# R\_CODERS

## R Analysis For Import Export

FOCUS IMPORT AND EXPORT ISSUES WITH R DEDICATED AFFECTS OF INLATION, CURRENCY AND FOREIGN EXCHANGE RATES

**Reason for Project Topic**

We started to hear import export amounts, foreign trade deficit, currency rates, inflation and many other financial terms more frequently in our daily life and financial analysis are getting more important day by day. There are a lot of trustable data sources for trading statistics, inflation and currency rates. We selected Turkey export/import analysis as Project topic because of that we want to understand the reasons behind economic status of Turkey and want to make an introduction to visualize and analyze financial data.

**Executive Summary**

* We investigated import/export amounts, currency rates, inflation rates and customer price index changes from 2010 to April 2018.
* We mainly looked changes in the export import amounts, main sectors for export and import and possible correlations between export/import amounts between currency rates and inflation.
* When export sub-sectors are investigated for last 3 years, we saw that top 3 export subsectors are manufacture of motor vehicles and trailers, manufacture of basic metals and manufacture of textiles.
* When import subsectors are investigated for last 3 years, we saw that top 3 import subsectors are manufacture of basic metals, manufacture of chemicals and chemical products, mining and quarrying.
* A significant decrease in import is seen between 2014-2016, when we investigated currency rates for the same period, we saw that dollar currency rate was increased dramatically which means that increases in currency rates have a decreasing impact over import amount.
* As an overall conclusion we can say that import amounts are always more than export amount which means there is a chronical foreign trade deficit problem, however foreign trade deficit is in a decreasing trend after 2013 except for 2017.

