part_I

2023-11-28

Q1

I load the data into relevant dataframes, df and df_code ansd then display the required subsets using select() and head() commands. I am also using the %>% (pipe) operator for clarity.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Load the data
df <- read.csv("/Users/joesouber/Downloads/global_financial_development.csv")
df_code <- read.csv("/Users/joesouber/Downloads/GFD_indicators.csv")

#subset of the data frame df consisting of the first 5 rows and the columns: country, indicator_code, y
subset_df <- df %>% select(country, indicator_code, year_2019) %>% head(5)
print(subset_df)
```

```
##   country indicator_code year_2019
## 1   Aruba           ai01        NA
## 2   Aruba           ai02  14.84391
## 3   Aruba           ai25 126.74416
## 4   Aruba           oi06        NA
## 5   Aruba           om01        NA
```

```
# Display a subset of the data frame df_code consisting of the first 5 rows and all columns
subset_df_code <- df_code %>% head(5)
print(subset_df_code)
```

```
##   indicator_code             indicator_name
## 1           ai01   BankAccountsPer1000Adults
## 2           ai02 BankBranchesPer100000Adults
## 3           ai25         ATMsPer100000Adults
## 4           oi06               Top5BankAsset
## 5           om01   CompaniesPer1000000People
```

Q2

Merging the two dataframes based on the indicator_code attribute. Then formatting and displaying a subset.

```r
# Merging data frames df and df_code
df_merged <- merge(df, df_code, by = "indicator_code")

# Remove the indicator_code column
df_merged <- select(df_merged, -indicator_code)

#subset displaying first 6 entries for country,indicator_name and year_2019
subset_df_merged <- df_merged %>% select(country, indicator_name, year_2019) %>% head(6)
print(subset_df_merged)
```

```
##                country            indicator_name year_2019
## 1                Aruba BankAccountsPer1000Adults        NA
## 2 Hong Kong SAR, China BankAccountsPer1000Adults        NA
## 3              Finland BankAccountsPer1000Adults        NA
## 4               Norway BankAccountsPer1000Adults        NA
## 5             Maldives BankAccountsPer1000Adults  986.1256
## 6              Belarus BankAccountsPer1000Adults        NA
```

Q3

Creating df_stock and displaying subset. I use the $ operator to highlight a specific column.

```r
# Create df_stock containing rows with indicator_name equal to "StockMarketReturn"
df_stock <- df_merged[df_merged$indicator_name == "StockMarketReturn", ]


# Reorder column year_2019 in descending order
df_stock <- df_stock[order(-df_stock$year_2019), ]

#subset
subset_df_stock <- df_stock %>% select(country, year_2019, year_2020, year_2021) %>% head(5)
print(subset_df_stock)
```

```
##                      country   year_2019   year_2020  year_2021
## 1224          Venezuela, RB 13304.47000 1307.307000 991.390900
## 1087                Jamaica    41.30103  -11.764520   1.432191
## 1126                 Brazil    23.28735   -1.875297  18.170990
## 1136 Bosnia and Herzegovina    17.51320   -3.595580  -1.577103
## 1105              Argentina    13.50410   22.834990  54.642480
```

Q4

Initially, I created functions to clean the data of unnecessary columns, then to calculate the required statistic and then to create a dataframe, which appends this calculated statistic onto a summary dataframe. I have broken down the operation of each function as follows:

1. `clean_data` function:

- **Purpose**: Cleans a data frame by filtering rows based on a specified indicator name and removing specific columns.
- **Parameters**:
  - `df_merged`
  - `indicator_name`
- **Steps**:

1. Filters rows where the indicator name matches the specified name.
2. Removes unnecessary columns (iso3, iso2, imfn, income, indicator_name).
3. Returns the cleaned data frame.

2. `top5_stat` function:

- **Purpose**: Calculates a statistic (average) based on numeric values, considering the top 5 values.
- **Parameters**:
  - Variable number of arguments (numeric values).
- **Steps**:
  1. Combines all input values into a single vector.
  2. Extracts numeric values from the vector.
  3. Removes missing values.
  4. Checks the number of non-missing numeric values.
  5. Returns NaN if no non-missing numeric values, the mean if 5 or fewer non-missing values, or the mean of the top 5 absolute values.

3. `create_summary_df` function:

- **Purpose**: Creates a summary data frame with an added column for the calculated average.
- **Parameters**:
  - `df_cleaned`: Cleaned data frame.
  - `column_name`: Name of the column to be added for the calculated average.
- **Steps**:
  1. For each row, calculates the average using the `top5_stat` function on numeric columns.
  2. Selects and renames columns, creating a summary data frame with country and the calculated average.
  3. Returns the summary data frame.

