Data Visualization and Manipulation

Hanna Carucci Viterbi, Olimpia Sannucci, Giulia Di Martino

13/12/2021

DATA ANALYSIS

In order to visualize data, ggplot2 was used throughout the analysis: the latter is an R package exploited for statistical computing and data representation. The main features of ggplot2 are: it improves the aesthetics of the graphics and it permits to build almost any type of chart. A BoxPlot has been adopted to compare the temperature and the apparent one of the cities under analysis. The second step was to upload the dataset.

Then, we inspected the dataset before visualizing it in order to understand what it was about

```
Forecast %>% glimpse()
```

```
## Rows: 2,352
## Columns: 9
## $ city                   <chr> "Rome", "Rome", "Rome", "Rome", "Rome", "Rome",…
## $ time                   <dttm> 2021-12-10 00:00:00, 2021-12-10 01:00:00, 2021…
## $ temperature            <dbl> 6.4, 6.4, 6.0, 5.4, 5.1, 4.7, 3.9, 3.3, 2.8, 3.…
## $ `apparent temperature` <dbl> 5.0, 4.7, 4.3, 3.8, 3.7, 3.1, 2.0, 1.3, 0.5, 1.…
## $ `relative humidity`    <dbl> 93, 91, 93, 95, 96, 98, 100, 100, 100, 98, 89, …
## $ precipitation          <dbl> 0.00, 0.07, 0.08, 0.00, 0.00, 0.00, 0.00, 0.00,…
## $ dewpoint               <dbl> 5.3, 5.1, 5.0, 4.8, 4.5, 4.4, 3.8, 3.3, 2.7, 3.…
## $ `freeing level`        <dbl> 1187, 1136, 1150, 1139, 1141, 1121, 1194, 1266,…
## $ pressure               <dbl> 1005.7, 1005.7, 1006.2, 1006.2, 1006.2, 1006.7,…
```

```
Forecast%>% distinct(Forecast$city)
```

```
## # A tibble: 14 × 1
##    `Forecast$city`
##    <chr>
##  1 Rome
##  2 Reykjavik
##  3 Jerusalem
##  4 Barcelona
##  5 Lamezia Terme
##  6 Florence
##  7 Urbania
##  8 San Martino in Pensilis
##  9 Schiavonea
## 10 Berlin
## 11 Budapest
## 12 Bruxelles
## 13 Malta
## 14 Paris
```

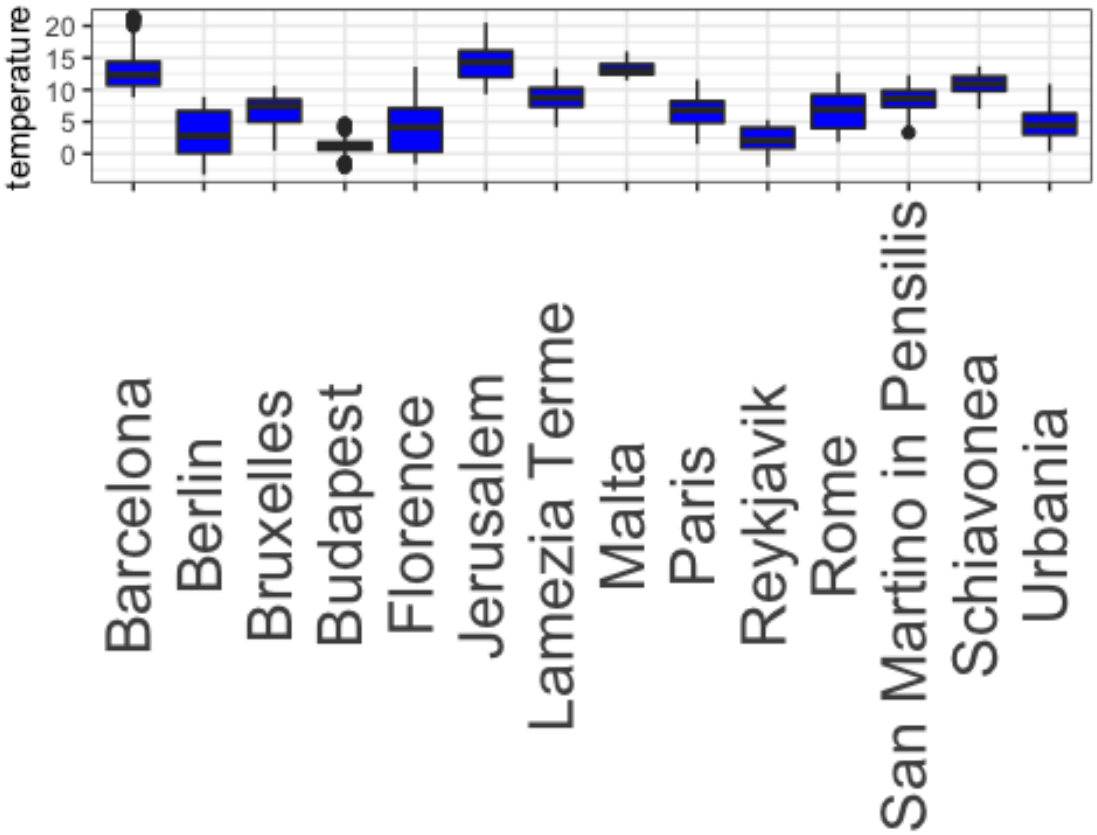
```
length(unique(Forecast$city))

## [1] 14

sum(is.na(Forecast))

## [1] 0

names(Forecast)

## [1] "city"                 "time"                 "temperature"
## [4] "apparent temperature" "relative humidity"    "precipitation"
## [7] "dewpoint"             "freeing level"        "pressure"
```

From the above code it emerged that the dataset does not have missing values. The only categorical variable is “city”, which distinguishes 14 different cities. (Barcelona,Berlin,Budapest,Bruxelles,Jerusalem, Paris, San Martino in Pensilis , Florence,Malta,Rome,Urbania,Barcelona,Budapest,Lamezia Terme, Rekjavik, Schiavonea). The numerical variables are: time, temperature, precipitation, pressure, freeing level,relative humidity, apparent temperature and dewpoint.