**Group Project - Initilization and Project Diagrams**

*25.11.2018*



Group Name : R\_Coders

Project Name : Import - Export Analysis

##Project Members

* Büşra Aydemir
* Leyla Yiğit
* Mehmet Ak
* Mercan Karacabey
* Merve Özen Şahin

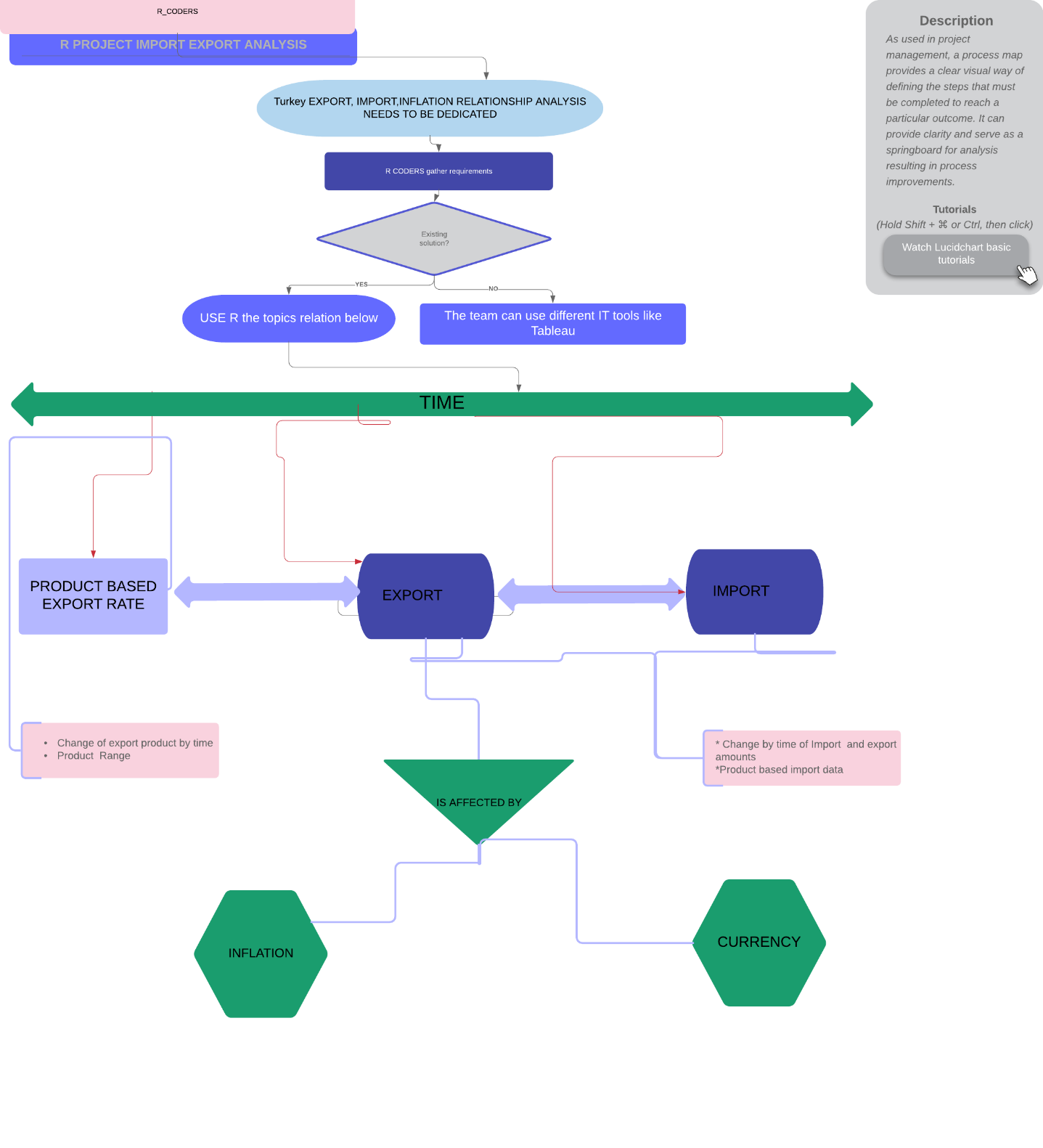
About Project

We will analyze the recent changes in inflation vs. data on exports and imports.We will analyze