Task 2

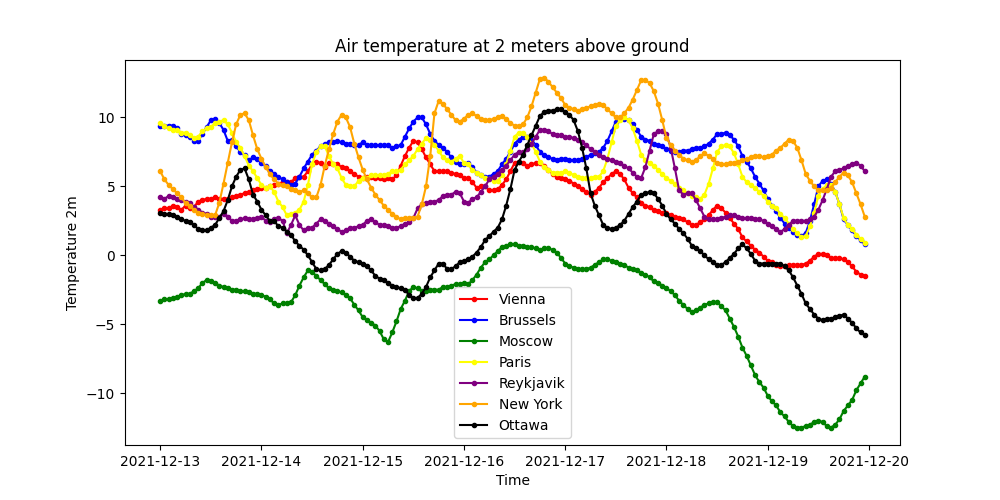
Edoardo Borriello  
Giulia Pinto  
Marta Rubino

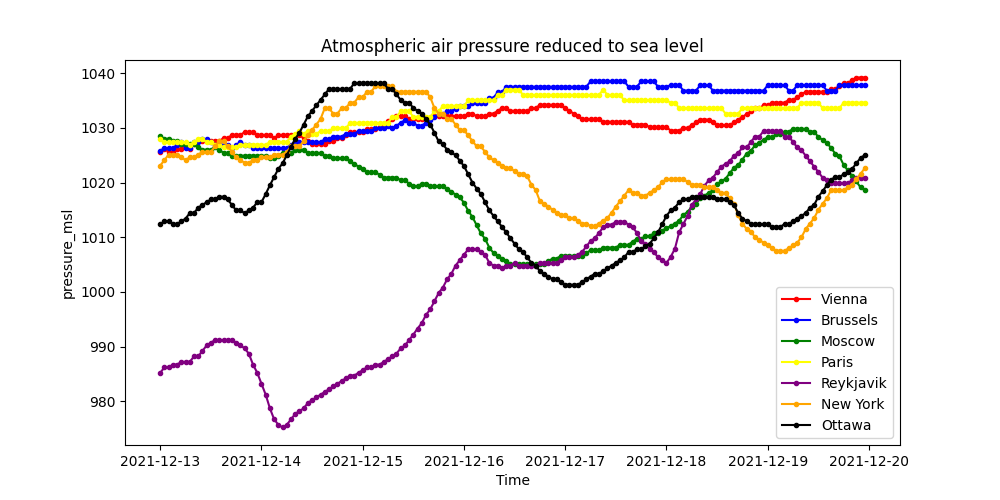
# Using Python and Dataset 1

There are seven cities under consideration in dataset 1. Size of dataset is 1176,6. First column is city, next is time and then temperature, pressure, precipitation and snow height columns.

