Euro Exchange

Manikanth Kanapur

2023-10-04

1. Installing packages `tidyverse`, `skimr`, `janitor`, `knitr`

The first step is to load the packages which would be used to perform the tasks needed to answer questions related to this dataset.

2. Loading libraries

This step is crucial as without it we cannot run our code. So, we have to load the packages which we have installed earlier. Tidyverse contains a list of other packages which are needed for data analysis tasks. Skimr is used to generate a summary of a dataframe which is crucial for data cleaning purposes, while the Janitor is used to clean the names of the attributes of the dataframe. The later is important because, we want to make sure that all the names are in the same format. Knitr is used to display the result of the dataframe in a tabular format in the final document.

```
library(tidyverse)
library(skimr)
library(janitor)
library(knitr)
```

3. Importing raw dataset

In this step, the raw dataset is imported which is in the form of a csv file

```
exchange_raw_df <- read_csv("euro_exchange.csv")
skim_without_charts(exchange_raw_df)
```

Data summary

Name	exchange_raw_df
Number of rows	6311
Number of columns	41
_______________________	
Column type frequency:	
character	37
Date	1
numeric	3
________________________	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
[Australian dollar]	0	1.00	1	6	0	3636	0
[Bulgarian lev]	402	0.94	1	6	0	106	0
[Brazilian real]	268	0.96	1	6	0	5447	0
[Canadian dollar]	0	1.00	1	6	0	3068	0
[Swiss franc]	0	1.00	1	6	0	3230	0
[Chinese yuan renminbi]	268	0.96	1	7	0	5305	0
[Cypriot pound]	3965	0.37	1	7	0	498	0
[Czech koruna]	0	1.00	1	6	0	3995	0
[Danish krone]	0	1.00	1	6	0	485	0
[Estonian kroon]	3181	0.50	1	7	0	2	0
[UK pound sterling]	0	1.00	1	7	0	3788	0
[Greek drachma]	5791	0.08	1	6	0	323	0
[Hong Kong dollar]	0	1.00	1	7	0	5770	0
[Croatian kuna]	370	0.94	1	6	0	2305	0
[Hungarian forint]	0	1.00	1	6	0	4471	0
[Indonesian rupiah]	0	1.00	1	8	0	6214	0
[Israeli shekel]	268	0.96	1	6	0	5105	0
[Indian rupee]	268	0.96	1	7	0	5723	0
[Japanese yen]	0	1.00	1	6	0	3700	0
[Korean won]	0	1.00	1	7	0	5867	0
[Lithuanian litas]	2152	0.66	1	7	0	771	0
[Latvian lats]	2407	0.62	1	6	0	1078	0
[Maltese lira]	3965	0.37	1	6	0	426	0
[Mexican peso]	0	1.00	1	7	0	6096	0
[Malaysian ringgit]	0	1.00	1	6	0	5010	0
[Norwegian krone]	0	1.00	1	7	0	3935	0

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
[New Zealand dollar]	0	1.00	1	6	0	4023	0
[Philippine peso]	0	1.00	1	6	0	5463	0
[Polish zloty]	0	1.00	1	6	0	4563	0
[Russian rouble]	317	0.95	1	8	0	5705	0
[Swedish krona]	0	1.00	1	7	0	4874	0
[Singapore dollar]	0	1.00	1	6	0	3845	0
[Slovenian tolar]	4226	0.33	1	8	0	1377	0
[Slovak koruna]	3703	0.41	1	6	0	2014	0
[Thai baht]	0	1.00	1	7	0	5486	0
[US dollar]	0	1.00	1	6	0	3734	0
[South African rand]	0	1.00	1	7	0	6062	0

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
Period:	0	1	1999-01-04	2023-05-26	2011-02-07	6311

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
[Iceland krona]	2407	0.62	108.01	34.29	68.07	83.70	89.36	138.10	305.00
[Romanian leu]	62	0.99	3.97	0.88	1.29	3.55	4.28	4.56	4.98
[Turkish lira]	62	0.99	3.91	4.31	0.37	1.73	2.21	3.95	21.58

Here, the "skim…" function is used to generate a summary of the imported dataset. With this, we can have an overview of what actually the data contains. Our main focus here is to check the datatypes and see if they need any modifications.

