Dplyr Practice questions

2022-09-05

1 IMDB movies

```
df <- read_csv("../Datasets/movie_metadata.csv")
```

```
## Rows: 5043 Columns: 28
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (12): color, director_name, actor_2_name, genres, actor_1_name, movie_ti...
## dbl (16): num_critic_for_reviews, duration, director_facebook_likes, actor_3...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

1. *How many movies are there in the dataset?*

```
paste0("There are ", nrow(df), " movies in the dataset.")
```

```
## [1] "There are 5043 movies in the dataset."
```

2. *Each movie has a unique IMDB ID. For example, in the link http://www.imdb.com/title/tt0449088/?ref_=fn_tt_tt_1, the id is 0449088. Create a column that stores this ID.*

This is a case of pattern search, used in extracting useful information out of a string. Here, notice that the column link has links in a particular format. The movie id is always preceeded by "title/" and succeded by "/?ref_" (verify that this is indeed true for all links). There can be multiple strategies to extract the ID. I am listing a few here:

i) split the link using "/" as the delimiter. Then, since all URLs have a consistent structure, the 5th element obtained (this we know after running the code on a link. c("http:", "","www.imdb.com", "title", "tt0499549", "?ref_=fn_tt_tt_1"))

ii) Extract the pattern: tt followed by numbers followed by forward slash. We can use regex to achieve that.

iii) Extract the pattern: a series of numbers ending with forward slash.

In all three cases, we will use mutate to create a new column. I will create functions to carry out each strategy.

```
f1 <- function(x) {
    x %>% str_split("/") %>% sapply(function(y) {y[5]}) %>% str_remove("tt")
}
f2 <- function(x) {
    x %>% str_extract("tt\\d+/")%>% str_remove("tt") %>% str_remove("/")
}
f3 <- function(x) {
    x %>% str_extract("\\d+/") %>% str_remove("/")
}

df <- df %>%
    mutate(ID1 = movie_imdb_link %>% f1,
           ID2 = movie_imdb_link %>% f2,
           ID3 = movie_imdb_link %>% f3)
# All 3 columns have the same ID

all(df$ID1 == df$ID2)
```

```
## [1] TRUE
```

```
all(df$ID2 == df$ID3)
```

```
## [1] TRUE
```

3. *ID should be unique for each movie. Check if that is true. Hint: use group_by, summarise.*

The idea is, if we count the number of times each unique ID repeats, none of the IDs should have a count more than 1.

```
dfCount <- df %>% group_by(ID1) %>% summarise(Count = n())
any(dfCount$Count > 1)
```

```
## [1] TRUE
```

Therefore, some of the IDs are not unique. We can go one step further and ask if the ID-movie name combinations are unique. That is, if some movies have been repeated or if multiple movies have the same IDs.

```
dfMICount <- df %>% group_by(movie_title, ID1) %>% summarise(Count = n(), .groups = "drop") %>%
    arrange(desc(Count))
dfMICount %>% head(10)
```

```
## # A tibble: 10 x 3
##    movie_title                  ID1     Count
##    <chr>                        <chr>   <int>
##  1 Ben-Hur                      2638144     3
##  2 Halloween                    0077651     3
##  3 Home                         2224026     3
##  4 King Kong                    0360717     3
##  5 Pan                          3332064     3
```

```
##  6 The Fast and the Furious      0232500     3
##  7 Victor Frankenstein           1976009     3
##  8 20,000 Leagues Under the Sea  0046672     2
##  9 A Dog's Breakfast             0796314     2
## 10 A Nightmare on Elm Street     0087800     2
```

Clearly, there are repeats in the dataframe. Now, are there any duplicate IDs in the new data frame? If so, multiple movies have the same ID.

```
dfMI_ICount <- dfMICount %>% group_by(ID1) %>%summarise(Count = n()) %>%
    filter(Count > 1)
dfMI_ICount
```

```
## # A tibble: 0 x 2
## # ... with 2 variables: ID1 <chr>, Count <int>
## # i Use 'colnames()' to see all variable names
```

Therefore, each movie does have a unique ID, and some movies have multiple entries. Similarly, any duplicate movies?