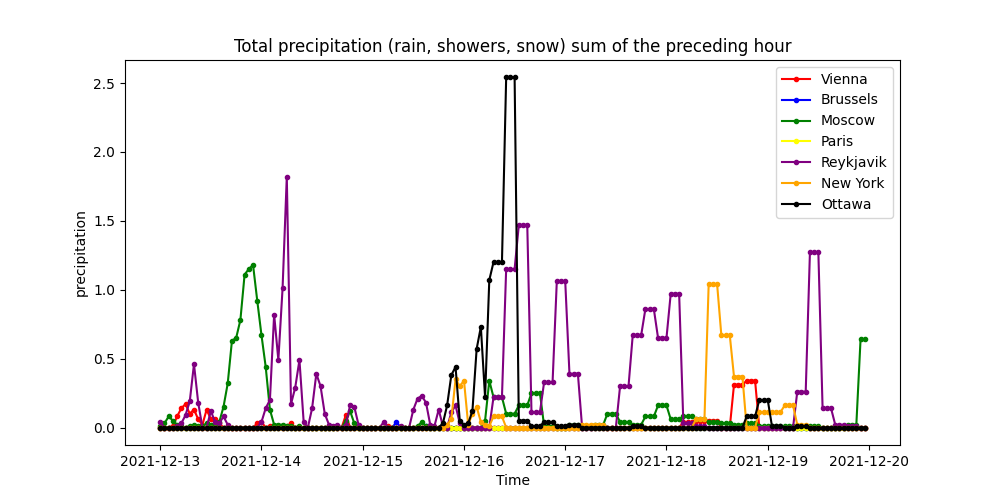
Time column was in object datatype. It is changed to datetime variable. This is done to make x axis readable.

Hourly plot using Matplotlib is done where on x axis is time and y axis has 7 curves for 7 cities. This has been done for temperature, pressure, precipitation and snow height. Legends, title, and axes labels are also added for better visualization of the dataset. A list of colors is used to show every curve of each city with different color. Datapoints of each hour are shown by a dot and a line is passing through all these dots. Code for visualization is generic. More cities or length of data can be added and it can add those too since we are using for loop to iterate through the list of cities.

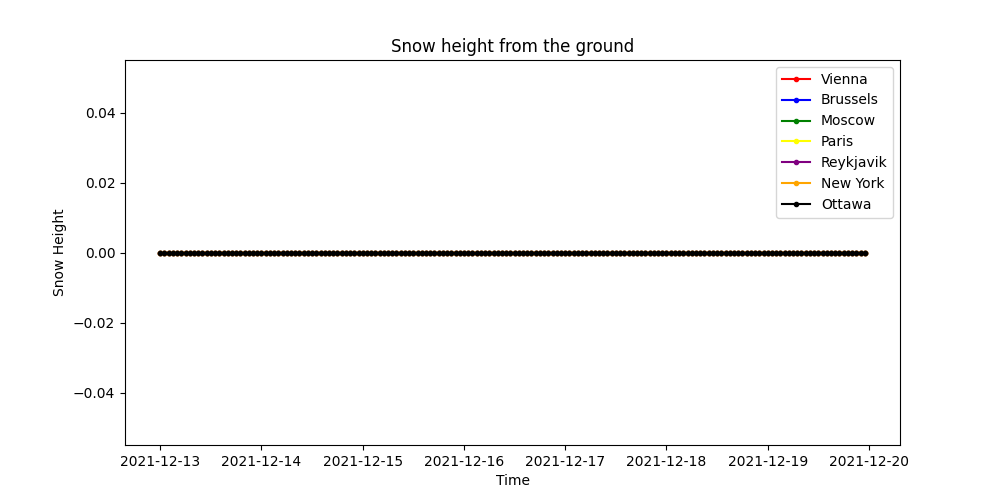


In this graph we can see that we have the minimum temperature in Moscow followed by Ottawa in which we can see a sudden climb in the middle of the week followed by an equally sudden descent. Vienna, Reykjavik and New York and Brussels and Paris start at the same level but during the week they will suffer a small but continuous drop in temperature levels. 

In this second graph instead we find that the lowest pressure among all the cities examined at the beginning of the week is in Reykjavik followed by Ottawa and subsequently by the other cities which appear to have a very similar level of pressure. We can see how the levels, apart for Vienna Brussels and Paris, vary a lot during the week while for these three cities the levels, as mentioned, are more or less stable and tend to rise.



The plot depicting the precipitation shows how all cities are affected by more or less strong phenomena. Paris and Vienna are the ones with lower levels while Reykjavik is the city where the levels have tended to be higher throughout the week. We also observe that in Ottawa there are no precipitation for most of the week apart from a single day where it reached much higher levels than in other cities.