the factors which affecting export and import in 3 categories:

* Inflation
* Exchange Rates
* Interest Rates

Project Schema:

Project Management System Procurement Process



Details

We will examine export and import sector based. Some of the sectors which we will examine:

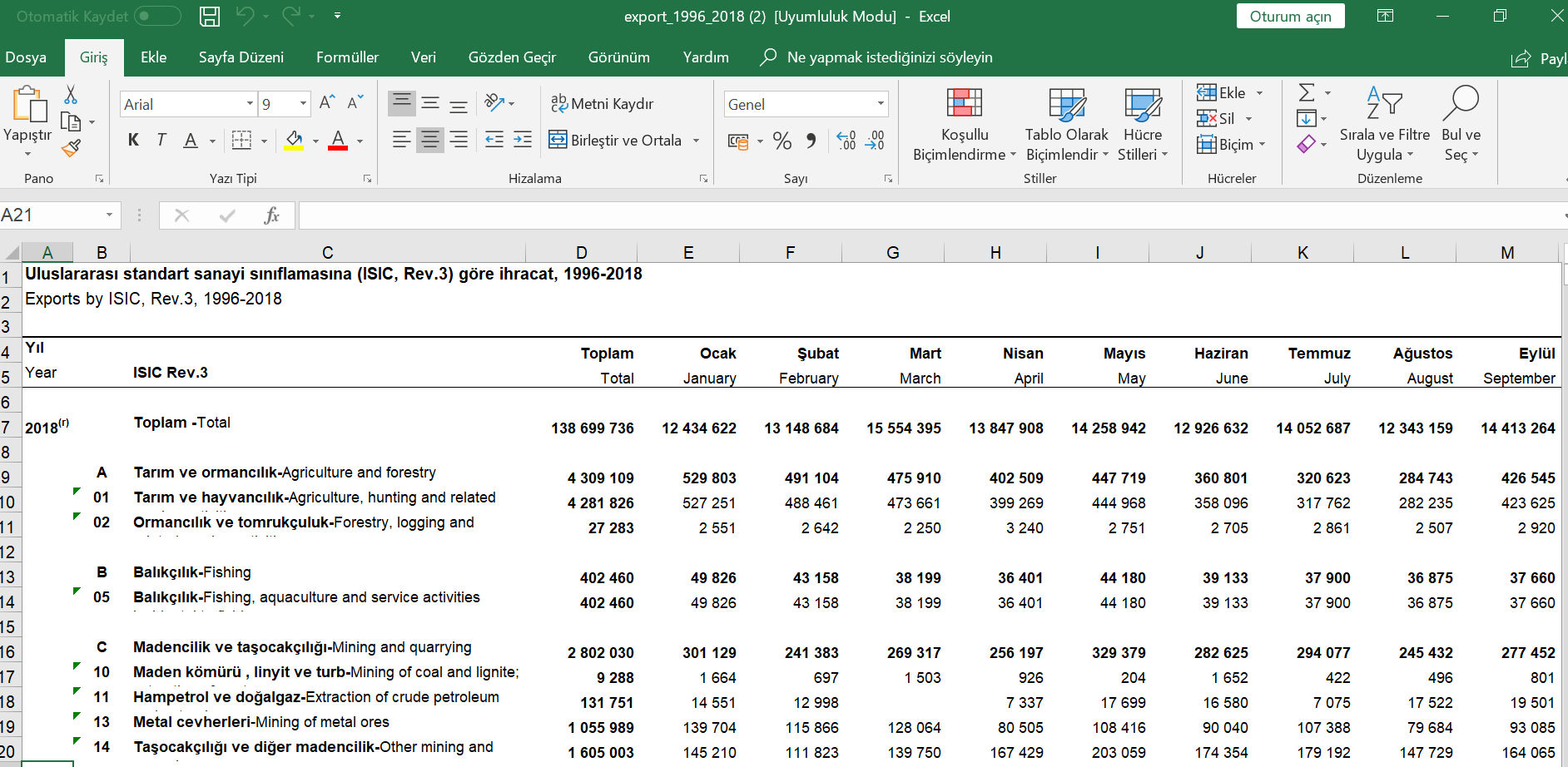
* Agriculture
* Production
* Textile
* Metal Industry
* Plastic