```
dfMI_MCount <- dfMICount %>% group_by(movie_title) %>%summarise(Count = n()) %>%
    filter(Count > 1)
dfMI_MCount
```

```
## # A tibble: 3 x 2
##   movie_title      Count
##   <chr>            <int>
## 1 Out of the Blue      2
## 2 The Dead Zone        2
## 3 The Host             2
```

Hence, each ID has a unique movie. so all ID-movie combinations are unique.

4. *Make a list of all keywords used in the dataset. Which keywords are repeated most often? (Note: this is not a dplyr based question.)*

Keywords are stored in the "plot_keyword" column. Each keyword is separated by a "|". Therefore, we write:

```
keyWords <- df$plot_keywords %>% str_split("\\|", ) %>% unlist
keyWordFrequency <- keyWords %>% table %>% sort(decreasing = T)
head(keyWordFrequency)
```

```
## .
##          love        friend        murder         death        police
##           198           166           161           132           126
## new york city
##            91
```

Here, the reason for using "\\|" and not "|" directly is that "|" is a recognized regex. If you don't want to concern yourself with a regex, you can ask stringr to stop interpreting the patterns by using the function fixed.

```
keyWords <- df$plot_keywords %>% str_split(fixed("|"), ) %>% unlist
keyWordFrequency <- keyWords %>% table %>% sort(decreasing = T)
head(keyWordFrequency)
```

```
## .
##          love        friend        murder         death        police
##           198           166           161           132           126
## new york city
##            91
```

5. *Create a column that contains the first genre in the list of genres used for a given movie.*

Here, there can be two strategies. Either remove everythin after the first "|", or extract everything upto the first "|" and then remove the "|".

```
df <- df %>% mutate(FirstGenre = genres %>% str_remove("\\|.*"),
                    FirstGenre2 = genres %>% str_extract("\\w+\\|") %>% str_remove("\\|"))
all(df$FirstGenre == df$FirstGenre2)
```

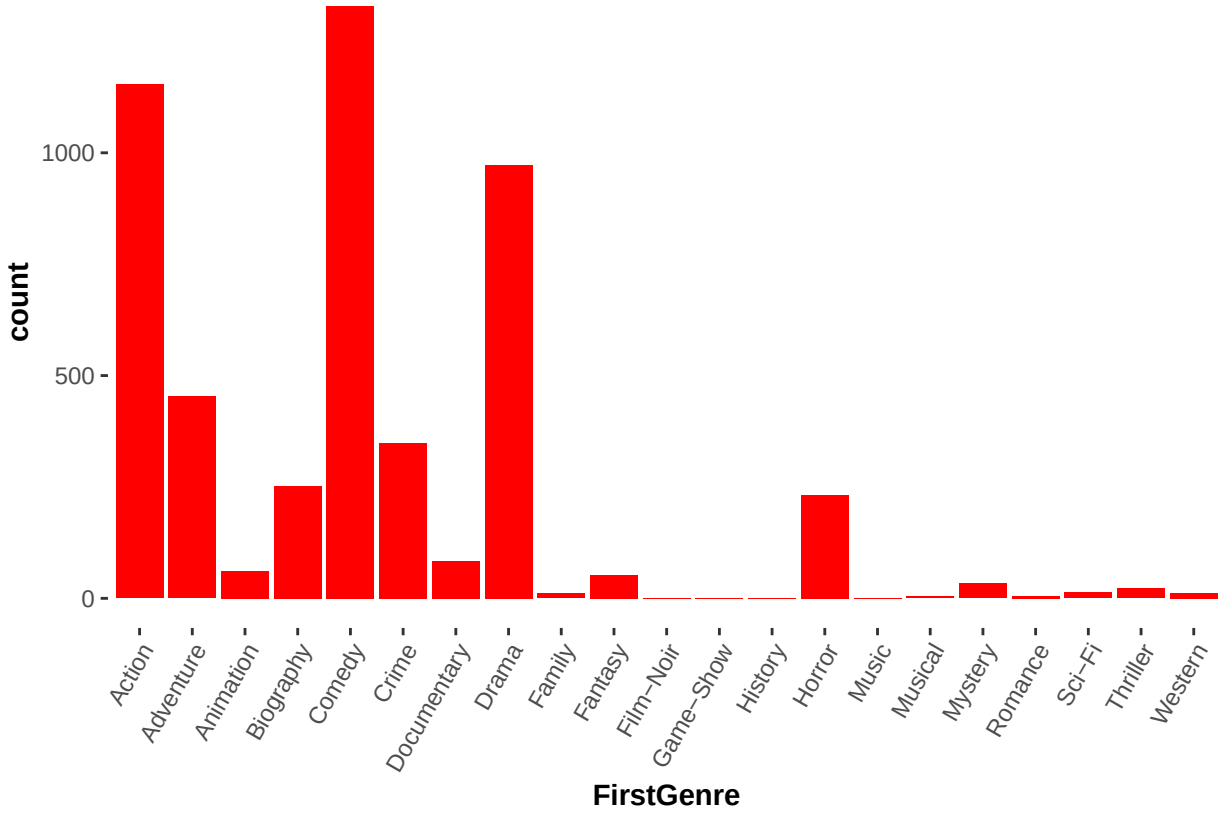
```
## [1] FALSE
```

See that the second strategy, though good on paper, is harder to implement and fails when there is only 1 genre for the movie (movie number 5 for example.)

6. *Create a barplot for the first genres, with the following conditions:*
    a. *Keep the panel background blank*
    b. *On x-axis, keep the text at an angle of 60, set the hjust and vjust arguments to 1*
    c. *Make axis titles bold*
    d. *Remove the grid.*
    e. *Fill the bars with red color*

Not much to explain here. Just straightforward.

