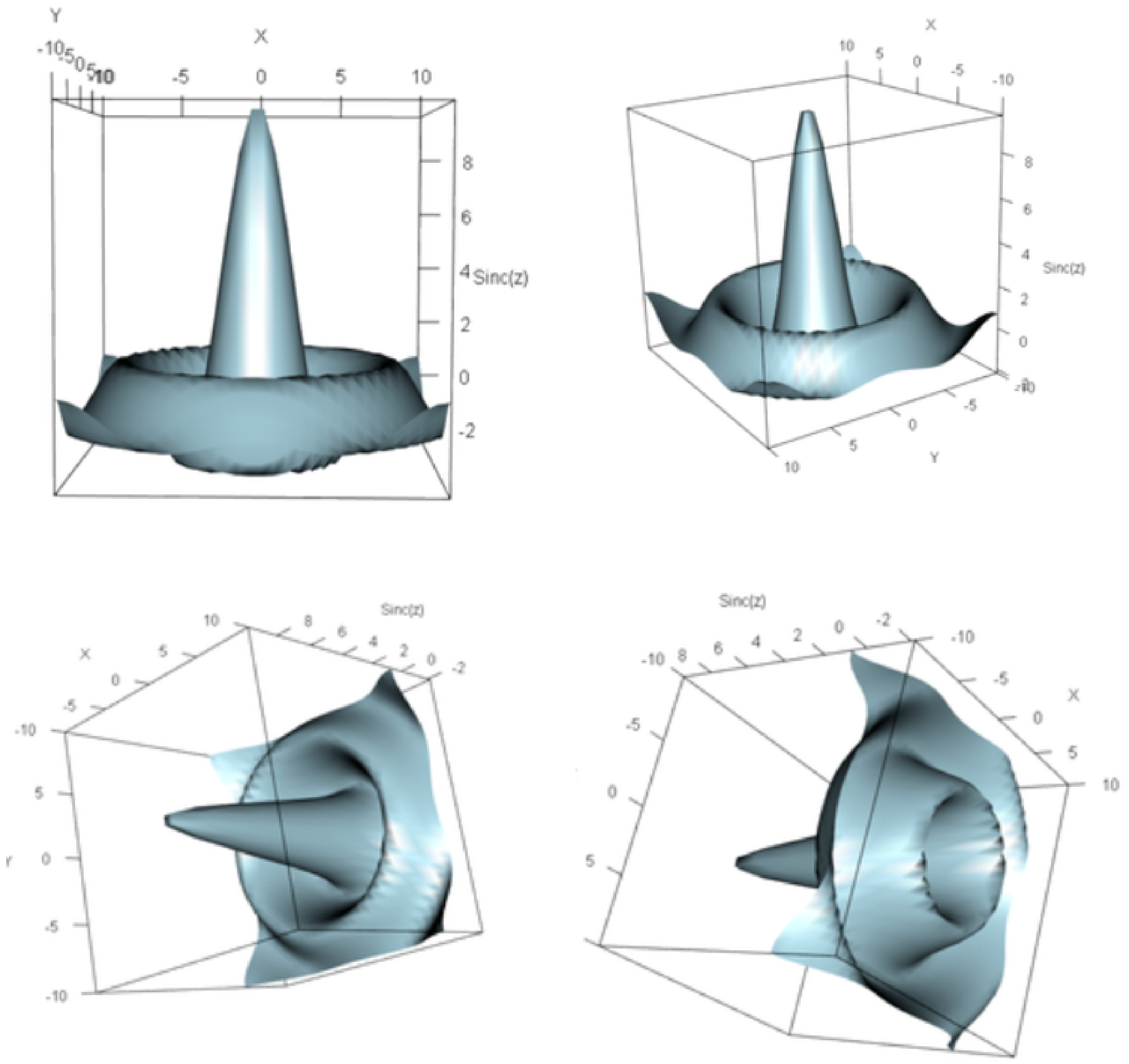


Figure 1: sample of snapshots of persp3d() of sinc function

Above are four screen captures of the plot generated using the code above. The relationship between X, Y and sinc(Z) is clear and well visualised using `rgl`.

The key difference between persp3d() and surface3d() is that persp3d() includes the axes which is needed when the user or viewer is interested in the values being displayed.