Where do we get data?

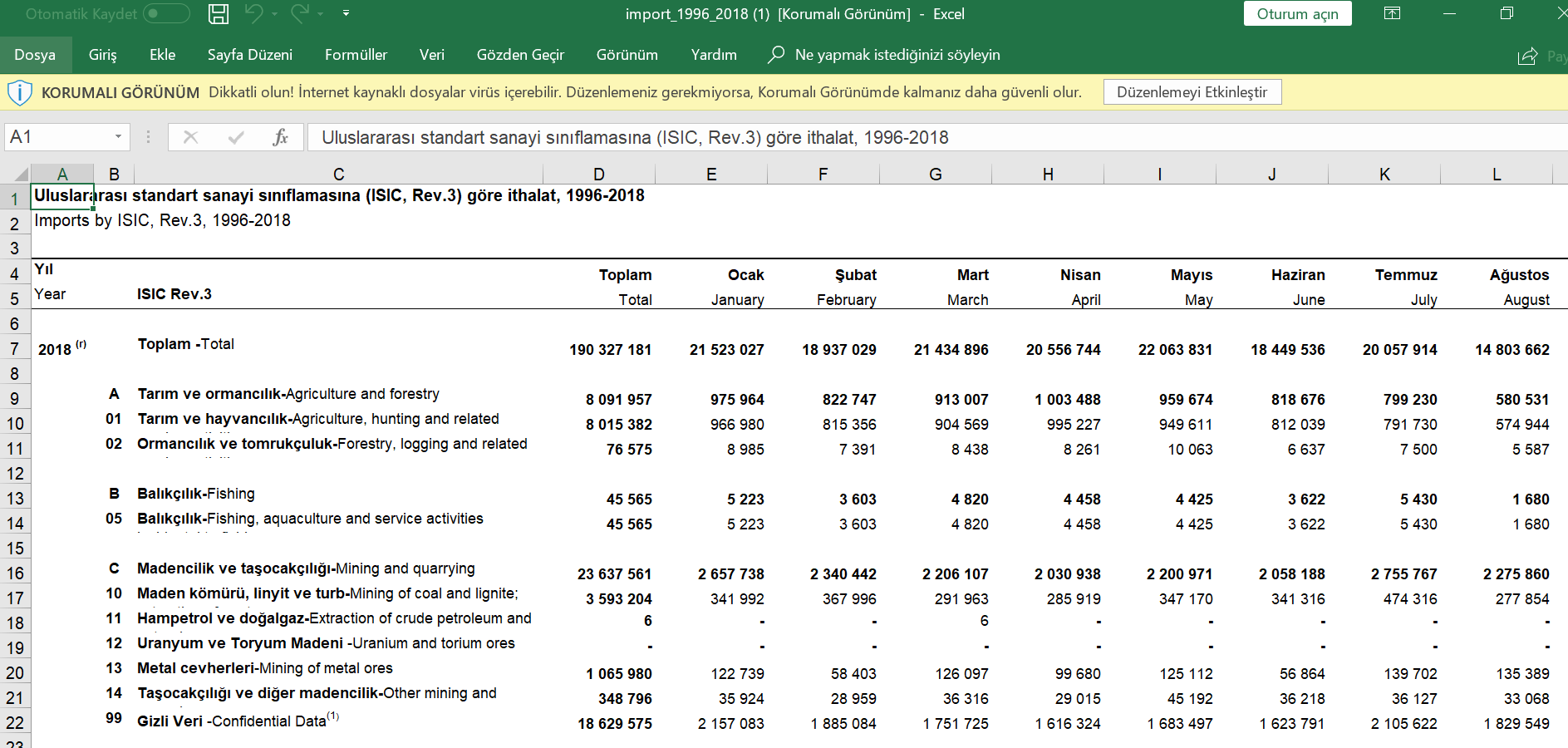
* For Import: [Import](http://www.oaib.gov.tr/)
* For Export: [Export](http://www.tim.org.tr/tr/ihracat-rakamlari.html)
* For Inflation: [Inflation](https://www.tcmb.gov.tr/wps/wcm/connect/TR/TCMB+TR/Main+Menu/Istatistikler/Enflasyon+Verileri/Tuketici+Fiyatlari)
* For Exchange Rate: [ExchangeRates](https://github.com/algopoly/EVDS)
* For Interest Rate: [InterestRate](https://evds2.tcmb.gov.tr/)