```
df %>% ggplot(aes(x = FirstGenre)) + geom_bar(fill = "red") +
    theme(panel.background = element_blank(),
          axis.text.x = element_text(angle = 60, hjust = 1, vjust = 1),
          axis.title = element_text(face = "bold"),
          panel.grid = element_blank())
```



7. *Find the average IMDB rating for each director. Which director has the highest average IMDB rating?*

This again is a question to be solved using group_by and summarise. For summarising, we use the mean function. Then, arrange the resultant dataframe by descending order of the score and we have the answer.

```
df %>% group_by(director_name) %>%
    summarise(Mean_score = mean(imdb_score, na.rm = T)) %>%
    arrange(desc(Mean_score)) %>%
    head(5)
```

```
## # A tibble: 5 x 2
##   director_name    Mean_score
##   <chr>                 <dbl>
## 1 John Blanchard          9.5
## 2 Cary Bell               8.7
## 3 Mitchell Altieri        8.7
## 4 Sadyk Sher-Niyaz        8.7
## 5 Charles Chaplin         8.6
```

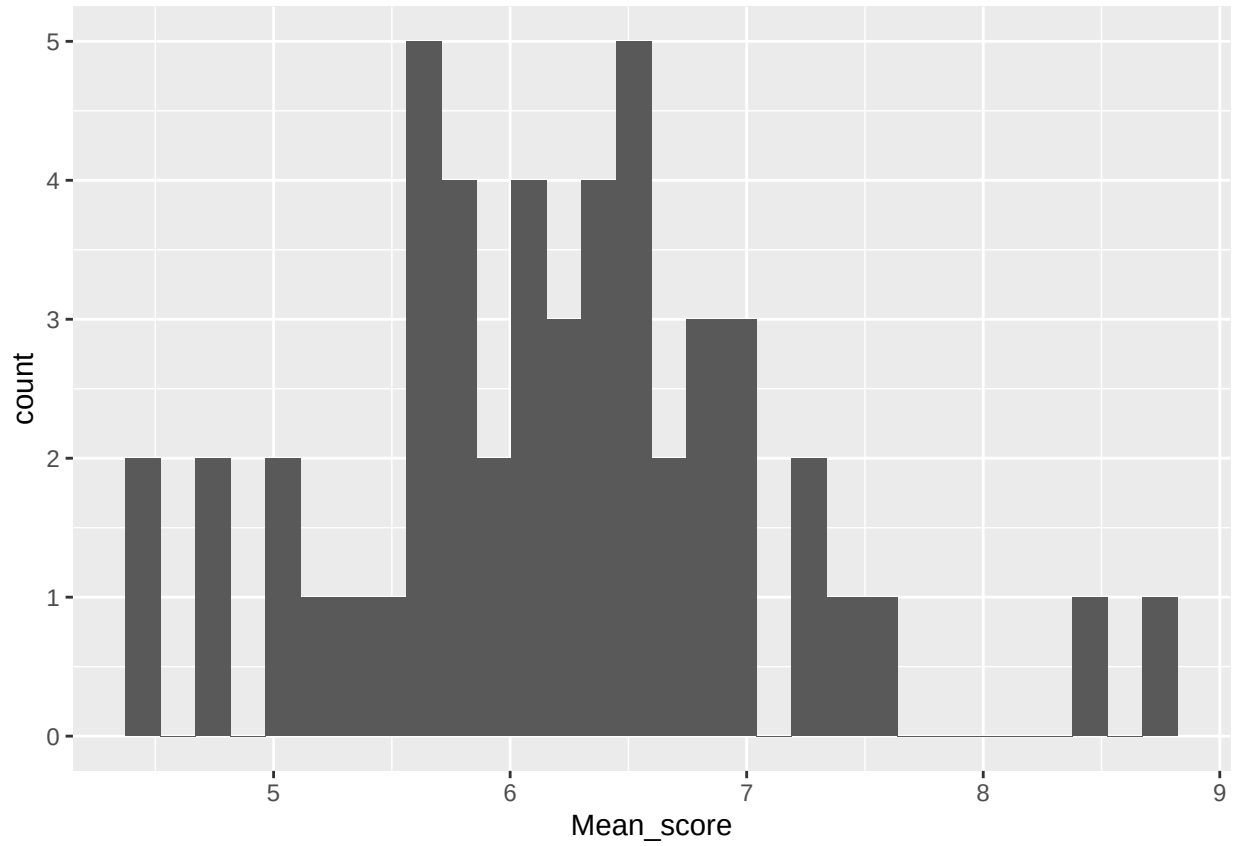
8. *English movies are made in multiple countries. Is there a bias in terms of the IMDB score of English movies for a given country?*

The idea of bias is as follows. If there is a bias, then movies made in a certain country (say USA or UK) have a higher IMDB score than movies in other countries (say India). Ideally, bias is something that should be checked using a statistical test. However, there are visual ways to check the same as well. We can use the following strategies.