4. Data cleaning

In this step, the raw dataset is cleaned. The summary of the dataset as shown above tells us that the data contains 2 types of variables, i.e., character, numeric and a date class. The list of characters should be numeric as we have a dataset which tells us how the currencies of different countries changed over time against Euro. In addition to this, the attributes are embedded inside [] which needs to be cleaned as well.

a) Removing duplicate rows from the dataset.

```
exchange_clean_df <- unique(exchange_raw_df)
```

b) Removal of [] from the column names.

```
col_names_new <- colnames(exchange_clean_df)
col_names_edit <- str_replace_all(col_names_new, "\\[|\\ ]\\s*", "")
colnames(exchange_clean_df) <- col_names_edit
```

First, gsub was used to get rid of [] from the column names but it removed [] partially. Probably, there are some hidden characters in the column names. Therefore, str was used instead. In this, \ followed by "[" and "]" is used to treat both "[" & "]" as characters, while \s* is used to remove empty space after "]". "|" is used to match either "[" or "]".

c) Fixing datatypes of the columns.

```
character_col <- sapply(exchange_clean_df, is.character)
exchange_clean_df[, character_col] <- lapply(exchange_clean_df[,
character_col], as.numeric)
```

Here, sapply means simplify and apply. It is used to check if the data in columns are characters or numeric. It will return TRUE for character & FALSE for numeric. Once you identify the columns with characters, you can convert them to numeric using lapply which means list apply.

d) Renaming column names to lower & snake case for convenience.

```
exchange_clean_df <- clean_names(exchange_clean_df)
exchange_clean_df <- rename(exchange_clean_df, date = period_unit)
skim_without_charts(exchange_clean_df)
```

Data summary

Name	exchange_clean_df
Number of rows	6311
Number of columns	41
_______________________	
Column type frequency:	
Date	1
numeric	40
_______________________	
Group variables	None

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
date	0	1	1999-01-	2023-05-	2011-02-	6311

skim_variable	n_missing	complete_rate	min	max	median	n_unique
			04	26	07	

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
australian_dollar	62	0.99	1.58	0.15	1.16	1.48	1.60	1.67	2.07
bulgarian_lev	460	0.93	1.95	0.00	1.94	1.96	1.96	1.96	1.96
brazilian_real	329	0.95	3.45	1.23	1.56	2.57	3.13	4.03	6.96
canadian_dollar	62	0.99	1.47	0.10	1.21	1.40	1.46	1.54	1.81
swiss_franc	62	0.99	1.33	0.22	0.94	1.10	1.28	1.54	1.68
chinese_yuan_renminbi	329	0.95	8.48	1.17	6.56	7.58	8.08	9.50	11.28
cypriot_pound	4007	0.37	0.58	0.00	0.57	0.57	0.58	0.58	0.59
czech_koruna	62	0.99	28.07	3.55	22.97	25.54	27.02	30.17	38.58
danish_krone	62	0.99	7.45	0.01	7.42	7.44	7.45	7.46	7.47
estonian_kroon	3237	0.49	15.65	0.00	15.65	15.65	15.65	15.65	15.65
uk_pound_sterling	62	0.99	0.78	0.10	0.57	0.68	0.80	0.87	0.98
greek_drachma	5797	0.08	331.15	6.07	320.78	325.41	330.52	336.80	340.75
hong_kong_dollar	62	0.99	9.27	1.22	6.44	8.48	9.22	10.18	12.47
croatian_kuna	431	0.93	7.47	0.13	7.10	7.38	7.48	7.56	7.77
hungarian_forint	62	0.99	290.67	41.41	228.16	254.63	280.84	312.69	430.65
indonesian_rupiah	62	0.99	13055.43	2826.79	6707.81	11352.51	13034.32	15531.39	18239.61
israeli_shekel	330	0.95	4.63	0.69	3.25	4.00	4.71	5.24	5.95
indian_rupee	329	0.95	66.69	13.87	38.50	56.15	67.88	78.74	92.06
iceland_krona	2407	0.62	108.01	34.29	68.07	83.70	89.36	138.10	305.00
japanese_yen	62	0.99	127.89	15.51	89.30	117.32	128.94	136.33	169.75