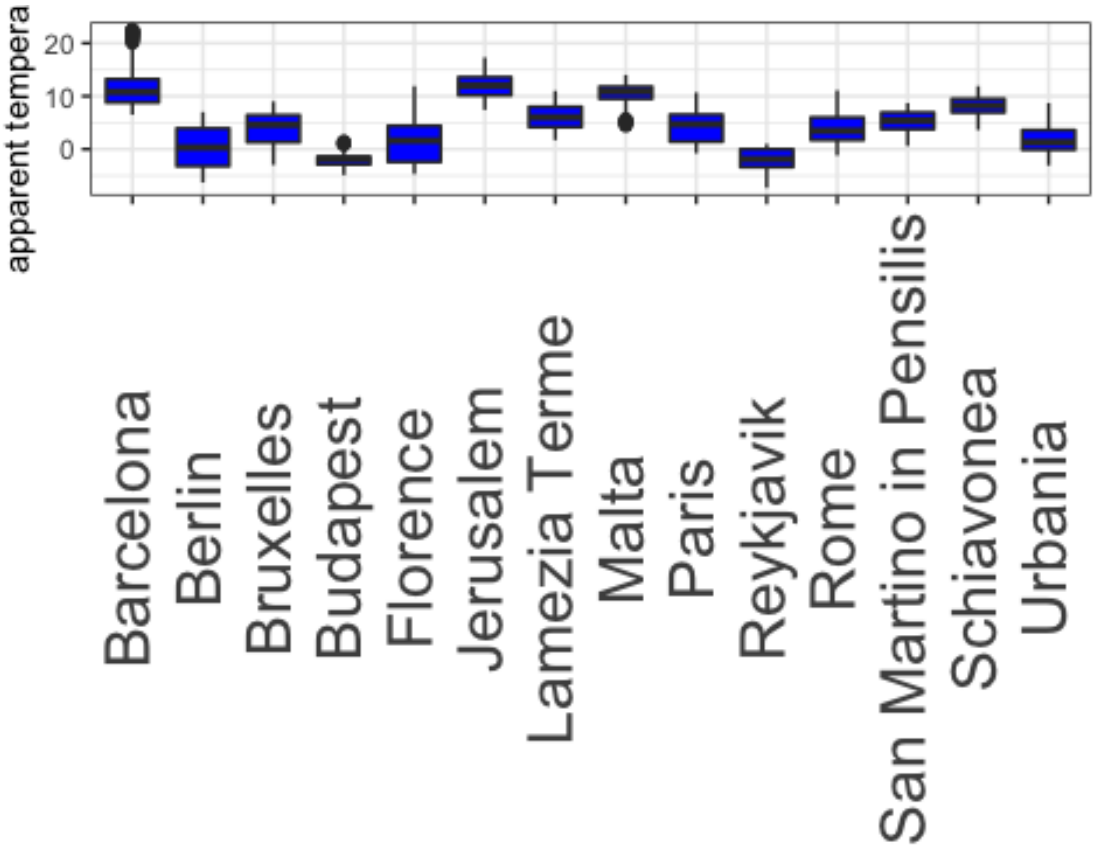
DATA VISUALIZATION

Boxplot

```
Forecast%>%
  ggplot(aes(x=city, y=temperature, fill=I("blue")))+
  geom_boxplot(position="dodge")+
  labs(x="")+
  theme_bw()+
  theme(axis.text.x=element_text(angle=90, vjust=.4, size=20))
```




```
Forecast%>%
  ggplot(aes(x=city, y=`apparent temperature`, fill=I("blue")))+
  geom_boxplot(position="dodge")+
  labs(x="")+
  theme_bw()+
  theme(axis.text.x=element_text(angle=90, vjust=.4, size=20))
```



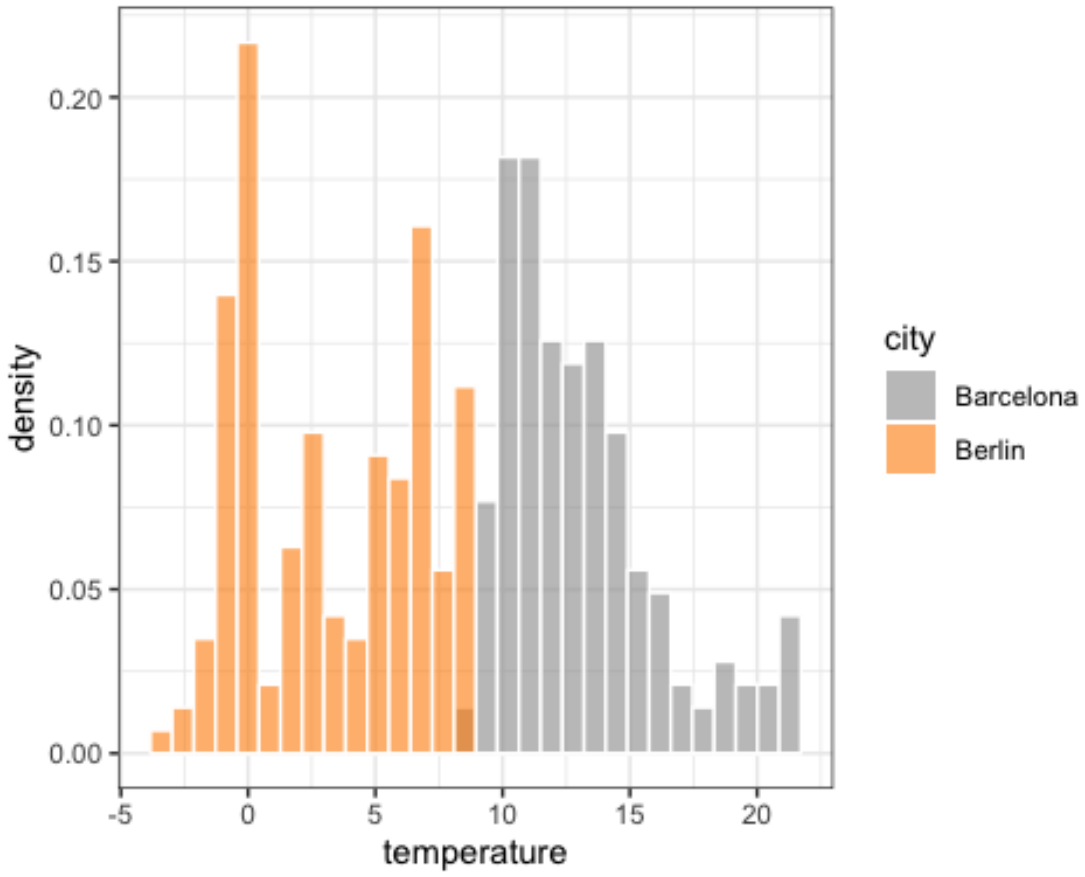
From the two Boxplots it can be observed that the temperature tends to be higher compared to the apparent temperature for each city.