i) If there is a bias, the mean imdb score for some countries would be higher. In other words, the distribution of mean imdb score per country would be skewed.
ii) Using the same distribution as above, we can compare it to a baseline imdb score to check if there is a bias. A valid baseline here is the mean across all movies.

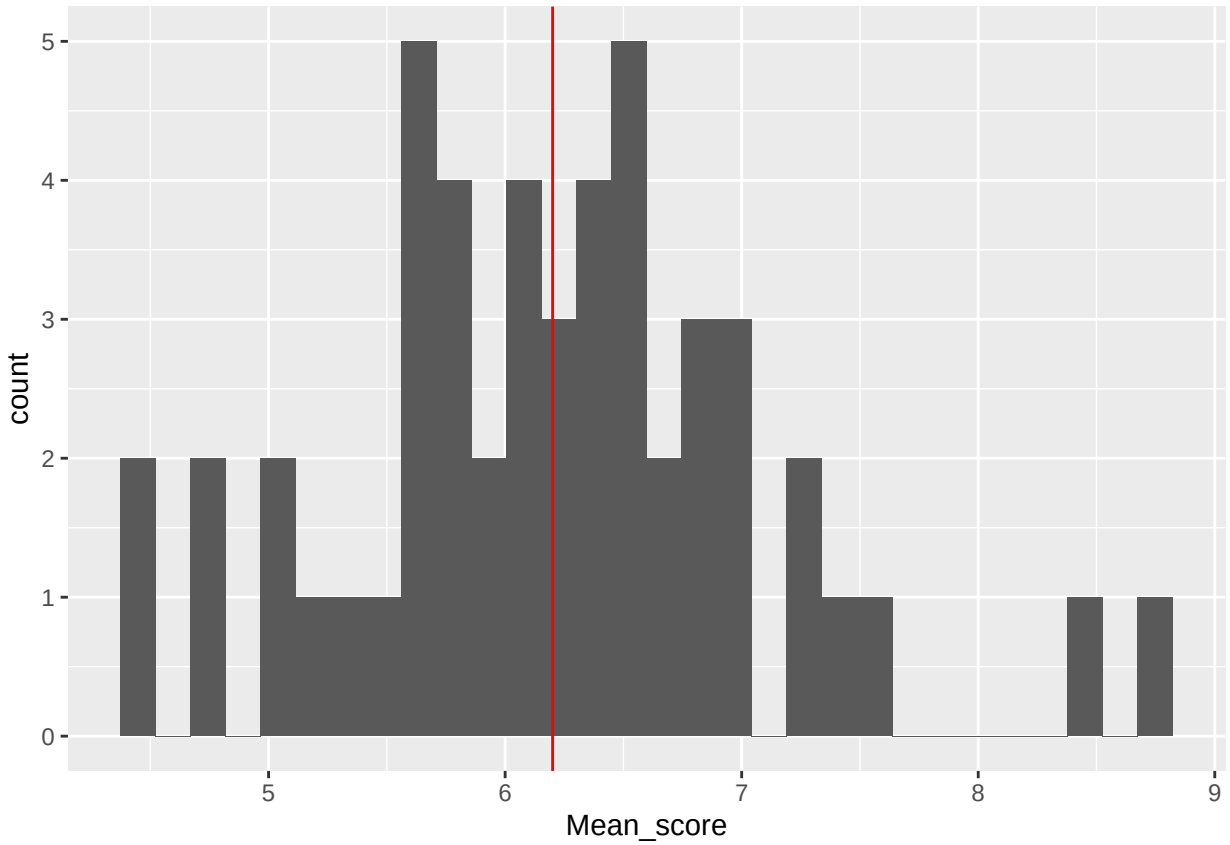
```
dfCountry <- df %>% filter(language == "English") %>%
    group_by(country) %>%
    summarise(Mean_score = mean(imdb_score))
#i)
ggplot(dfCountry, aes(x = Mean_score)) + geom_histogram()
```

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

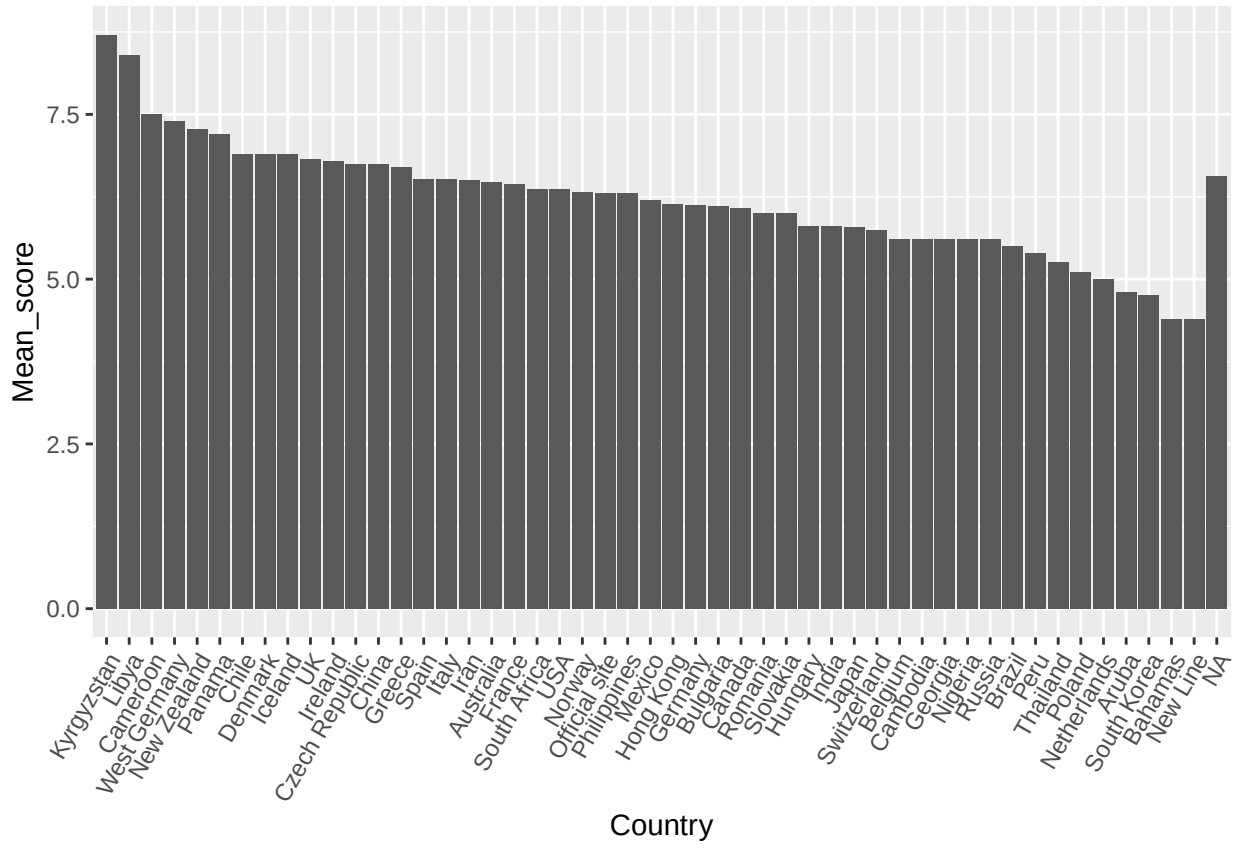