```r
# Function to clean data by filtering based on an indicator name and removing specific columns
clean_data <- function(df_merged, indicator_name) {
  # Filter rows based on indicator_name
  df_indicator <- df_merged[df_merged$indicator_name == indicator_name, ]

  # Remove unnecessary columns
  df_cleaned <- df_indicator %>%
    select(-iso3, -iso2, -imfn, -income, -indicator_name)

  # Return the cleaned data frame
  return(df_cleaned)
}

# Function to calculate a statistic (average) based on numeric values,
# considering the top 5 values.
top5_stat <- function(...) {
  values <- c(...)

  # numeric values
  numeric_values <- values[sapply(values, is.numeric)]

  # Remove missing values
  non_missing_values <- numeric_values[!is.na(numeric_values)]

  # Check for the number of non-missing values
  if (length(non_missing_values) == 0) {
```

```
    return(NaN)
  } else if (length(non_missing_values) <= 5) {
    return(mean(non_missing_values))  # Return the mean if 5 or fewer non-missing values
  } else {

    # Calculate the mean of the top 5 absolute values
    top5_values <- tail(sort(abs(non_missing_values), decreasing = TRUE), 5)
    return(mean(top5_values))
  }
}

# Function to create a summary data frame with an added column for
# the calculated average
create_summary_df <- function(df_cleaned, column_name) {
  df_summary <- df_cleaned %>%
    rowwise() %>%
    mutate(AverageValue = top5_stat(c_across(where(is.numeric)))) %>%
    select(country, AverageValue) %>%
    rename({{column_name}} := AverageValue)

  # Return the summary data frame
  return(df_summary)
}
```

Here are the functions in use, resulting in the required dataframe, of which the first 6 rows are displayed.

```
# Example usage:

# Clean data for different indicators
df_stock_cleaned <- clean_data(df_merged, "StockMarketReturn")
df_bank_accounts_cleaned <- clean_data(df_merged, "BankAccountsPer1000Adults")
df_bank_branches_cleaned <- clean_data(df_merged, "BankBranchesPer100000Adults")
df_top5_cleaned <- clean_data(df_merged, "Top5BankAsset")
df_companies_cleaned <- clean_data(df_merged, "CompaniesPer1000000People")

# Create summary data frames for each indicator
df_summary_stocks <- create_summary_df(df_stock_cleaned, "stockMarketReturn")
df_summary_bank_accounts <- create_summary_df(df_bank_accounts_cleaned, "BankAccountsper1000Adults")
df_summary_bank_branches <- create_summary_df(df_bank_branches_cleaned, "BankBranchesper100000Adults")
df_summary_top5 <- create_summary_df(df_top5_cleaned, "top5BankAsset")
df_summary_companies <- create_summary_df(df_companies_cleaned, "CompaniesPer1000000People")

# Combine the summary data frames using left joins
df_combined <- df_summary_stocks %>%
  left_join(df_summary_bank_accounts, by = "country") %>%
  left_join(df_summary_bank_branches, by = "country") %>%
  left_join(df_summary_top5, by = "country") %>%
  left_join(df_summary_companies, by = "country")

# View the combined data frame
head(df_combined, 6)
```

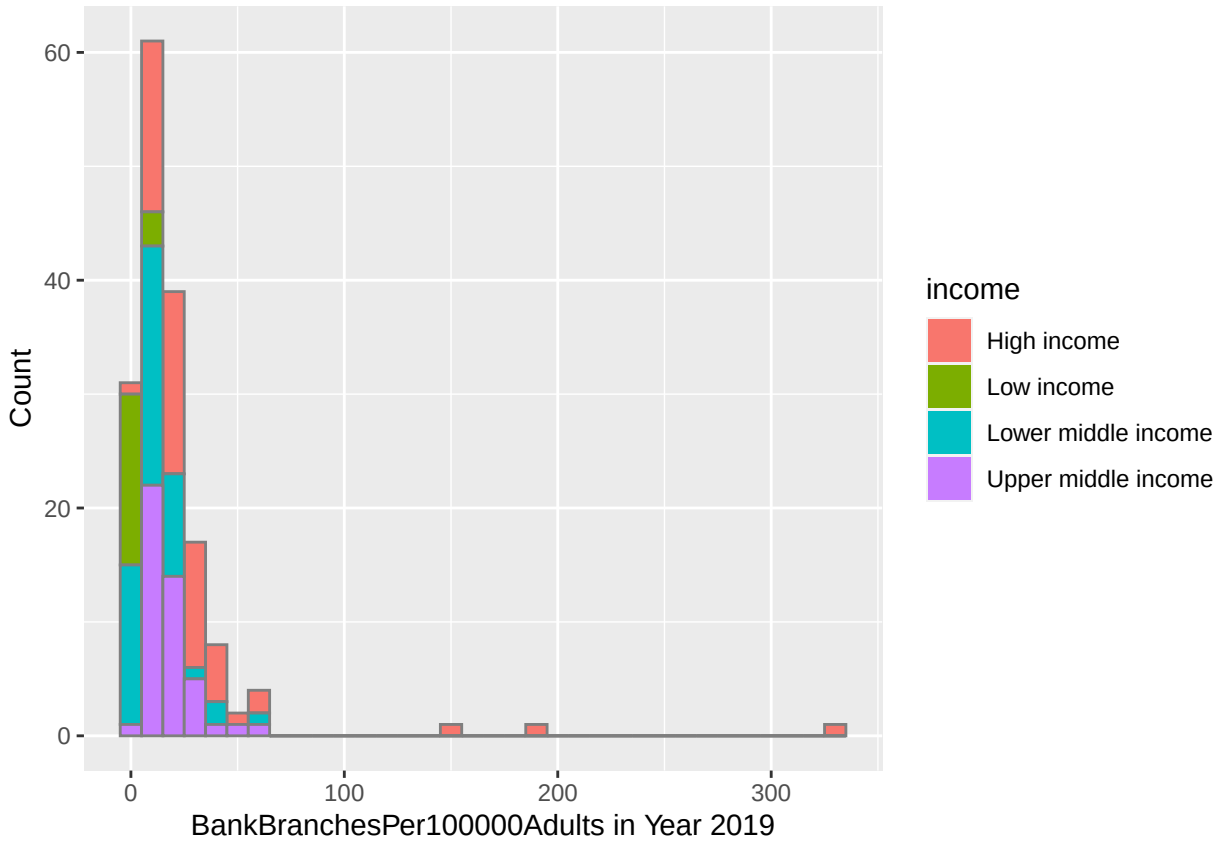
```
## # A tibble: 6 x 6
## # Rowwise:
##   country     stockMarketReturn BankAccountsper1000Adults BankBranchesper10000~1
```