Histograms

In the following histogram we can see that Barcelona has higher temperatures with respects to Berlin.

```
Forecast %>% filter(city %in% c("Barcelona", "Berlin")) %>%
  ggplot(aes(x = temperature, y = ..density.., color = I("white"), fill = city)) +
  geom_histogram(alpha = 0.6, position = "identity") +
  scale_fill_manual(values= c("grey60", "darkorange")) +
  theme_bw()
```

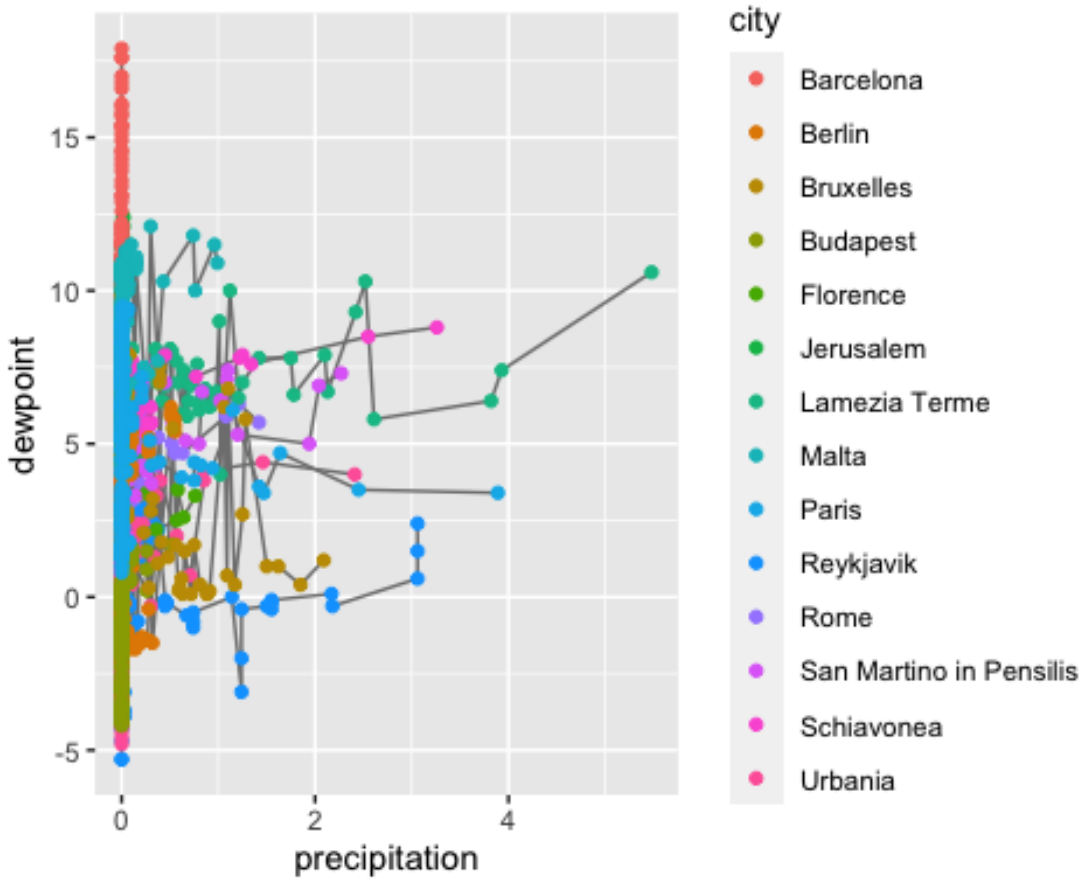
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



#Geom_line

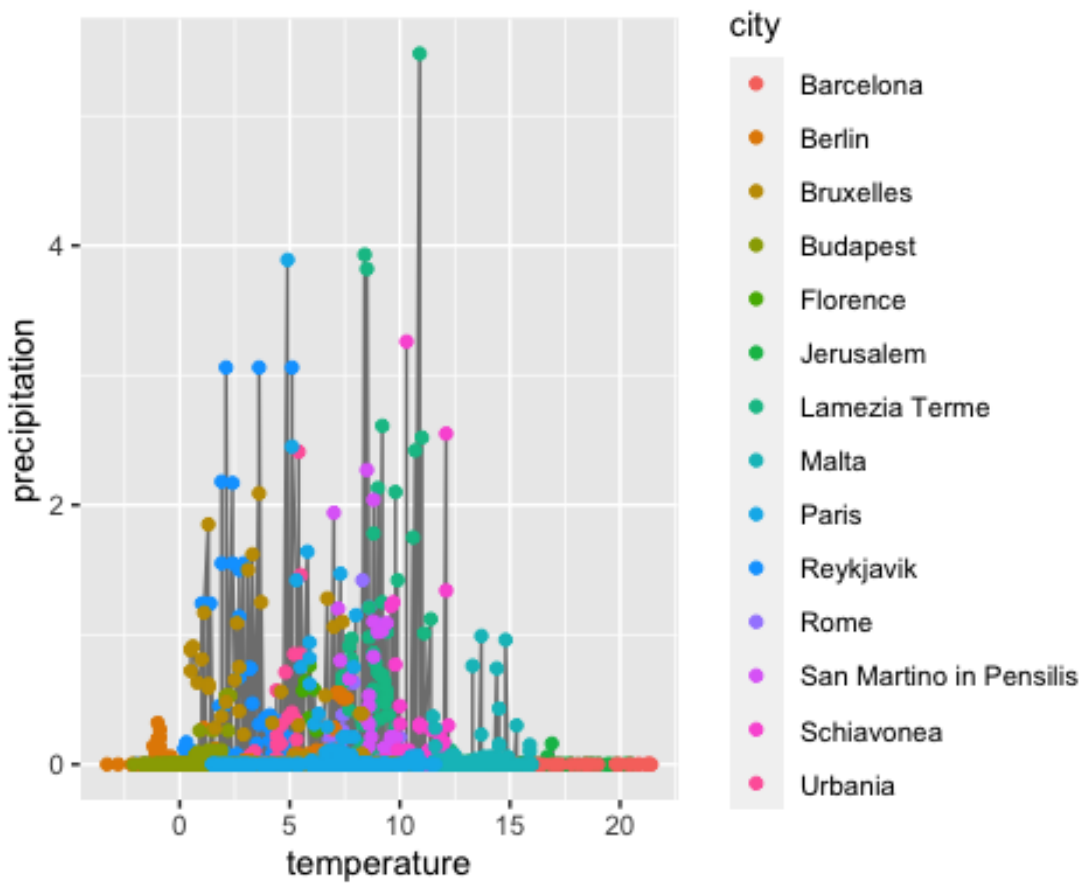
Furthermore, aesthetic mappings and geom functions (geom_line, geom_point) were used to build graphs. The geom is defined as the geometric object which displays the data. The relation between precipitation and dewpoint in each city was assessed in the following graph: it can be deduced that the city with highest precipitation is Lamezia Terme. The city with both lower precipitation and dewpoint is Reykjavik. Finally, the city with the highest dewpoint is Barcelona.

```
ggplot(Forecast, aes(precipitation, dewpoint))+
  geom_line(aes(group= city), colour="grey50")+
  geom_point(aes(colour=city))
```