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
korean_won	62	0.99	1350.51	160.44	938.67	1250.31	1334.06	1429.97	1993.95
lithuanian_litas	2214	0.65	3.53	0.21	3.30	3.45	3.45	3.45	4.72
latvian_lats	2469	0.61	0.66	0.05	0.52	0.63	0.70	0.70	0.71
maltese_lira	4007	0.37	0.42	0.01	0.39	0.41	0.43	0.43	0.44
mexican_peso	62	0.99	16.69	4.69	7.62	13.70	16.90	20.49	27.09
malaysian_ringgit	62	0.99	4.42	0.46	3.16	4.11	4.57	4.76	5.19
norwegian_krone	62	0.99	8.66	0.98	7.22	7.95	8.26	9.43	12.32
new_zealand_dollar	62	0.99	1.81	0.20	1.39	1.66	1.76	1.96	2.55
philippine_peso	62	0.99	57.44	7.81	36.84	53.09	58.34	62.46	76.76
polish_zloty	62	0.99	4.17	0.31	3.21	3.98	4.19	4.36	4.95
romanian_leu	62	0.99	3.97	0.88	1.29	3.55	4.28	4.56	4.98
russian_rouble	379	0.94	48.24	19.63	23.19	34.47	40.19	68.91	117.20
swedish_krona	62	0.99	9.51	0.70	8.05	9.07	9.31	10.11	11.71
singapore_dollar	62	0.99	1.74	0.22	1.38	1.57	1.66	1.97	2.23
slovenian_tolar	4262	0.32	224.63	16.10	187.13	213.49	230.30	239.51	240.03
slovak_koruna	3751	0.41	39.51	4.15	30.13	37.47	40.91	42.77	47.48
thai_baht	62	0.99	41.59	4.75	33.20	37.89	40.24	45.07	53.54
turkish_lira	62	0.99	3.91	4.31	0.37	1.73	2.21	3.95	21.58
us_dollar	62	0.99	1.19	0.16	0.83	1.09	1.18	1.31	1.60
south_african_rand	62	0.99	12.01	3.87	6.08	8.78	11.16	15.46	21.01

4. Data analysis

As the dataset is clean now as shown in the previous step, we can proceed ahead with the analysis part. Since there are many countries in the dataset, let's focus only on India now and check how the value of Indian Rupee changed against Euro over time. To do this, first we will create a new dataframe which would give us the summary of the data pertaining to India only using pipes.

```
exchange_india_summary <- exchange_clean_df %>%
  summarise(min_india = round(min(indian_rupee, na.rm = T),2), max_india =
            round(max(indian_rupee, na.rm = T),2),
            mean_india = round(mean(indian_rupee, na.rm = T),2), std_india =
            round(sd(indian_rupee, na.rm = T),2), min_year = min(date),
            max_year = max(date))
knitr::kable(exchange_india_summary)
```

min_india	max_india	mean_india	std_india	min_year	max_year
38.5	92.06	66.69	13.87	1999-01-04	2023-05-26

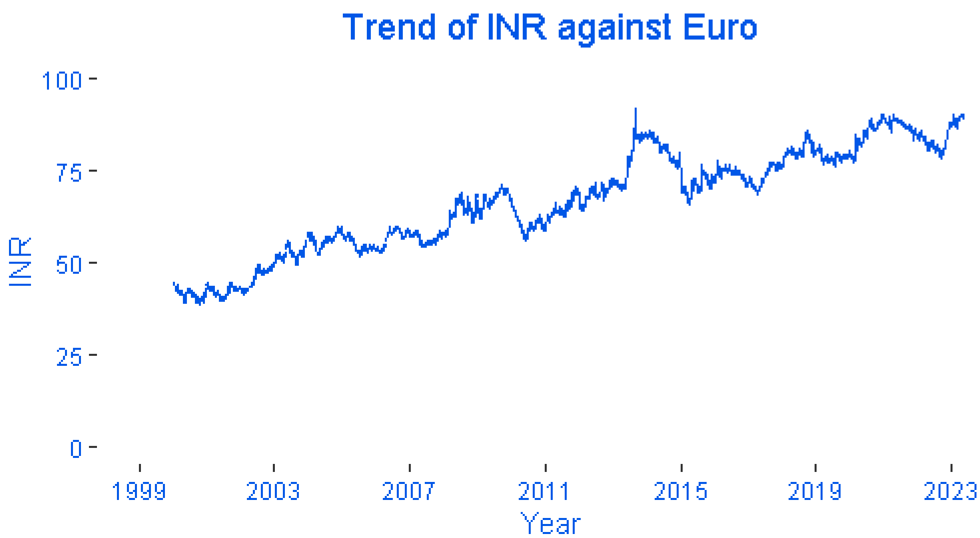
a) Checking for date at which the Indian Rupee touched the maximum against Euro

```
max_exchange_date <- exchange_clean_df %>%
  filter(indian_rupee == exchange_india_summary$max_india) %>%
  select(date) %>%
  pull()
```

b) Now, lets plot everything and see how the trend of INR against Euro appears over time.