Import data uploaded Github. [Here](https://github.com/MEF-BDA503/gpj18-r_coders/tree/master/import_data)

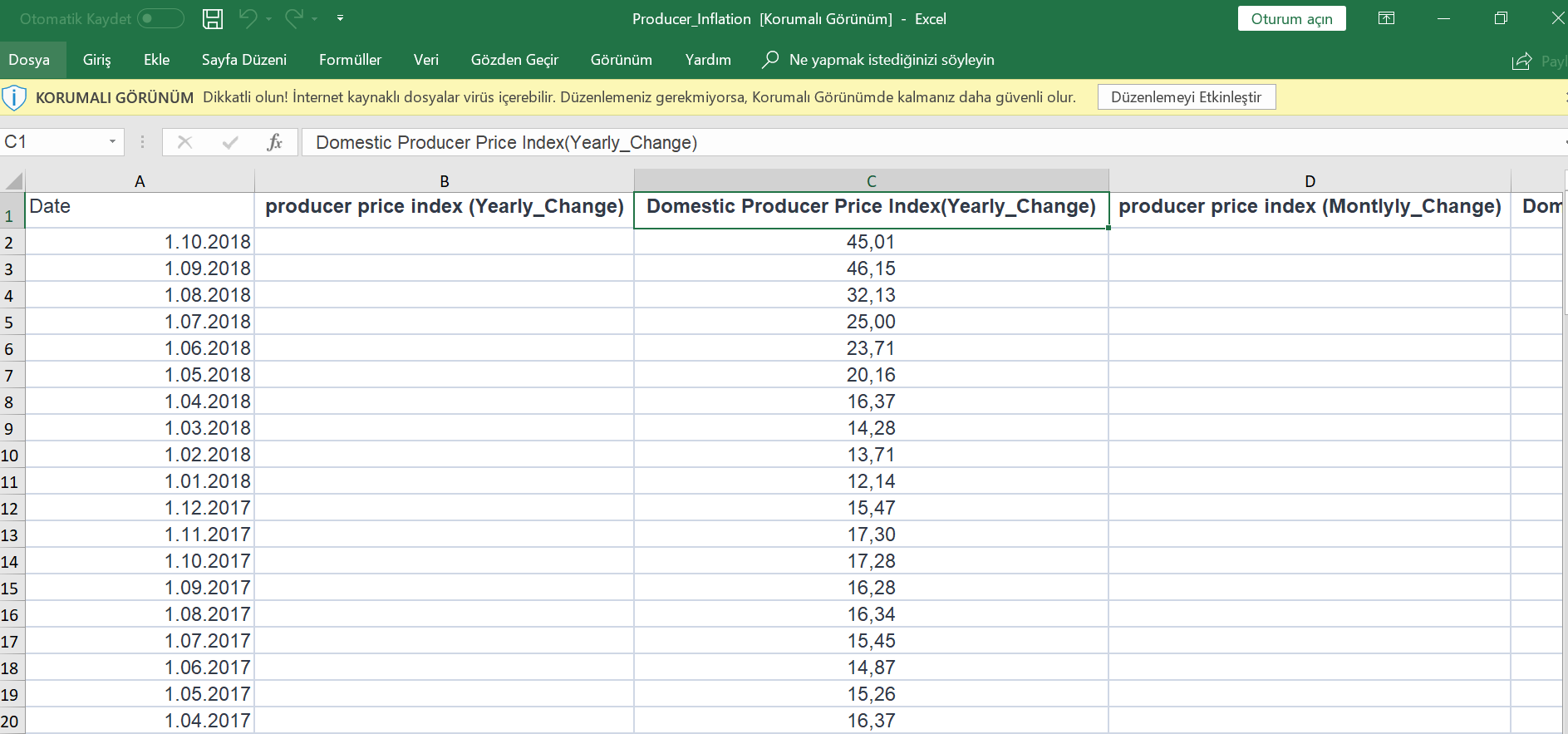
**Export Raw data structure**



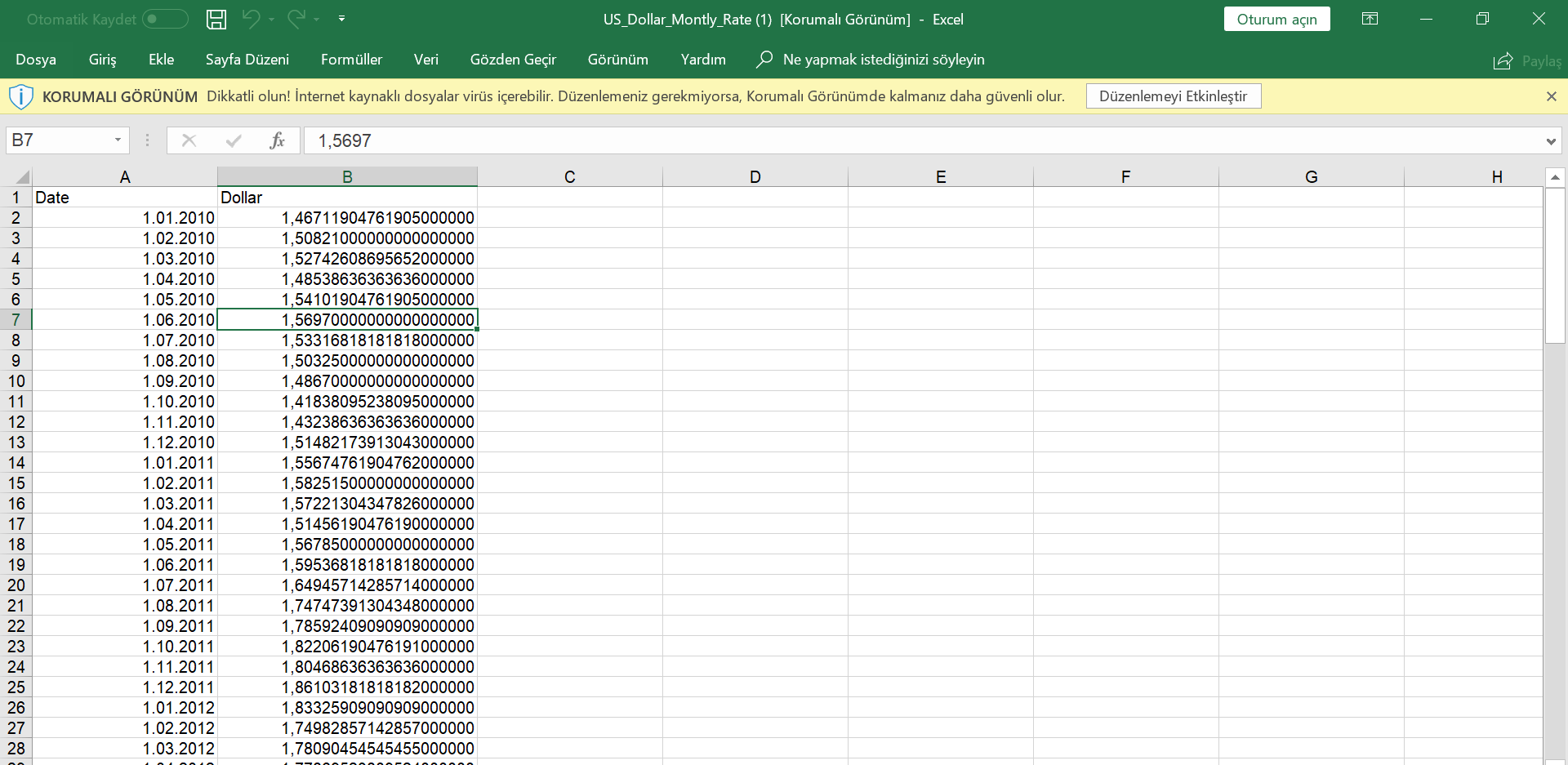
**Import Raw Data Structure**



**Producer Inflation Raw Data**



**USD Rate Raw Data**



**Data preparation**

* 1. **Data Identification**

The Data Identification stage is dedicated to identifying the datasets required for the analysis project and their sources. Our analysis examines the change between 2010 and 2018. Our main data used between 2010 - 2018 import and export results. In addition, inflation and currency figures are our main variables.