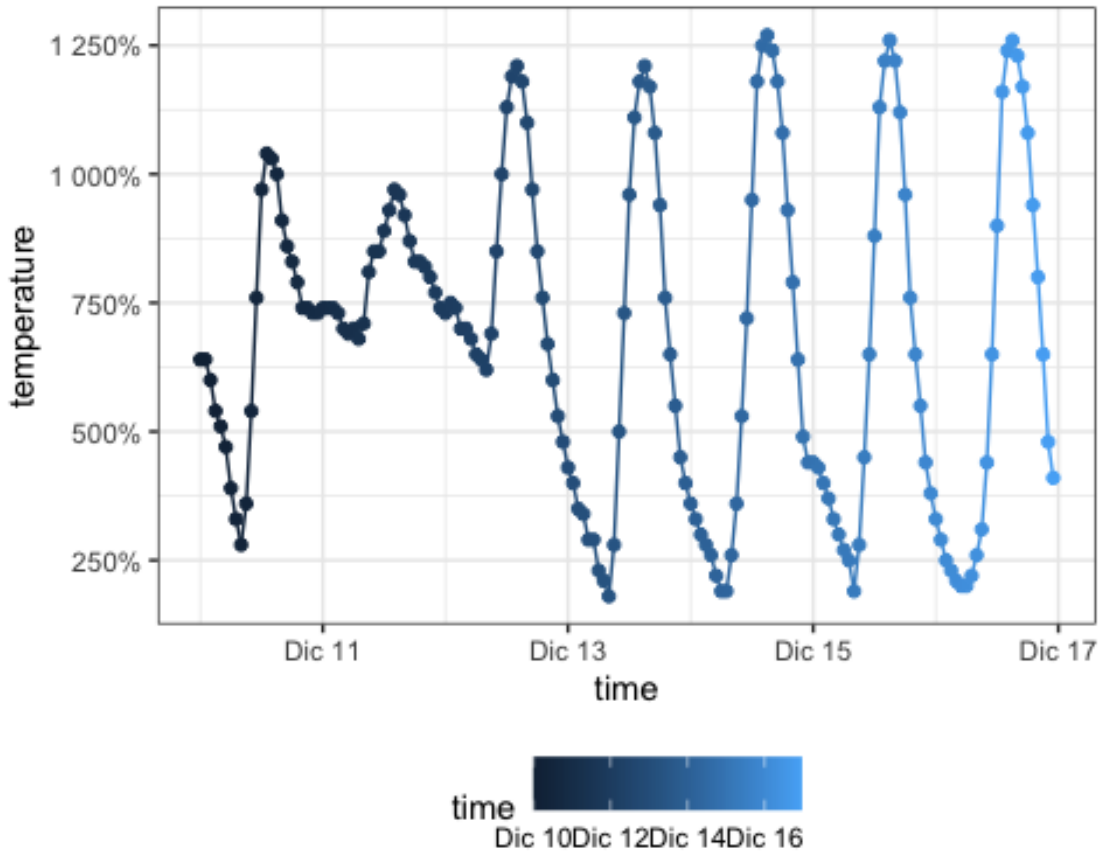
Then, the temperature and precipitation were compared: the city with highest precipitation resulted to be Lamezia Terme, followed by Paris and Reykjavik. The city with both lower temperature and dewpoint is Berlin. Barcelona is the city with the highest temperature.

```
ggplot(Forecast, aes(temperature, precipitation))+
  geom_line(aes(group= city), colour="grey50")+
  geom_point(aes(colour=city))
```

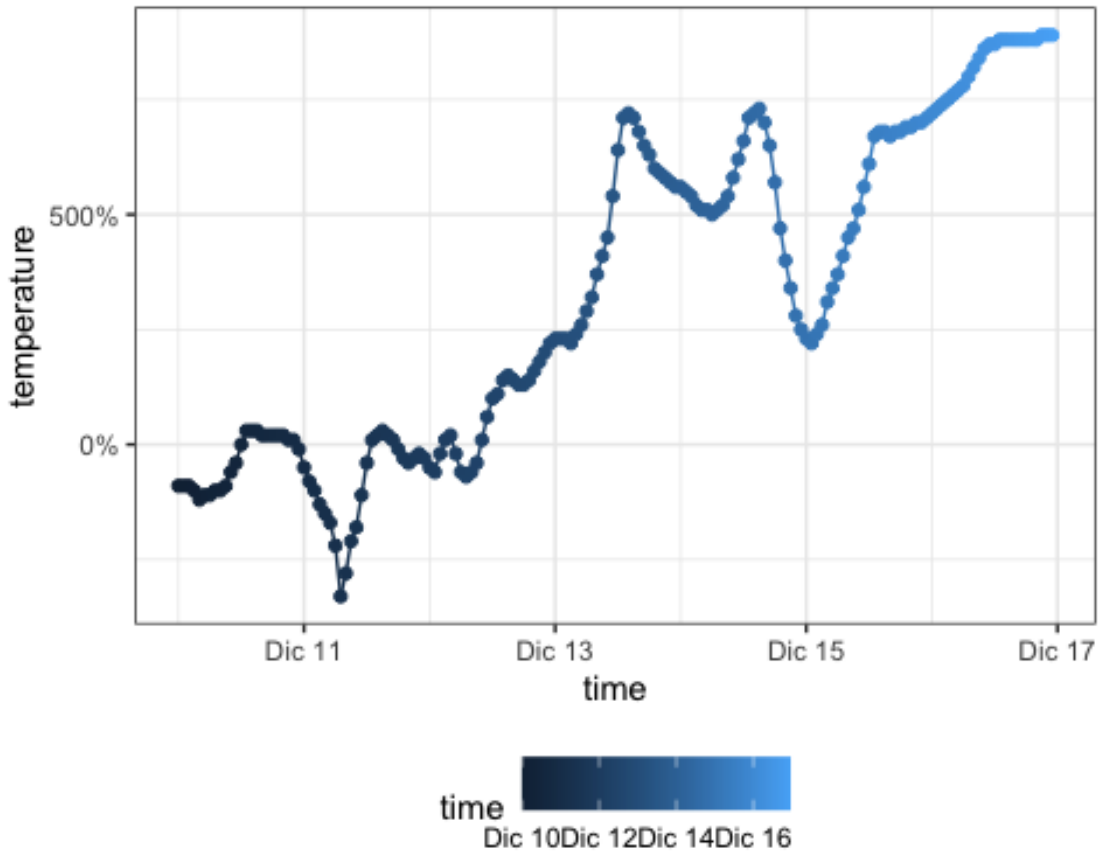


A time series was then created for the city of Rome, Barcelona and Rekjavik.