In this section, the plot is divided into parts to follow easily without any confusion. With p1_india, we get a plot with labels. In p2_india, we add themes to the p1_india. In p3_india, we adjust the scales of both x and y-axis. Here, on the x-axis, we convert the dates to date class as we don't want the scale function to treat date either as a character or numeric datatype. The quality of the output plot might not be good in the rmd document as it is saved in a html format. The quality of the plot can be improved by adjusting the dpi in ggsave.

```
p1_india <- ggplot(exchange_clean_df, aes(x = date, y = indian_rupee)) +
geom_line(color = "#0057e7") + labs(title = "Trend of INR against Euro", x =
"Year", y = "INR")

p2_india <- p1_india + theme(plot.title = element_text(hjust = 0.5, color =
"#0057e7"), panel.grid =  element_blank(), panel.background =
element_blank(), axis.text.x = element_text(color = "#0057e7"), axis.text.y =
element_text(color = "#0057e7"), axis.title = element_text(color =
"#0057e7"))

p3_india <- p2_india + scale_y_continuous(limits = c(0, 100)) +
scale_x_date(breaks = seq(from = as.Date("1999-01-04"), to = as.Date("2023-
05-26"), by = "4 years"), date_labels = "%Y")

print(p3_india)
```



c) Lets investigate quarterly trends of INR against Euro

The main purpose of this part is to check if the fluctuation in the value of INR against Euro is seasonal. First, a new dataframe is created which would divide the dates in our cleaned dataset to quarters by using quarter function. By default the division is in 4 parts, i.e., Jan-Mar, Apr-Jun, Jul-Sep, and Oct-Dec.

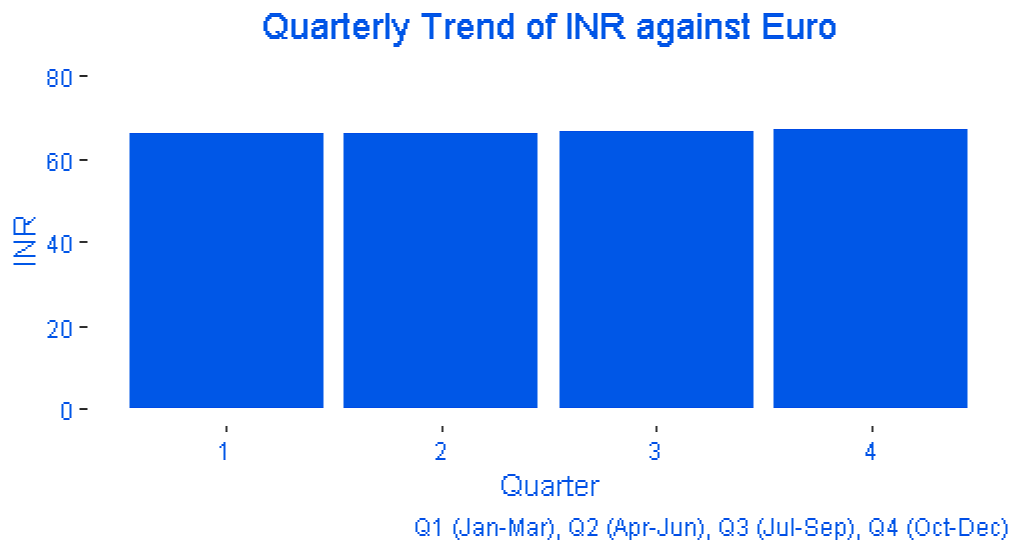
```
exchange_quarter_df <- exchange_clean_df %>%
  select(-date) %>%
  mutate(date_quarter = quarter(exchange_clean_df$date)) %>%
  group_by(date_quarter) %>%
  summarise(indian_rupee_quarter = round(mean(indian_rupee, na.rm = T),2))
knitr::kable(exchange_quarter_df)
```