* 1. **Data Acquisition & Filtering**

*IMPORT Data Source:*

(" http://www.oaib.gov.tr/tr/default.html")

*EXPORT DATA SOURCES*

("http://www.tim.org.tr/tr/ihracat-rakamlari.html")

*INFLATION*

("https://www.tcmb.gov.tr/wps/wcm/connect/TR/TCMB+TR/Main+Menu/Istatistikler/Enflasyon+Verileri/Tuketici+Fiyatlari")

*EXCHANGE RATE*

("https://github.com/algopoly/EVDS")

*INTEREST RATE*

("https://evds2.tcmb.gov.tr/")

* 1. **Data Extraction**

Download export and import raw excel files and put them in temp files and read them. And remove the temp files after reading. We use “readxl” library for read data.

First create a temporary file, download file from repository to the temp file and read excel files.

* 1. **Data Validation & Cleansing**

Invalid data can skew and falsify analysis results. The Data Validation and Cleansing stage

is dedicated to establishing often complex validation rules and removing any known invalid data.

We removed the NA rows and change the column names of data. Also we don’t use total amount, because of this removed the total amount columns. And convert char data types to numeric data types for all columns except sector type code and sector name columns.

We use “plyr”, “tidyverse”, “stringr” packages for data validation and cleansing stage.

* 1. **Data Aggregation & Representation**

Data may be spread across multiple datasets, requiring that datasets be joined together via common fields, for example date or ID. In our cases, the same data fields appear in multiple datasets, such as sector type code and sector name. Either way, a method of data reconciliation is required or the dataset representing the correct value needs to be determined.

Data analysis

Explotary Analysis

*R\_Coders*

*December 20, 2018*

**Libraries**

library("tidyverse")

The **tidyverse** is an opinionated collection of **R packages** designed for data science. All **packages**share an underlying design philosophy, grammar, and data structures.

library("readxl")

The **readxl package** makes it easy to get data out of Excel and into **R**. Compared to many of the existing **packages** (e.g. gdata, xlsx, xlsReadWrite) **readxl** has no external dependencies, so it's easy to install and use on all operating systems.

library("ggplot2")

ggplot2 is a system for declaratively creating graphics, based on [The Grammar of Graphics](http://amzn.to/2ef1eWp).

library("plotly")

**Plotly's R** graphing **library** makes interactive, publication-quality graphs online

#library("xlsx")

An **R package** to read, write, format Excel 2007 and Excel 97/2000/XP/2003 files

library("sqldf")

[sqldf](https://cran.r-project.org/package=sqldf) is an R package for runing [SQL statements](https://en.wikipedia.org/wiki/SQL) on R data frames, optimized for convenience.

library("dplyr")

**dplyr** is a powerful **R**-**package** to transform and summarize tabular data with rows and columns.

install.packages("tidyverse", repos = "https://cran.r-project.org")

## Installing package into 'C:/Users/ozenm/Documents/R/win-library/3.5'

## (as 'lib' is unspecified)

## also installing the dependency 'dplyr'

## package 'dplyr' successfully unpacked and MD5 sums checked

## package 'tidyverse' successfully unpacked and MD5 sums checked

##

## The downloaded binary packages are in

## C:\Users\ozenm\AppData\Local\Temp\RtmpkrTOIg\downloaded\_packages

install.packages("dplyr", repos = "https://cran.r-project.org")

## Installing package into 'C:/Users/ozenm/Documents/R/win-library/3.5'

## (as 'lib' is unspecified)

## package 'dplyr' successfully unpacked and MD5 sums checked

##

## The downloaded binary packages are in

## C:\Users\ozenm\AppData\Local\Temp\RtmpkrTOIg\downloaded\_packages

install.packages("readxl", repos = "http://cran.us.r-project.org")

## Installing package into 'C:/Users/ozenm/Documents/R/win-library/3.5'

## (as 'lib' is unspecified)

## package 'readxl' successfully unpacked and MD5 sums checked

##

## The downloaded binary packages are in

## C:\Users\ozenm\AppData\Local\Temp\RtmpkrTOIg\downloaded\_packages

install.packages("ggplot2", repos = "http://cran.us.r-project.org")