```
#ii)
ggplot(dfCountry, aes(x = Mean_score)) + geom_histogram() +
    geom_vline(xintercept = mean(dfCountry$Mean_score), color = "red")
```

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




```
#iii) While the distribution is symmetric, some countries have low and high means. The question then is
ggplot(dfCountry, aes(x = reorder(country, -Mean_score), y = Mean_score)) +
    geom_bar(stat = "identity") +
    labs(x = "Country") +
    theme(axis.text.x = element_text(angle = 60, hjust = 1, vjust = 1))
```



2 Quality of life dataset

```
df <- read_csv("../Datasets/quality_of_life.csv")
```

```
## Rows: 240 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): City
## dbl (10): Rank, Quality of Life Index, Purchasing Power Index, Safety Index,...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

1. *How many cities are there in the dataset? How many quality indices?*

```
paste0("The number of cities is ", nrow(df))
```

```
## [1] "The number of cities is 240"
```

```
paste0("The number of quality indices is ", ncol(df)-2)
```

```
## [1] "The number of quality indices is 9"
```

2. *Notice that there are spaces in the column names. Having spaces in variable names is not desirable. Replace spaces with "_".*

```
colnames(df) <- colnames(df) %>% str_replace_all(" ", "_")
```

3. *Arrange the dataset based on the City.*

```
df2 <- df %>% arrange(City)
```

4. *A ranking has been assigned to each city. Which column was used to assign that? In other words, sorting by which column gives you the same dataframe?*

So for each index, we sort the dataframe and check if the resultant dataframe is the same as the original dataframe. So normally, we would use a loop or an apply function with the column name as the iterator. However, that does not work with arrange. See below.

```
x <- colnames(df)[4] # second index
df2 <- df %>% arrange(desc(x))
df3 <- df %>% arrange(desc(Purchasing_Power_Index))
all(df2 == df3)
```

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

So, we need a different approach here. What we will do is, we can rename that column inside the for loop of the apply function, so that we can use arrange using the new name.

```
indices <- colnames(df)[-c(1,2)]
id <- sapply(1:length(indices), function(x) {
    df1 <- df
    colnames(df)[x + 2] <- "Col"
    df3 <- df %>% arrange(desc(Col))
    all(df3 == df)
})
paste0("The cities are ranked based on ", indices[id])
```

```
## [1] "The cities are ranked based on Quality_of_Life_Index"
```

5. *Create column containing the country of each city.*

The country for each city is in the city column. We can either split the city by comma and get the country, or extract the characters after comma and delete the comma.

```
df <- df %>% mutate(Country = City %>% str_split(",") %>%
                          sapply(function(x) {x[2]}))
```

6. *Find the average Quality of life index for each country. Which country is better to live in?*