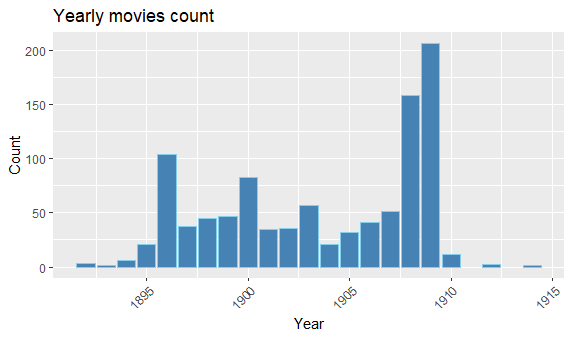


This graph shows the snow depth levels which are practically 0 for all the cities considered in our work. All data can also be found by the API from which the information is taken.

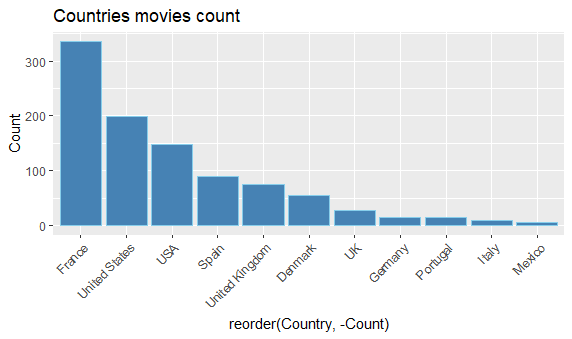
# Using R and Dataset 2

Dataset provided in this link github.com/loicvdk/python\_and\_r\_luiss\_2021/blob/main/output\_db.csv is downloaded as Dataset 2 for R data analysis.

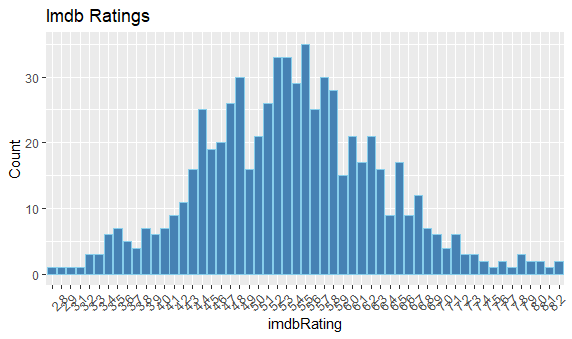
The dataset has movies information from 1892 to 1914.



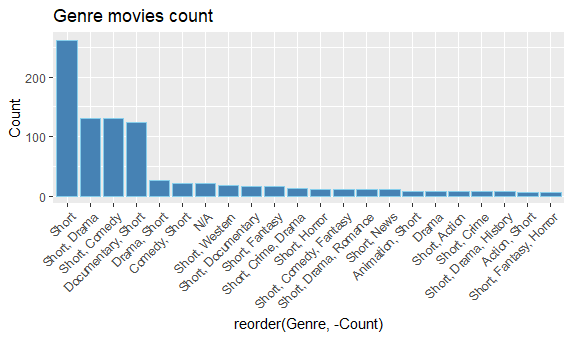
Movies released in each year are shown in this bar plot where we can see that the movie collection, apart from two peaks in 1896 1900, have more or less the same number of films until 1908/09 where we can observe much higher levels than in previous years, and then almost zeroed in the years adjacent to the beginning of the first world war.



Movies released by each country with count>5 are shown in this bar plot in decreasing order. Here we can easily see that France is the place where the majority of film production takes place, followed by the United States. Far behind all the other European states such as Germany or Italy which are very close to 0



Imdb rating of movies should show a binomial distribution. Above bar plot shows that. As can be easily understood, the majority of the votes extend into the average range of votes with a peak at 5.5. The trend tends to decrease for both the highest and lowest grades.