## Installing package into 'C:/Users/ozenm/Documents/R/win-library/3.5'

## (as 'lib' is unspecified)

## package 'ggplot2' successfully unpacked and MD5 sums checked

##

## The downloaded binary packages are in

## C:\Users\ozenm\AppData\Local\Temp\RtmpkrTOIg\downloaded\_packages

install.packages("plotly", repos = "http://cran.us.r-project.org")

## Installing package into 'C:/Users/ozenm/Documents/R/win-library/3.5'

## (as 'lib' is unspecified)

## package 'plotly' successfully unpacked and MD5 sums checked

##

## The downloaded binary packages are in

## C:\Users\ozenm\AppData\Local\Temp\RtmpkrTOIg\downloaded\_packages

install.packages("gapminder", repos = "http://cran.us.r-project.org")

## Installing package into 'C:/Users/ozenm/Documents/R/win-library/3.5'

## (as 'lib' is unspecified)

## package 'gapminder' successfully unpacked and MD5 sums checked

##

## The downloaded binary packages are in

## C:\Users\ozenm\AppData\Local\Temp\RtmpkrTOIg\downloaded\_packages

*#install.packages("xlsx", repos = "http://cran.us.r-project.org")*

install.packages("sqldf", repos = "http://cran.us.r-project.org")

## Installing package into 'C:/Users/ozenm/Documents/R/win-library/3.5'

## (as 'lib' is unspecified)

## package 'sqldf' successfully unpacked and MD5 sums checked

##

## The downloaded binary packages are in

## C:\Users\ozenm\AppData\Local\Temp\RtmpkrTOIg\downloaded\_packages

**library**("tidyverse")

## -- Attaching packages --------------------------------------------------- tidyverse 1.2.1 --

## <U+221A> ggplot2 3.1.0 <U+221A> purrr 0.2.5

## <U+221A> tibble 1.4.2 <U+221A> dplyr 0.7.8

## <U+221A> tidyr 0.8.2 <U+221A> stringr 1.3.1

## <U+221A> readr 1.3.0 <U+221A> forcats 0.3.0

## -- Conflicts ------------------------------------------------------ tidyverse\_conflicts() --

## x dplyr::filter() masks stats::filter()

## x dplyr::lag() masks stats::lag()

**library**("readxl")

**library**("ggplot2")

**library**("plotly")

##

## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':

##

## last\_plot

## The following object is masked from 'package:stats':

##

## filter

## The following object is masked from 'package:graphics':

##

## layout

**library**("gapminder")

*#library("xlsx")*

**library**("sqldf")

## Loading required package: gsubfn

## Loading required package: proto

## Loading required package: RSQLite

**library**("dplyr")

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/imp\_data\_final.rds?raw=true?raw=true",destfile=tmp,mode = 'wb')

imp\_data\_final<-read\_rds(tmp)

file.remove(tmp)

## [1] TRUE

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/exp\_data\_final.rds?raw=true?raw=true",destfile=tmp,mode = 'wb')

exp\_data\_final<-read\_rds(tmp)

file.remove(tmp)

## [1] TRUE

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/imp\_data.rds?raw=true?raw=true",destfile=tmp,mode = 'wb')

imp\_data<-read\_rds(tmp)

file.remove(tmp)

## [1] TRUE

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/exp\_data.rds?raw=true?raw=true",destfile=tmp,mode = 'wb')

exp\_data<-read\_rds(tmp)

file.remove(tmp)

## [1] TRUE

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/Producer\_Inflation.rds?raw=true?raw=true",destfile=tmp,mode = 'wb')

producer\_inf<-read\_rds(tmp)

file.remove(tmp)

## [1] TRUE

*# Create a temporary file*

tmp=tempfile(fileext=".xls")

*# Download file from repository to the temp file*

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Excel/export\_import\_sectors.xls?raw=true",destfile=tmp,mode='wb')

*# Read that excel file.*

sectors <- read\_excel(tmp)

## readxl works best with a newer version of the tibble package.

## You currently have tibble v1.4.2.

## Falling back to column name repair from tibble <= v1.4.2.

## Message displays once per session.

*# Remove the temp file*

file.remove(tmp)

## [1] TRUE

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/US\_Dollar\_Montly\_Rate.rds?raw=true?raw=true",destfile=tmp,mode = 'wb')

usd\_rate<-read\_rds(tmp)

file.remove(tmp)