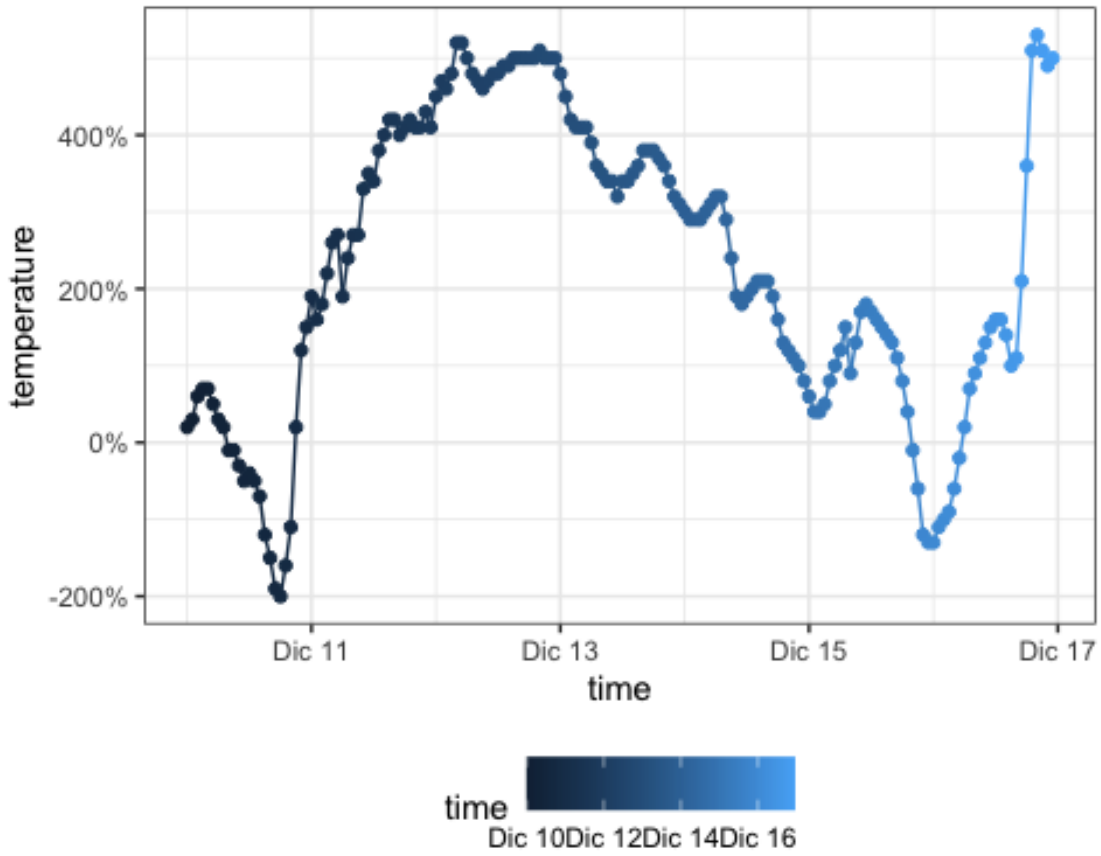
```
Forecast %>% filter(city == "Rome") %>%
  ggplot(aes(x=time, y=temperature, color = time)) + geom_point() + geom_line()+
  scale_y_continuous( labels =  scales :: percent) +
  theme_bw() +
  theme(legend.position = "bottom")
```




```
Forecast %>% filter(city == "Berlin") %>%
  ggplot(aes(x=time, y=temperature, color = time)) + geom_point() + geom_line()+
  scale_y_continuous( labels =  scales :: percent) +
  theme_bw() +
  theme(legend.position = "bottom")
```

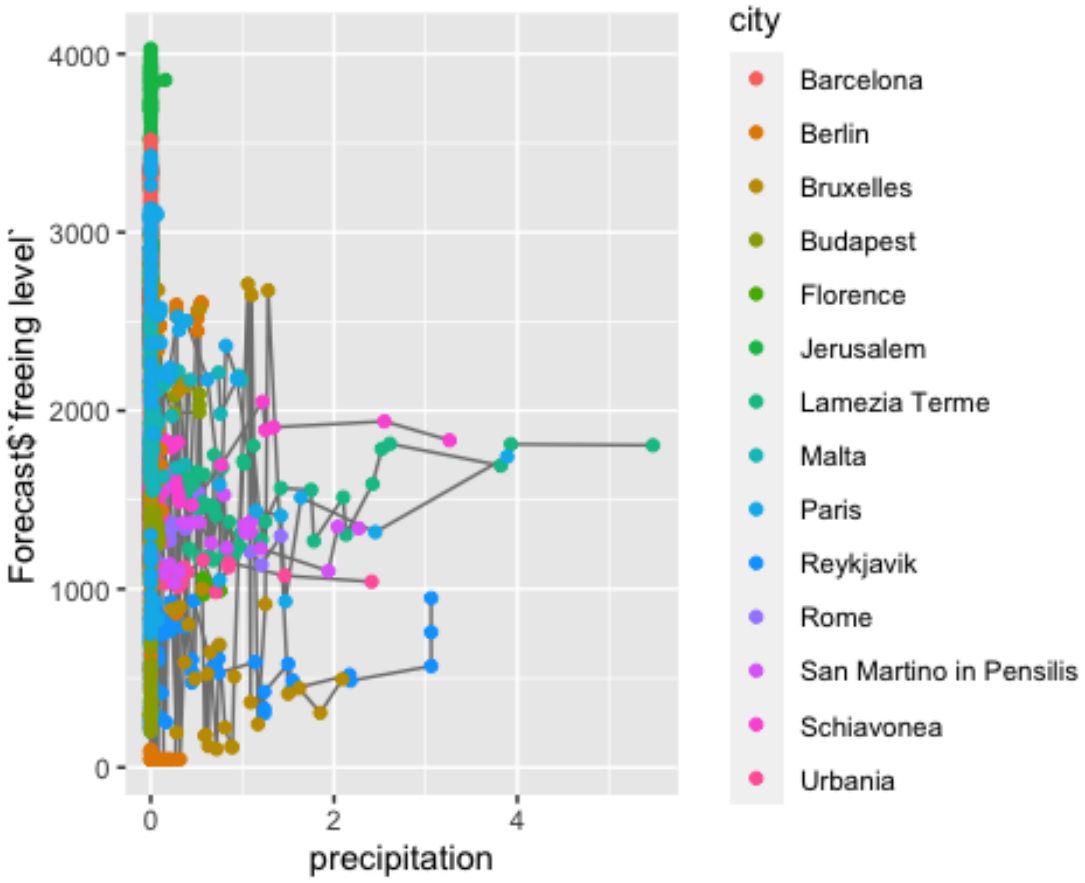



```
Forecast %>% filter(city == "Reykjavik") %>%
  ggplot(aes(x=time, y=temperature, color = time)) + geom_point() + geom_line()+
  scale_y_continuous( labels =  scales :: percent) +
  theme_bw() +
  theme(legend.position = "bottom")
```



Finally, the precipitation and freeing level of the cities has been inspected: the city with highest precipitation is Lamezia Terme, the one with highest dewpoint is Florence. On the other hand, the city with both the lowest precipitation and dewpoint is Berlin.