Part 3: Functions/Programming

My aim for this part is to create functions and methods based on my analysis from part 1. That is, to take in data that is divided by year and region and county for a set of numeric variable. While my analysis from part 1 was based on cattle numbers, these functions will be have wide use, and be able to provide statistical analysis to a wide array of data including demographics, emissions etc. where the data is in the correct format i.e. column 1 = year, column 2 = county, column 3 = region and the remaining columns are numeric variables of interest.

print()

The purpose of the `print()` method is to print the key information of the data. This method will print out the years included in the data, the regions in the data and the numerical variables of the class. This function only takes the data as an argument.

There is also a `region()` method included. This takes the data and a region as the arguments and returns the counties in that region.

```r
#####################################################################################
#print - (1) get info on years and regions of data
# (2) prints remaining variables in data to provide info on what is being analyzed

# one arguement of the function - the class
print.county <- function(wkr) {
  # print unique values of the region column
  cat("The regions in the dataset are:\n")
  for (i in 2:length(unique(wkr$Region))-1){
    cat(unique(wkr$Region)[i], ",\n")
  }
  cat(unique(wkr$Region)[length(unique(wkr$Region))], "\n\n")
  # print max and minimum values of the year column
  cat("The years being analysed are from",min(wkr$Year) , "to", max(wkr$Year), "\n\n")
  # print numerical variable names i.e. from column 4 onwards
  cat("The variables being analysed are \n")
  for (i in 4:length(names(df_county))){
    cat(names(df_county)[i], "\n")
  }
}

# example on given data
print.county(df_county)
```

```
## The regions in the dataset are:
## Border ,
## West ,
## Mid-West ,
## South-East ,
## South-West ,
## Dublin and Mid-East ,
## Midlands
##
## The years being analysed are from 2015 to 2020
##
## The variables being analysed are
## Total_Cattle
## Dairy_Cows
## Percent_Dairy
```

```r
# lists counties of a given region - reg
# reg passed in as string
region.county <- function(wkr, reg = NA) {
  # initialise dummy variable to store county list
  counties = NA
  cat("The counties in the", reg, "region are:")
  cat("\n")
  # select counties where region matches the argument
  for (i in 2:length(wkr$County)){
    if (wkr$Region[i] == reg) {
      counties[i] = wkr$County[i]
    }
  }
  na.omit(counties)
  #print unique counties in the list
  for (j in 2:length(unique(counties))){
    cat(unique(counties)[j], ",\n")
  }
}
# example on given data
region.county(df_county, 'Border')
```

```
## The counties in the Border region are:
## Sligo ,
## Cavan ,
## Donegal ,
## Monaghan ,
## Leitrim ,
```

The `print()` method displays the important info of the class provided and can be extended to all forms of data split by region and year.

summary()

The summary method for this class will return valuable descriptive statistics on the variables being analyzed. Optional arguments will allow the user to get aggregated means by region, year, or both.

```r
# summary method
# data argumant reguired + 3 optional arguments
summary.county <- function(wkr,  rgn = FALSE, yr = FALSE, rgn_yr = FALSE) {
  # loop over numerical variables - column four onwards
  for (i in 4:length(names(wkr))){
    # print sumary statistics for that variable
    cat("The summary statistics of", names(wkr)[i], "for the counties of Ireland is \n")
    print(summary(df_county[[i]]))
    # identify and locate miniumn and maximum values for the variable
    cat("The maximum value for", names(wkr)[i], "was recorded in" ,
        wkr[[2]][wkr[[i]] == max(wkr[[i]])], "in",
        wkr[[1]][wkr[[i]] == max(wkr[[i]])] , "\n")
    cat("The minimum value for", names(wkr)[i], "was recorded in" ,
        wkr[[2]][wkr[[i]] == min(wkr[[i]])], "in",
        wkr[[1]][wkr[[i]] == min(wkr[[i]])] , "\n")
    # if optional argument rgn is specified and set to TRUE
    # return mean by region for the variable
    if (rgn == TRUE){
      cat("The regional means for" , names(wkr)[i], "are \n")
      regional_mean = aggregate(wkr[[i]], list(wkr[[3]]), FUN=mean)
      colnames(regional_mean) = c("region", "mean")
      print(regional_mean)
    }
    # if optional argument yr is specified and set to TRUE
    # return mean by year for the variable
    if (yr == TRUE) {
      cat("\n The yearly means for" , names(wkr)[i], "are \n")
      regional_mean = aggregate(wkr[[i]], list(wkr[[1]]), FUN=mean)
      colnames(regional_mean) = c("year", "mean")
      print(regional_mean)
    }
    # if optional argument rgn_yr is specified and set to TRUE
    # return mean by year and region for the variable
    if (rgn_yr == TRUE) {
      cat("\n The yearly regional means for" , names(wkr)[i], "are \n")
      regional_year = aggregate(wkr[[i]], list(wkr[[3]], wkr[[1]]), FUN=mean)
      colnames(regional_mean) = c("region","year", "mean")
      print(regional_mean)
    }
    cat("\n\n")
  }
}

#example with given data
summary.county(df_county, yr = TRUE)
```