Movies of each genre are shown in above bar plot in decreasing order. We are taking count > 5 here as well. All films are short and the most popular genre is, apart from Shorts, Drama, practically on a par with Comedy and Documentaries. All other genres are far less popular and very close to level 0

Task 1

1 import requests

2

3 cities = ['Vienna', 'Brussels' , 'Moscow', 'Paris', 'Reykjavik', 'New York', 'Ottawa'] 4 latitude = ['48.2092', '50.8371', '55.7558', '48.8567', '64.1353', '40.71', '45.4235']

5 longitude = ['16.3728', '4.3676', '37.6176', '2.3510', '-21.8952', '-74.01', '-75.6979']

6

7

1. for i in range (0,len(cities)):
2. url = 'https://api.open-meteo.com/v1/forecast?latitude='+latitude[i]+'&longitude='+lon longitude[i]+'&hourly=temperature\_2m,pressure\_msl,precipitation,snow\_height'
3. response = requests.get(url)
4. file = open("./file-data\_"+cities[i]+".json", "w+")
5. print(file.name)
6. file.writelines(response.text)
7. file.close() 15
8. import json
9. import pandas 18 time = []

19 temperature = [] 20 pressure = []

21 precipitation = [] 22 snow\_height = []

23 l = []

1. for j in range(0,len(cities)):
2. dati\_json = json.load(open("./file-data\_"+cities[j]+".json"))
3. for h in dati\_json["hourly"]["time"]:
4. time.append(h)
5. for h in dati\_json["hourly"]["temperature\_2m"]:
6. temperature.append(h)
7. for h in dati\_json["hourly"]["pressure\_msl"]:
8. pressure.append(h)
9. for h in dati\_json["hourly"]["precipitation"]:
10. precipitation.append(h)
11. for h in dati\_json["hourly"]["snow\_height"]:
12. snow\_height.append(h)
13. for k in range(0,len(temperature)):
14. n = [cities[j], time[k], temperature[k], pressure[k], precipitation[k], snow\_heigh
15. l.append(n)
16. time = []
17. temperature = []
18. pressure = []
19. precipitation = []
20. snow\_height = [] 44

45 csv\_file\_path = 'meteo.csv' 46 df = pandas.DataFrame(l)

47 df.columns = ['city', 'time', 'temperature', 'pressure', 'precipitation', 'snow\_height']

48 df.to\_csv(csv\_file\_path, index = False)

./file-data\_Vienna.json

./file-data\_Brussels.json

./file-data\_Moscow.json

./file-data\_Paris.json

./file-data\_Reykjavik.json

./file-data\_New York.json

./file-data\_Ottawa.jso

Task 2 in Python

1. import pandas as pd
2. import matplotlib as plt
3. df.head()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **city** | **time** | **temperature** | **pressure** | **precipitation** | **snow\_height** |
| **0** Vienna | 2021-12-11T00:00 | 0.0 | 1006.1 | 0.0 | 0 |
| **1** Vienna | 2021-12-11T01:00 | 0.0 | 1007.1 | 0.0 | 0 |
| **2** Vienna | 2021-12-11T02:00 | -0.1 | 1008.2 | 0.0 | 0 |
| **3** Vienna | 2021-12-11T03:00 | -0.1 | 1008.2 | 0.0 | 0 |
| **4** Vienna | 2021-12-11T04:00 | -0.1 | 1009.2 | 0.0 | 0 |