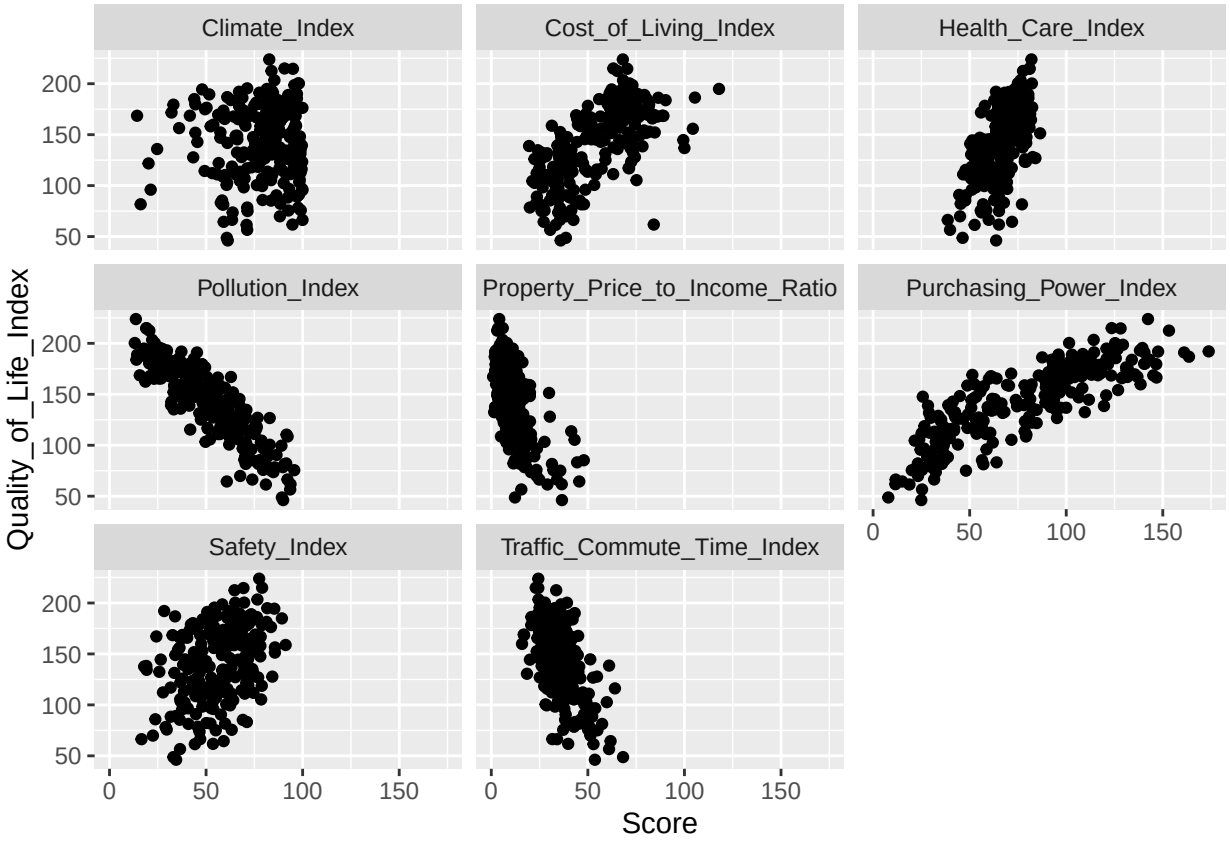
```
df %>% group_by(Country) %>%
    summarise(Mean_index = mean(Quality_of_Life_Index)) %>%
    arrange(desc(Mean_index)) %>%
    head
```

```
## # A tibble: 6 x 2
##   Country        Mean_index
##   <chr>               <dbl>
## 1 " NC"                203.
## 2 " Australia"         202.
## 3 " Netherlands"       194.
## 4 " NM"                192.
## 5 " OK"                191.
## 6 " Switzerland"       191.
```

7. *Make scatter plot of the Quality of life index (y axis) against the other indices (x-axis, one plot for each index). Use the same conditions as above. Which index can explain the quality of life best?*

Two methods. We can either generate multiple scatterplots or use facet_wrap. The second method is better

```
df3 <- df %>% gather(key = "Index", value = "Score", -Rank, -Quality_of_Life_Index, -City, -Country)
ggplot(df3, aes(x = Score, y = Quality_of_Life_Index)) +
    geom_point() +
    facet_wrap(~Index, ncol = 3)
```



Purchasing power index has the best relationship with the quality of life index.