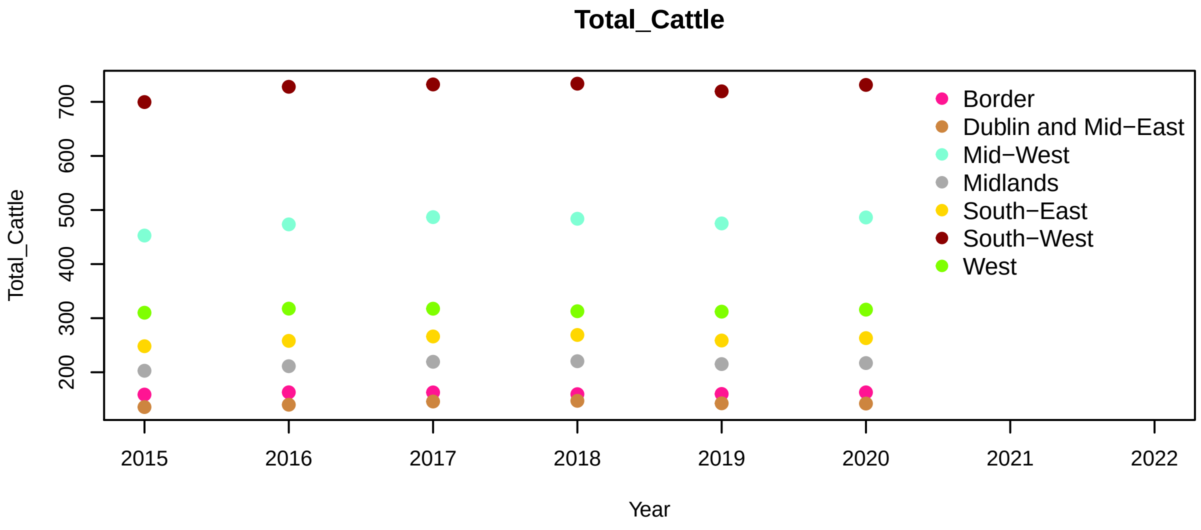
```
## The summary statistics of Total_Cattle for the counties of Ireland is
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    23.0   128.1   235.1   278.3   322.7  1125.0
## The maximum value for Total_Cattle was recorded in Cork in 2018
## The minimum value for Total_Cattle was recorded in Dublin in 2015
##
##  The yearly means for Total_Cattle are
##   year     mean
## 1 2015 267.8385
## 2 2016 277.7346
## 3 2017 283.2115
## 4 2018 282.6231
## 5 2019 277.2538
## 6 2020 281.3269
##
##
## The summary statistics of Dairy_Cows for the counties of Ireland is
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.90   16.68   32.00   55.64   71.70  390.90
## The maximum value for Dairy_Cows was recorded in Cork in 2020
## The minimum value for Dairy_Cows was recorded in Leitrim Leitrim in 2019 2020
##
##  The yearly means for Dairy_Cows are
##   year     mean
## 1 2015 49.84231
## 2 2016 53.76923
## 3 2017 55.10000
## 4 2018 56.95385
## 5 2019 57.86923
## 6 2020 60.28462
##
##
## The summary statistics of Percent_Dairy for the counties of Ireland is
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.905  10.612  15.799  16.163  22.088  35.216
## The maximum value for Percent_Dairy was recorded in Cork in 2020
## The minimum value for Percent_Dairy was recorded in Leitrim in 2020
##
##  The yearly means for Percent_Dairy are
##   year     mean
## 1 2015 14.99617
## 2 2016 15.66842
## 3 2017 15.70336
## 4 2018 16.22720
## 5 2019 16.93783
## 6 2020 17.44366
```