df.shape

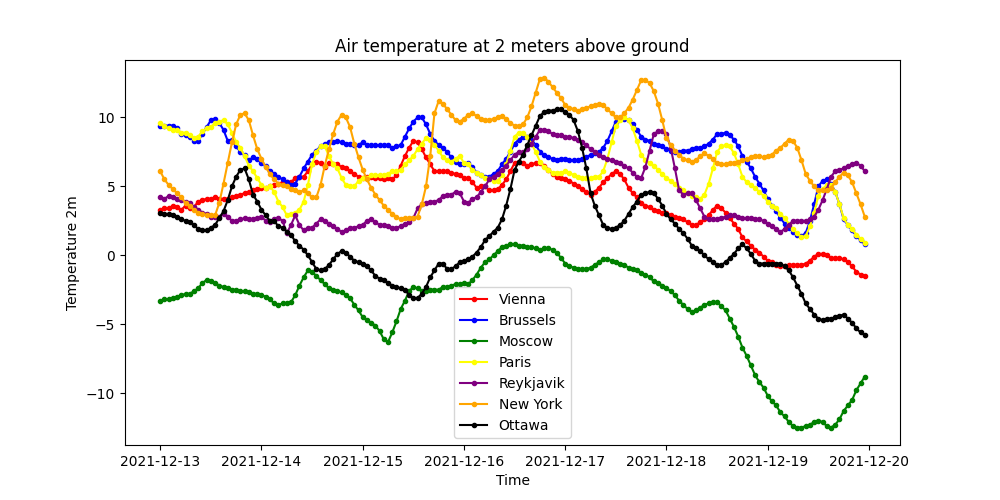
(1176, 6)

Temperature

1 df['time'] = pd.to\_datetime(df['time'])

2 i=0

1. plt.figure (figsize=(10,5))
2. c = ["red", "blue", "green", "yellow", "purple", "orange", "black"]
3. for city in cities:
4. df\_V = df[df['city']==city]
5. plt.plot(df\_V['time'],df\_V['temperature'],marker='.', color=c[i])
6. i=i+1
7. plt.legend(cities)
8. plt.xlabel("Time")
9. plt.ylabel("Temperature 2m")
10. plt.title("Air temperature at 2 meters above ground")
11. plt.show()



Pressure

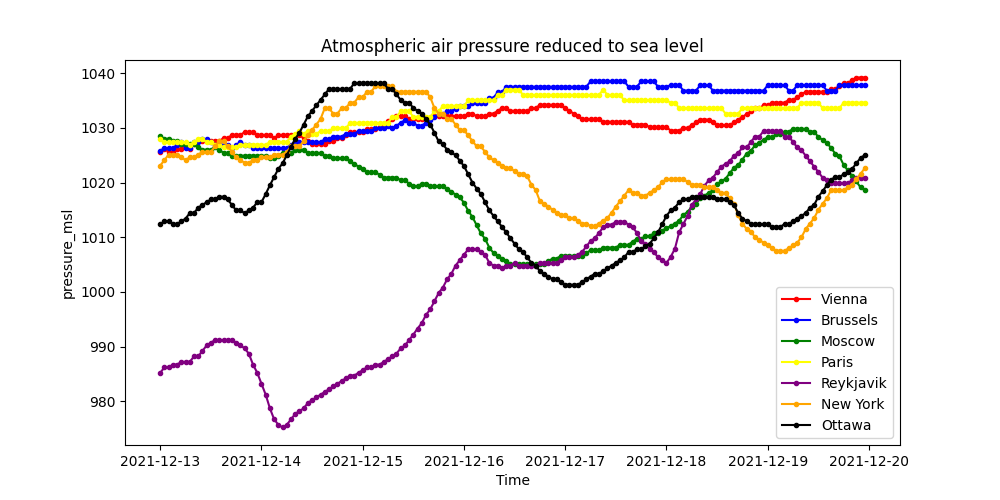
1 df['time'] = pd.to\_datetime(df['time'])

2 i=0

1. plt.figure (figsize=(10,5))
2. c = ["red", "blue", "green", "yellow", "purple", "orange", "black"]
3. for city in cities:
4. df\_V = df[df['city']==city]
5. plt.plot(df\_V['time'],df\_V['pressure'],marker='.', color=c[i])
6. i=i+1

9

1. plt.legend(cities)
2. plt.xlabel("Time")
3. plt.ylabel("pressure\_msl")
4. plt.title("Atmospheric air pressure reduced to sea level")
5. plt.show()



Precipitation

1 df['time'] = pd.to\_datetime(df['time'])