```
##   <chr>                   <dbl>                   <dbl>                   <dbl>
## 1 Iceland                  4.79                    NaN                    32.0
## 2 Poland                   1.74                    921.                   26.0
## 3 Switzerland              1.22                    NaN                    39.6
## 4 Portugal                 2.70                    NaN                    33.8
## 5 Israel                   3.42                    964.                   16.7
## 6 Chile                    2.18                    NaN                    13.8
## # i abbreviated name: 1: BankBranchesper100000Adults
## # i 2 more variables: top5BankAsset <dbl>, CompaniesPer1000000People <dbl>
```

Q5

Using the ggplot2 package to plot the histogram.

```r
library(ggplot2)

# extracting relevant columns into new dataframe
new_df <- df_merged[, c("income", "year_2019", "indicator_name")]




# Filter data for the specific indicator, remove missing values, and values
# greater than 500
filtered_data <- new_df[
  new_df$indicator_name == "BankBranchesPer100000Adults" &
  !is.na(new_df$year_2019) &
  new_df$year_2019 <= 500,
]

# Create histogram
univar_plot <- ggplot(data = filtered_data, aes(x = year_2019, fill = income,
                                                color = "darkgrey")) + xlab("BankBranchesPer100000Adults

univar_plot+geom_histogram(binwidth = 10)+ylab("Count")+
  scale_color_manual(values = c("High income" = "orange",
                                "Low income" = "green", "Upper middle income" = "turquoise", "Lower mid
```



Q6

The given code employs the tidyr and ggplot2 libraries to create a line plot illustrating the trend of "CompaniesPer1000000People" over time for selected countries.

Initially, the data is filtered to include only specified countries ("Australia," "Belgium," "Switzerland," "United Kingdom") and the indicator 'CompaniesPer1000000People'. The data is then transformed into a tidy format, selecting columns starting with "year_" and converting them into long format using pivot_longer. Years are extracted as numeric values, and the data is further filtered to include the years from 1975 to 2014. Finally, a line plot is generated using ggplot2, where the x-axis represents years, the y-axis represents the indicator values, and different countries are distinguished by color.

```
library(tidyr)

# Filter data for specified countries and indicator
filtered_data_q6 <- df_merged %>%
  filter(country %in% c("Australia", "Belgium", "Switzerland", "United Kingdom"),
         indicator_name == "CompaniesPer1000000People")

tidy_data <- filtered_data_q6 %>%
  select(country, starts_with("year_")) %>%
  pivot_longer(cols = starts_with("year_"), names_to = "year", values_to = "value") %>%
  mutate(year = as.numeric(stringr::str_extract(year, "\\d+"))) %>%
  filter(year >= 1975 & year <= 2014)
library(ggplot2)

# Plot line plot using ggplot2
```

```r
ggplot(tidy_data, aes(x = year, y = value, color = country)) +
  geom_line() +
  labs(title = "CompaniesPer1000000People Over Time",
       x = "Year",
       y = "CompaniesPer1000000People") +
  theme_minimal()
```