date_quarter	indian_rupee_quarter
1	66.43
2	66.25
3	66.78
4	67.32

d) Plotting quarterly trends of INR against Euro

```
p1_quarter <- ggplot(exchange_quarter_df, aes(x = date_quarter, y =
exchange_quarter_df$indian_rupee_quarter)) + geom_col(fill = "#0057e7")

p2_quarter <- p1_quarter + labs(title = "Quarterly Trend of INR against
Euro", x = "Quarter", y = "INR", caption = "Q1 (Jan-Mar), Q2 (Apr-Jun), Q3
(Jul-Sep), Q4 (Oct-Dec)")

p3_quarter <- p2_quarter + theme(plot.title = element_text(hjust = 0.5, color
= "#0057e7"), axis.title = element_text(color = "#0057e7"), axis.text.x =
element_text(color = "#0057e7"), axis.text.y = element_text(color =
```

```
"#0057e7"), panel.background = element_blank(), panel.grid = element_blank(),
plot.caption = element_text(color = "#0057e7"))

p4_quarter <- p3_quarter + scale_y_continuous(limit = c(0,80))

print(p4_quarter)
```



e) Identification of correlation between INR and currencies of some other major countries which include USA, Canada, China and Japan

The correlation b/w INR and other currencies is to find out how the fluctuation in INR effects the exchange rates of other countries. If the correlation is positive, this means that when the value of INR increase, the value of other currency also increase. When the correlation is negative, it means that when the value of INR decrease, the value of other currency also decrease. Either way, if the value of correlation is over 0.9, it can be considered strong otherwise weak.

```
correlation_value <- c(round(cor(exchange_clean_df$indian_rupee,
exchange_clean_df$us_dollar, use = "complete.obs"), 2),
                       round(cor(exchange_clean_df$indian_rupee,
exchange_clean_df$chinese_yuan_renminbi, use = "complete.obs"), 2),
                       round(cor(exchange_clean_df$indian_rupee,
exchange_clean_df$canadian_dollar, use = "complete.obs"), 2),
                       round(cor(exchange_clean_df$indian_rupee,
exchange_clean_df$japanese_yen, use = "complete.obs"), 2))
correlation_countries <- c("USA", "China", "Canada", "Japan")
correlation_with_india_df <- data.frame(correlation_countries,
correlation_value)

knitr::kable(correlation_with_india_df)
```

correlation_countries	correlation_value
USA	0.25
China	-0.36
Canada	0.07
Japan	0.21

5. Application of Machine Learning (ML) models

Since from the first plot, i.e., trend of INR against Euro, we can clearly see that with date the value of INR is increasing. This relationship is linear, and we can use this information to create a model, and with this model, we can predict the value of INR in the future. Selection of a suitable ML models is important to perform this task. So, we can proceed ahead with a liner regression model which also aligns with the trend of INR.

Since, we are working with 2 variables, i.e., date and indian_rupee, lets create a new dataframe containing only these 2 variables. Here, as the value of INR is changing over time, we can treat it as dependent variable and date as independent variable. We need to omit na values as the model shouldn't contain them to run.

```
exchange_model <- exchange_clean_df %>%
  select(date, indian_rupee) %>%
  na.omit()
```

Now, lets create a linear regression model and check the summary on how it looks.

```
lr_model <- lm(indian_rupee ~ date, data = exchange_model)
summary(lr_model)

##
## Call:
## lm(formula = indian_rupee ~ date, data = exchange_model)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.9248 -3.0698 -0.5581  2.5033 21.5998
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.472e+01  3.556e-01  -41.39   <2e-16 ***
## date         5.342e-03  2.303e-05  231.93   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.389 on 5980 degrees of freedom
## Multiple R-squared:    0.9,  Adjusted R-squared:  0.8999
## F-statistic: 5.379e+04 on 1 and 5980 DF,  p-value: < 2.2e-16
```

From the above summary, there are few important areas to focus, i.e., value of R-square, value of INR estimate against date, and p-value. Lets break these things down: High-value of R square in our case tells us how well the model was able to capture ups and downs in the INR value over time. p-value should be less than 0.05 for a strong relationship between the two variables in our dataframe, and in our case, the smaller p-value suggests that the change in INR is strongly related to date. The value of INR estimate against date is the value of INR change for every step of increase in the date.

Based on all these results, we can make a conclusion that our model is good and can be relied upon to predict the value of INR in the future. It is important to remember that the date is not the only factor which effects the change of INR. There could be some other factors, for e.g., Govt. policies, FDI's, Inflation, etc,. However, in the context of our dataset, we can rely on the date to predict the value of INR in the future. Now, lets focus on the date for prediction.

```
future_date <- as.Date("2050-01-01")
```

The next step is to feed this date to the model to predict the value of INR.

```
predict_inr <- predict(lr_model, newdata = data.frame(date = future_date))
print(predict_inr)

##        1
## 141.3819
```