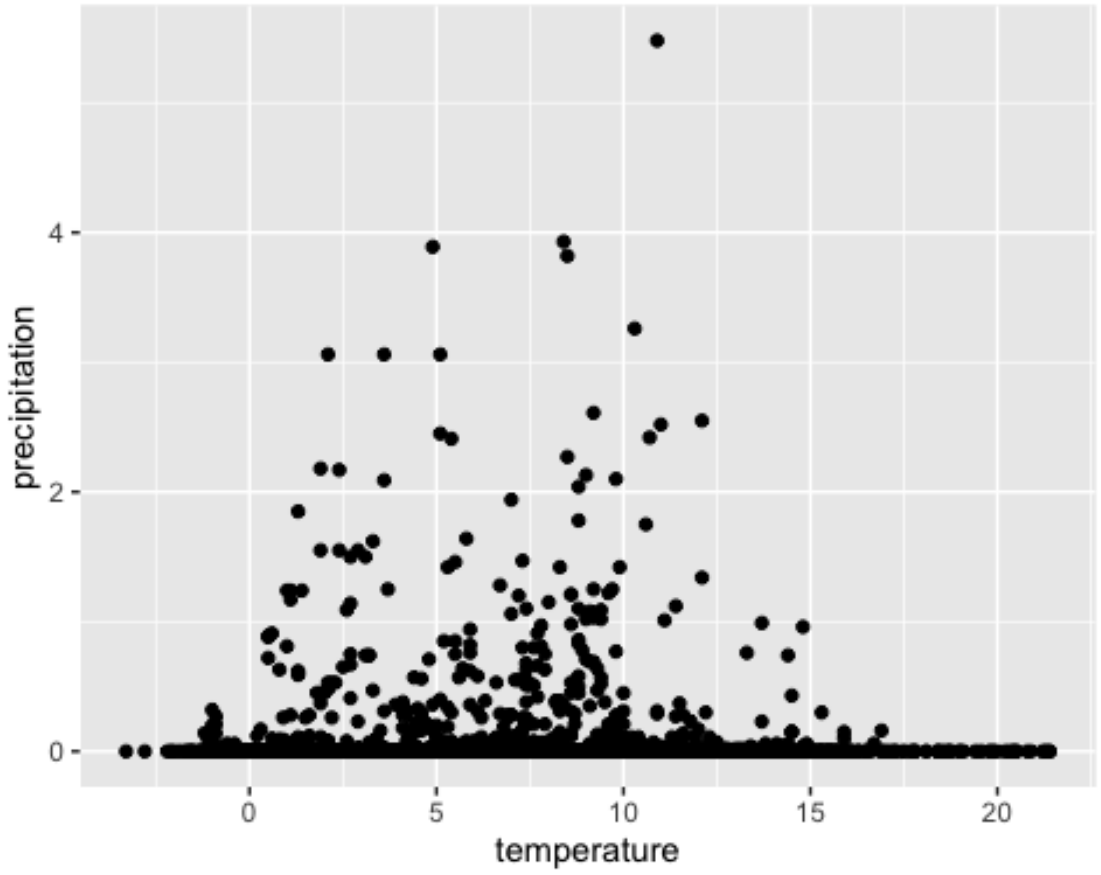
```
ggplot(Forecast, aes(precipitation, Forecast$`freeing level`))+
  geom_line(aes(group= city), colour="grey50")+
  geom_point(aes(colour=city))
```



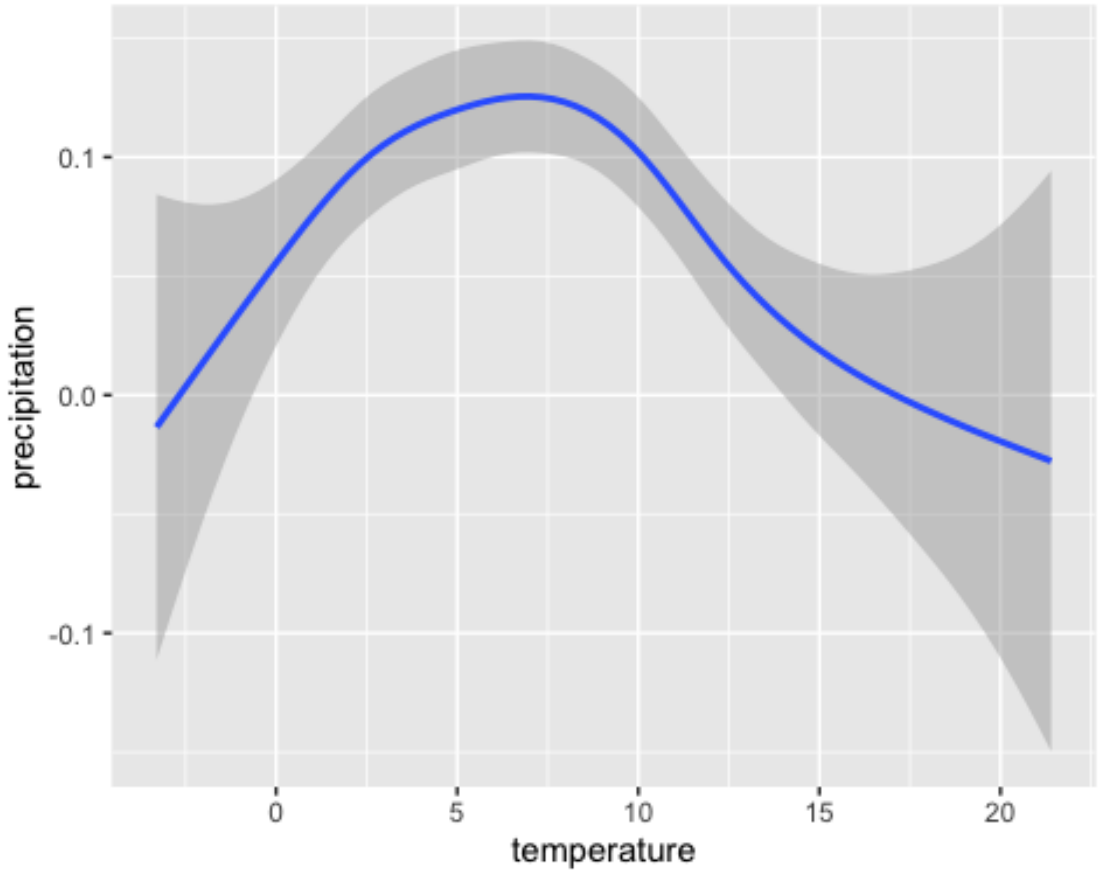
#Geometric Objects

Moreover, throughout the analysis, geometric objects were also deployed in order to represent data. For the following plots the point geom and smooth geom were used to build the graphs. From the following two plots it can be observed that when the precipitation is high the temperature tends to be low and vice versa.