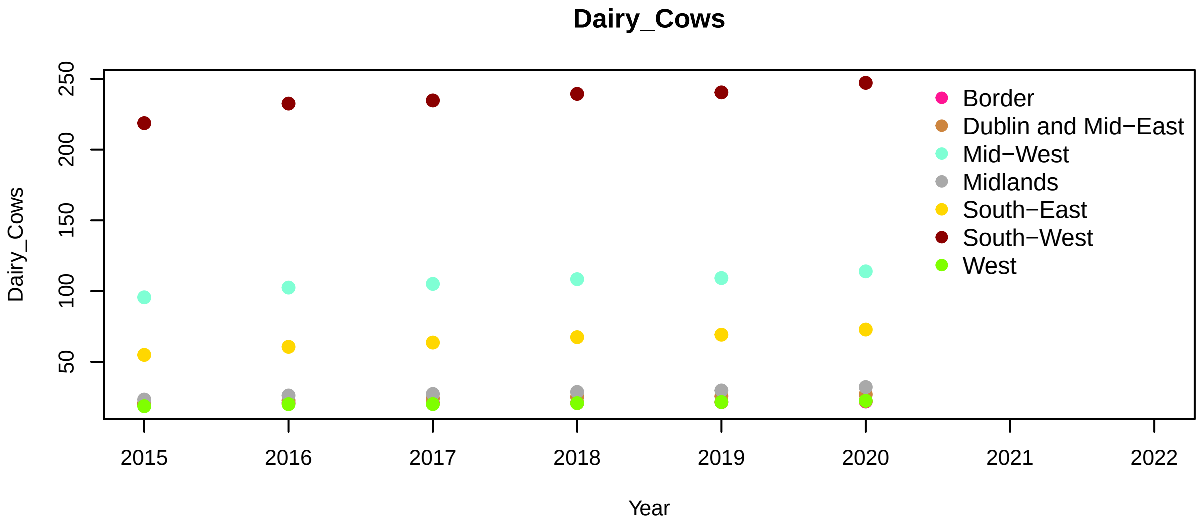
The `summary()` method above can quickly summarize and quantify the key findings from the analysis in Part 1. From the example it is shown all three variables have increased over the period 2015 to 2020. This `summary` method can also be used to perform similar analysis on any dataset that is of the same form as the one used in the example and is adaptable to the number and category of variables.

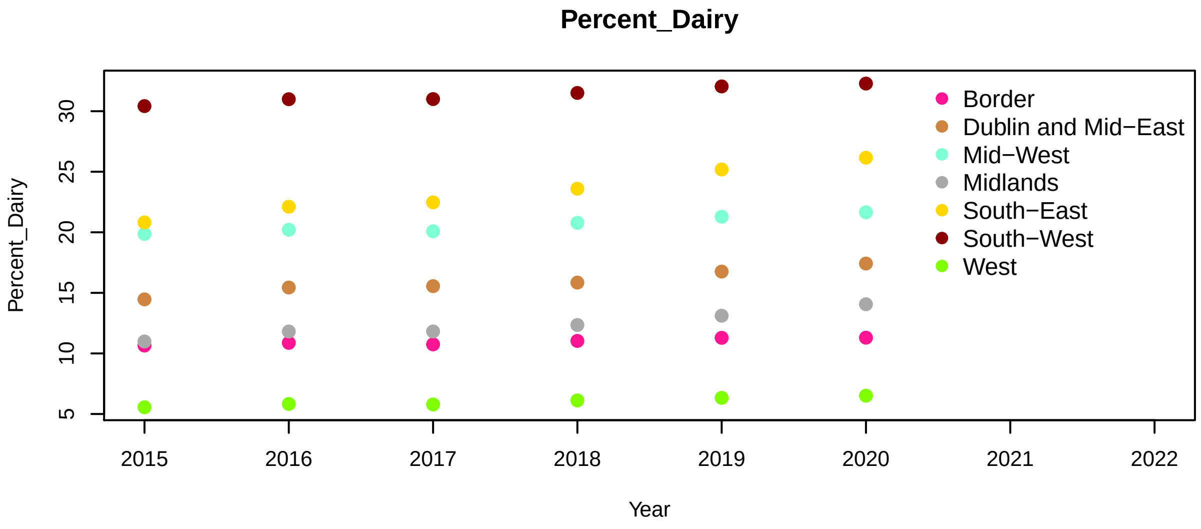
plot()