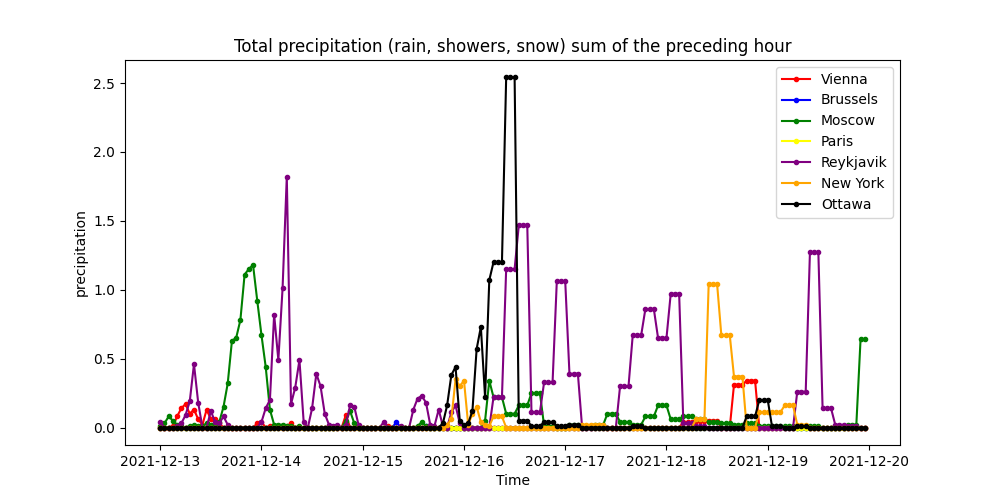
2 i=0

1. plt.figure (figsize=(10,5))
2. c = ["red", "blue", "green", "yellow", "purple", "orange", "black"]
3. for city in cities:
4. df\_V = df[df['city']==city]
5. plt.plot(df\_V['time'],df\_V['precipitation'],marker='.', color=c[i])

8 i=i+1

9

1. plt.legend(cities)
2. plt.xlabel("Time")
3. plt.ylabel("precipitation")
4. plt.title("Total precipitation (rain, showers, snow) sum of the preceding hour")
5. plt.show()



Snow Height

1 df['time'] = pd.to\_datetime(df['time'])

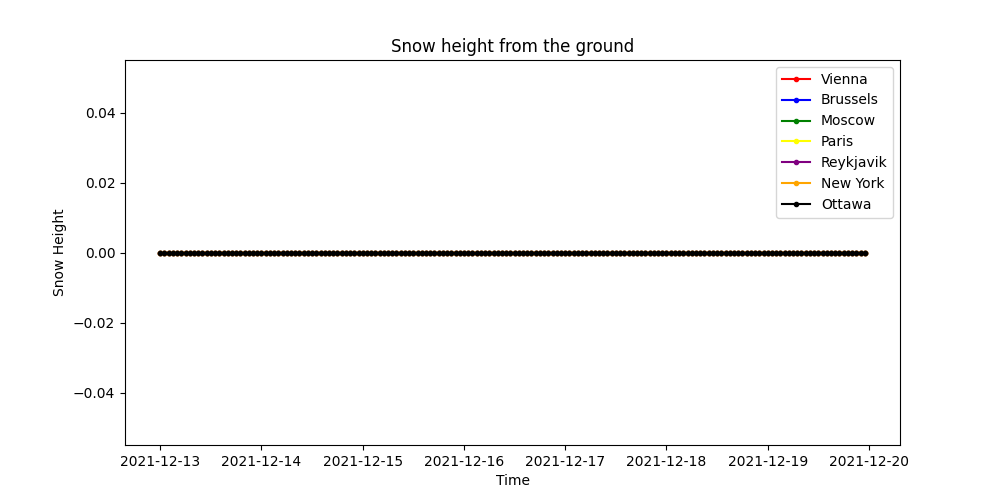
2 i=0

1. plt.figure (figsize=(10,5))
2. c = ["red", "blue", "green", "yellow", "purple", "orange", "black"]
3. for city in cities:

|  |  |  |
| --- | --- | --- |
| 6 | df\_V = df[df['city']==city] |  |
| 7 | plt.plot(df\_V['time'],df\_V['snow\_height'],marker='.', | color=c[i]) |
| 8 | i=i+1 |  |
| 9 |  |  |
| 10 | plt.legend(cities) |  |
| 11 | plt.xlabel("Time") |  |
| 12 | plt.ylabel("Snow Height") |  |