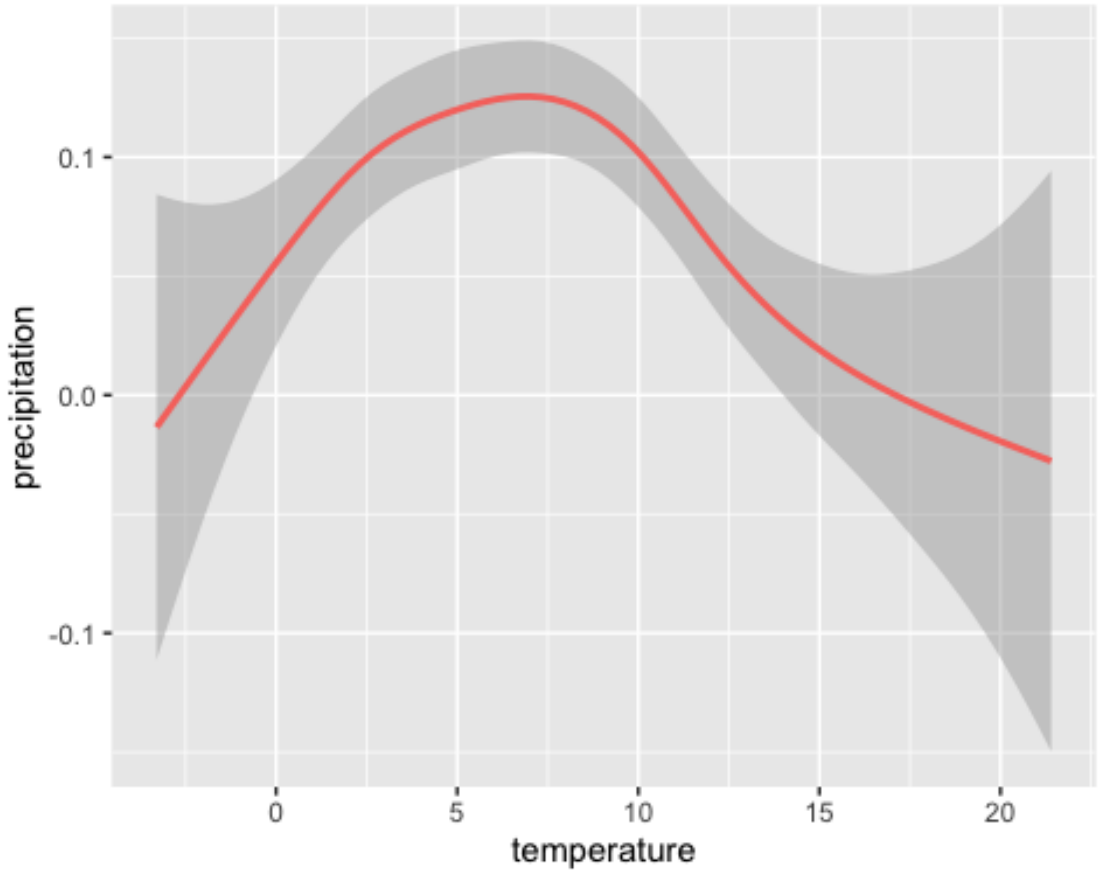
```
ggplot(data = Forecast) +
  geom_point(mapping = aes(x = temperature, y = precipitation))
```




```
ggplot(data = Forecast) +
  geom_smooth(mapping = aes(x = temperature, y = precipitation, group = "drv"))

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
ggplot(data = Forecast) +
  geom_smooth(
    mapping = aes(x = temperature, y = precipitation, color = "drv"),
    show.legend = FALSE
  )

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



DATA MANIPULATION

In order to manipulate data, five key dplyr functions were used: the latter allowed to solve the majority of the data manipulation's challenges: 1) Pick observations by their values with the function filter(): 2) Reorder the rows with the function arrange() 3) Pick variables by their names (select()) 4) Create new variables with the function of already existing variables (mutate()) 5) Collapse many values down to a single summary (summarise())

We use the summarise() function to compute the min and max, as well as the mean and the median of the variable temperature.

```
summarise(Forecast, temp_mean = mean(temperature), temp_med = median(temperature),
temp_min = min(temperature), temp_max = max(temperature))

## # A tibble: 1 × 4
##   temp_mean temp_med temp_min temp_max
##       <dbl>    <dbl>    <dbl>    <dbl>
## 1      7.48      7.6     -3.3     21.4

data <- filter(Forecast, temperature == -3.3 | temperature == 21.4)
data

## # A tibble: 3 × 9
##   city      time                temperature `apparent tempera… `relative
humidi…
```

```
##   <chr>     <dttm>                    <dbl>              <dbl>
<dbl>
## 1 Barcelona 2021-12-15 14:00:00        21.4               21.8
58
## 2 Barcelona 2021-12-16 15:00:00        21.4               22.1
63
## 3 Berlin    2021-12-11 07:00:00        -3.3               -6.3
100
## # … with 4 more variables: precipitation <dbl>, dewpoint <dbl>,
## #   freeing level <dbl>, pressure <dbl>
```

With the function filter() we can see that Barcelona, with 21.4, and Berlin with -3.3 are the two cities that featured respectively the highest and the lowest temperatures. Specifically, -3.3 in Berlin was reached on the 11th of December 2021, at 7 am, while 21.4.

```
by_day = group_by(Forecast, time)
by_day
```

```
## # A tibble: 2,352 × 9
## # Groups:   time [168]
##    city  time                temperature `apparent temperature` `relative
humid…
##    <chr> <dttm>                    <dbl>                  <dbl>
<dbl>
##  1 Rome  2021-12-10 00:00:00         6.4                    5
93
##  2 Rome  2021-12-10 01:00:00         6.4                    4.7
91
##  3 Rome  2021-12-10 02:00:00         6                      4.3
93
##  4 Rome  2021-12-10 03:00:00         5.4                    3.8
95
##  5 Rome  2021-12-10 04:00:00         5.1                    3.7
96
##  6 Rome  2021-12-10 05:00:00         4.7                    3.1
98
##  7 Rome  2021-12-10 06:00:00         3.9                    2
100
##  8 Rome  2021-12-10 07:00:00         3.3                    1.3
100
##  9 Rome  2021-12-10 08:00:00         2.8                    0.5
100
## 10 Rome  2021-12-10 09:00:00         3.6                    1.5
98
## # … with 2,342 more rows, and 4 more variables: precipitation <dbl>,
## #   dewpoint <dbl>, freeing level <dbl>, pressure <dbl>
```

```
summarise(by_day, temp = mean(temperature))
```

```
## # A tibble: 168 × 2
##    time                 temp
##    <dttm>              <dbl>
```

```
##  1 2021-12-10 00:00:00  5.41
##  2 2021-12-10 01:00:00  5.24
##  3 2021-12-10 02:00:00  5.14
##  4 2021-12-10 03:00:00  5.14
##  5 2021-12-10 04:00:00  5.24
##  6 2021-12-10 05:00:00  5.39
##  7 2021-12-10 06:00:00  5.39
##  8 2021-12-10 07:00:00  5.49
##  9 2021-12-10 08:00:00  5.51
## 10 2021-12-10 09:00:00  5.94
## # … with 158 more rows
```

With the above code we can get the average temperature per day.