For the `plot()` method, I will create a function that will create a scatterplot for each variable on the single plot. With the average value of the variable by region and year on the Y-axis, year on the X-axis and colour coded by region. This will allow the viewer to instantly trends in the data by seeing (1) how the regions compare by variable and (2) how the variable changes over time for each region.

```r
# plot function, one argument, data name
plot.county <- function(wkr) {
  # number of numerical variables for plot layout
  n = length(names(wkr))-3
  par(mfrow = c(n, 1))
  # loop over each numerical variables
  for (i in 4:length(names(wkr))){
    # create plot data using aggregated (mean) data by region and year
    plot_data = aggregate(wkr[[i]], list(wkr[[1]], wkr[[3]]), FUN=mean)
    colnames(plot_data) = c("Year", "Region", "Variable")
    # create colour column for each region
    plot_data$Colour = NA
    for (j in 1:nrow(plot_data)) {
      if (plot_data$Region[j] == 'Border') {
        plot_data$Colour[j] = 'deeppink'
      }
      if (plot_data$Region[j] == 'West') {
        plot_data$Colour[j] = 'chartreuse'
      }
      if (plot_data$Region[j] == 'Mid-West') {
        plot_data$Colour[j] = 'aquamarine'
      }
      if (plot_data$Region[j] == 'South-East') {
        plot_data$Colour[j] = 'gold'
      }
      if (plot_data$Region[j] == 'South-West') {
        plot_data$Colour[j] = 'darkred'
      }
      if (plot_data$Region[j] == 'Dublin and Mid-East') {
        plot_data$Colour[j] = 'tan3'
      }
      if (plot_data$Region[j] == 'Midlands') {
        plot_data$Colour[j] = 'darkgrey'
      }
    }
    # plot year vs variable with regions as the colour
    plot(plot_data$Year, plot_data$Variable,
         col = plot_data$Colour, pch = 19, xlim = c(2015, 2022), cex= 1.25
         , xlab = names(plot_data)[1], ylab = names(wkr)[i], main=names(wkr)[i])
    # add legend to match region with colour
    legend("topright", legend = unique(plot_data$Region),
           col = unique(plot_data$Colour), pch = 19, bty = "n", cex = 1.1)
  }
}
# example using given data
plot.county(df_county)
```







The plot above echos my findings from Part 1 with regards to South West having the highest number of cattle and dairy cows. This can now be extended easily to any yearly regional data to perform similar analysis using these methods.

Bibliography

```r
toBibtex(citation("tidyverse"))
```

```
## @Article{,
##   title = {Welcome to the {tidyverse}},
##   author = {Hadley Wickham and Mara Averick and Jennifer Bryan and Winston Chang and Lucy D'Agostino Mc
##   year = {2019},
##   journal = {Journal of Open Source Software},
##   volume = {4},
##   number = {43},
##   pages = {1686},
##   doi = {10.21105/joss.01686},
## }
```

```r
toBibtex(citation("dplyr"))
```

```
## @Manual{,
##   title = {dplyr: A Grammar of Data Manipulation},
##   author = {Hadley Wickham and Romain François and Lionel Henry and Kirill Müller},
##   year = {2021},
##   note = {R package version 1.0.7},
##   url = {https://CRAN.R-project.org/package=dplyr},
## }
```

```r
toBibtex(citation("knitr"))
```

```
## @Manual{,
##   title = {knitr: A General-Purpose Package for Dynamic Report Generation in R},
##   author = {Yihui Xie},
##   year = {2021},
##   note = {R package version 1.34},
##   url = {https://yihui.org/knitr/},
## }
##
## @Book{,
##   title = {Dynamic Documents with {R} and knitr},
##   author = {Yihui Xie},
##   publisher = {Chapman and Hall/CRC},
##   address = {Boca Raton, Florida},
##   year = {2015},
##   edition = {2nd},
##   note = {ISBN 978-1498716963},
##   url = {https://yihui.org/knitr/},
## }
##
## @InCollection{,
##   booktitle = {Implementing Reproducible Computational Research},
##   editor = {Victoria Stodden and Friedrich Leisch and Roger D. Peng},
##   title = {knitr: A Comprehensive Tool for Reproducible Research in {R}},
##   author = {Yihui Xie},
##   publisher = {Chapman and Hall/CRC},
##   year = {2014},
##   note = {ISBN 978-1466561595},
##   url = {http://www.crcpress.com/product/isbn/9781466561595},
## }
```

```r
toBibtex(citation("rgl"))
```

```
## @Manual{,
##   title = {rgl: 3D Visualization Using OpenGL},
##   author = {Duncan Murdoch and Daniel Adler},
##   year = {2021},
##   note = {R package version 0.107.14},
##   url = {https://CRAN.R-project.org/package=rgl},
## }
```