13 plt.title("Snow height from the ground")

14 plt.show()



Task 2 in R

library(dplyr)

library(ggplot2)

df <- read.csv(file = 'output\_db.csv') %>% as\_data\_frame()

df\_groupby = df %>% group\_by(Year)

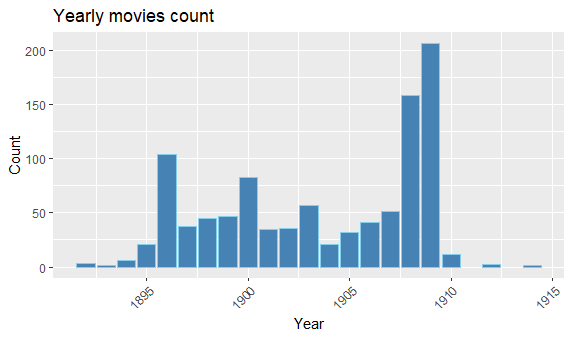
year\_mov = df\_groupby %>% summarise(Count = n()) %>%

arrange(desc(Year), desc(Count))

ggplot(year\_mov[1:21,], aes(Year, Count)) +

geom\_bar(stat="identity", color='skyblue',fill='steelblue')+

ggtitle("Yearly movies count") +

theme(axis.text.x=element\_text(angle=45, hjust=1)) 

df\_groupby = df %>% group\_by(Country)

country\_mov = df\_groupby %>% summarise(Count = n()) %>%

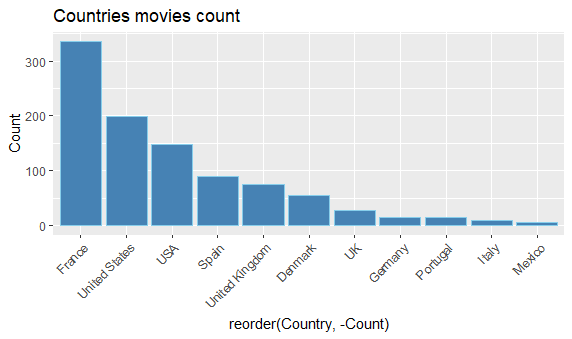
arrange(desc(Count))

ggplot(country\_mov[1:11,], aes(reorder(Country,-Count), Count)) +

geom\_bar(stat="identity", color='skyblue',fill='steelblue')+

ggtitle("Countries movies count") +

theme(axis.text.x=element\_text(angle=45, hjust=1))



df\_groupby = df %>% group\_by(imdbRating)

imdb\_mov = df\_groupby %>% summarise(Count = n()) %>%

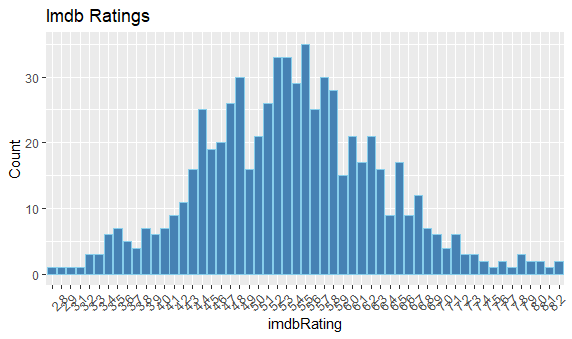
arrange(desc(Count))

ggplot(imdb\_mov[2:56,],aes(imdbRating , Count)) +

geom\_bar(stat="identity", color='skyblue',fill='steelblue')+

ggtitle("Imdb Ratings") +

theme(axis.text.x=element\_text(angle=45, hjust=1))



df\_groupby = df %>% group\_by(Genre)

genre\_mov = df\_groupby %>% summarise(Count = n()) %>%

arrange(desc(Count))

ggplot(genre\_mov[1:22,], aes(reorder(Genre,-Count), Count)) +

geom\_bar(stat="identity", color='skyblue',fill='steelblue')+

ggtitle("Genre movies count") +

theme(axis.text.x=element\_text(angle=45, hjust=1))