```
data3 <- filter(Forecast, city == "Rome")
data3

## # A tibble: 168 × 9
##    city  time                temperature `apparent temperature` `relative
humid…
##    <chr> <dttm>                    <dbl>                  <dbl>
<dbl>
##  1 Rome  2021-12-10 00:00:00         6.4                    5
93
##  2 Rome  2021-12-10 01:00:00         6.4                    4.7
91
##  3 Rome  2021-12-10 02:00:00         6                      4.3
93
##  4 Rome  2021-12-10 03:00:00         5.4                    3.8
95
##  5 Rome  2021-12-10 04:00:00         5.1                    3.7
96
##  6 Rome  2021-12-10 05:00:00         4.7                    3.1
98
##  7 Rome  2021-12-10 06:00:00         3.9                    2
100
##  8 Rome  2021-12-10 07:00:00         3.3                    1.3
100
##  9 Rome  2021-12-10 08:00:00         2.8                    0.5
100
## 10 Rome  2021-12-10 09:00:00         3.6                    1.5
98
## # … with 158 more rows, and 4 more variables: precipitation <dbl>,
## #   dewpoint <dbl>, freeing level <dbl>, pressure <dbl>

fore <- dplyr::select(Forecast, time, ends_with("temperature"))
fore1 <- mutate(fore, dif = Forecast$`apparent temperature`/temperature)
fore1

## # A tibble: 2,352 × 4
##    time                temperature `apparent temperature`   dif
##    <dttm>                    <dbl>                  <dbl> <dbl>
```

```
##  1 2021-12-10 00:00:00         6.4                    5   0.781
##  2 2021-12-10 01:00:00         6.4                    4.7 0.734
##  3 2021-12-10 02:00:00         6                      4.3 0.717
##  4 2021-12-10 03:00:00         5.4                    3.8 0.704
##  5 2021-12-10 04:00:00         5.1                    3.7 0.725
##  6 2021-12-10 05:00:00         4.7                    3.1 0.660
##  7 2021-12-10 06:00:00         3.9                    2   0.513
##  8 2021-12-10 07:00:00         3.3                    1.3 0.394
##  9 2021-12-10 08:00:00         2.8                    0.5 0.179
## 10 2021-12-10 09:00:00         3.6                    1.5 0.417
## # … with 2,342 more rows
```

We created a new dataset with the function mutate() to see the proportion between the temperature and the apparent temperature. The apparent temperature is in average lower than the real temperature.

```
transmute(Forecast, dif = Forecast$`apparent temperature`/temperature, hum =
temperature/Forecast$`relative humidity`, city)
```

```
## # A tibble: 2,352 × 3
##      dif    hum city
##    <dbl>  <dbl> <chr>
##  1 0.781 0.0688 Rome
##  2 0.734 0.0703 Rome
##  3 0.717 0.0645 Rome
##  4 0.704 0.0568 Rome
##  5 0.725 0.0531 Rome
##  6 0.660 0.0480 Rome
##  7 0.513 0.039  Rome
##  8 0.394 0.033  Rome
##  9 0.179 0.028  Rome
## 10 0.417 0.0367 Rome
## # … with 2,342 more rows
```

The min_rank() function that returns the same values as rank when the ties method is set to “min”, that is, ties are assigned the minimum ranking possible

```
rank <- Forecast%>% group_by(city) %>% filter(min_rank(desc(temperature))==20) %>%
  dplyr::select(city, temperature, time)
rank
```

```
## # A tibble: 13 × 3
## # Groups:   city [7]
##    city                    temperature time
##    <chr>                         <dbl> <dttm>
##  1 Rome                           11.3 2021-12-12 12:00:00
##  2 Rome                           11.3 2021-12-15 13:00:00
##  3 Reykjavik                       4.8 2021-12-12 03:00:00
##  4 Reykjavik                       4.8 2021-12-12 07:00:00
##  5 Reykjavik                       4.8 2021-12-12 11:00:00
##  6 Reykjavik                       4.8 2021-12-12 12:00:00
##  7 Reykjavik                       4.8 2021-12-13 00:00:00
```

```
##  8 Barcelona                     17.1 2021-12-16 19:00:00
##  9 San Martino in Pensilis       11.1 2021-12-13 15:00:00
## 10 San Martino in Pensilis       11.1 2021-12-15 15:00:00
## 11 Berlin                         7.6 2021-12-16 04:00:00
## 12 Malta                         15   2021-12-14 16:00:00
## 13 Paris                          9.7 2021-12-12 21:00:00
```

Finally, we can use the join() function. Specifically, inner_join() returns rows when there is a match in both tables. In this case, we are marging rank and data3 with city as primary key.

```
fin <- inner_join(rank, data3, by = "city")
fin
```

```
## # A tibble: 336 × 11
## # Groups:   city [1]
##    city  temperature.x time.x              time.y              temperature.y
##    <chr>         <dbl> <dttm>              <dttm>                      <dbl>
##  1 Rome           11.3 2021-12-12 12:00:00 2021-12-10 00:00:00           6.4
##  2 Rome           11.3 2021-12-12 12:00:00 2021-12-10 01:00:00           6.4
##  3 Rome           11.3 2021-12-12 12:00:00 2021-12-10 02:00:00           6
##  4 Rome           11.3 2021-12-12 12:00:00 2021-12-10 03:00:00           5.4
##  5 Rome           11.3 2021-12-12 12:00:00 2021-12-10 04:00:00           5.1
##  6 Rome           11.3 2021-12-12 12:00:00 2021-12-10 05:00:00           4.7
##  7 Rome           11.3 2021-12-12 12:00:00 2021-12-10 06:00:00           3.9
##  8 Rome           11.3 2021-12-12 12:00:00 2021-12-10 07:00:00           3.3
##  9 Rome           11.3 2021-12-12 12:00:00 2021-12-10 08:00:00           2.8
## 10 Rome           11.3 2021-12-12 12:00:00 2021-12-10 09:00:00           3.6
## # … with 326 more rows, and 6 more variables: apparent temperature <dbl>,
## #   relative humidity <dbl>, precipitation <dbl>, dewpoint <dbl>,
## #   freeing level <dbl>, pressure <dbl>
```