## [1] TRUE

Format Data

names(exp\_data\_final)[names(exp\_data\_final) == 'Date'] <- 'Export\_Date'

names(exp\_data)[names(exp\_data) == 'Date'] <- 'Export\_Date'

names(imp\_data\_final)[names(imp\_data\_final) == 'Date'] <- 'Import\_Date'

names(imp\_data\_final)[names(imp\_data\_final) == 'Export\_Total\_Amount'] <- 'Import\_Total\_Amount' *#fix*

names(imp\_data)[names(imp\_data) == 'Date'] <- 'Import\_Date'

**library**("dplyr")

exp\_data <- inner\_join(exp\_data,sectors, by=c("Sector\_Type\_Code"="Sub\_Sector\_Type\_Code"))

imp\_data <- inner\_join(imp\_data,sectors, by=c("Sector\_Type\_Code"="Sub\_Sector\_Type\_Code"))

exp\_data$Export\_Year<-as.numeric(format(exp\_data$Export\_Date,"%Y"))

exp\_data$Export\_Year\_Month<-format(exp\_data$Export\_Date,"%Y-%m")

exp\_data\_final$Export\_Year<-as.numeric(format(exp\_data\_final$Export\_Date,"%Y"))

exp\_data\_final$Export\_Year\_Month<-format(exp\_data\_final$Export\_Date,"%Y-%m")

imp\_data$Import\_Year<-as.numeric(format(imp\_data$Import\_Date,"%Y"))

imp\_data$Import\_Year\_Month<-format(imp\_data$Import\_Date,"%Y-%m")

imp\_data\_final$Import\_Year<-as.numeric(format(imp\_data\_final$Import\_Date,"%Y"))

imp\_data\_final$Import\_Year\_Month<-format(imp\_data\_final$Import\_Date,"%Y-%m")

imp\_data<- imp\_data %>%

select (Import\_Date,Sector\_Type\_Code,Sector\_Type\_Code.y,Main\_Sector\_Flag,Sector\_Name\_Eng,Amount,Import\_Year,Import\_Year\_Month)

exp\_data<- exp\_data %>%

select (Export\_Date,Sector\_Type\_Code,Sector\_Type\_Code.y,Main\_Sector\_Flag,Sector\_Name\_Eng, Amount,Export\_Year,Export\_Year\_Month)

colnames(imp\_data)[colnames(imp\_data) == 'Amount'] <- 'Import\_Amount'

colnames(exp\_data)[colnames(exp\_data) == 'Amount'] <- 'Export\_Amount'

colnames(imp\_data)[colnames(imp\_data) == 'Sector\_Type\_Code'] <- 'Sub\_Sector\_Type\_Code'

colnames(exp\_data)[colnames(exp\_data) == 'Sector\_Type\_Code'] <- 'Sub\_Sector\_Type\_Code'

colnames(imp\_data)[colnames(imp\_data) == 'Sector\_Type\_Code.y'] <- 'Sector\_Type\_Code'

colnames(exp\_data)[colnames(exp\_data) == 'Sector\_Type\_Code.y'] <- 'Sector\_Type\_Code'

imp\_data$Import\_Amount[is.na(imp\_data$Import\_Amount)] <- 0

imp\_data\_final$Import\_Total\_Amount[is.na(imp\_data\_final$Import\_Total\_Amount)] <- 0

exp\_data$Export\_Amount[is.na(exp\_data$Export\_Amount)] <- 0

exp\_data\_final$Export\_Total\_Amount[is.na(exp\_data\_final$Export\_Total\_Amount)] <- 0

exp\_data\_final <- exp\_data\_final %>%

filter(Export\_Date<'2018-11-01')

exp\_data <- exp\_data %>%

filter(Export\_Date<'2018-11-01')

imp\_data\_final <- imp\_data\_final %>%

filter(Import\_Date<'2018-11-01')

imp\_data <- imp\_data %>%

filter(Import\_Date<'2018-11-01')

saveRDS(imp\_data,file="imp\_data\_v2.rds")

saveRDS(imp\_data\_final,file="imp\_data\_final\_v2.rds")

saveRDS(exp\_data,file="exp\_data\_v2.rds")

saveRDS(exp\_data\_final,file="exp\_data\_final\_v2.rds")

Review Import Data Structure

str(imp\_data)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 4236 obs. of 8 variables:

## $ Import\_Date : Date, format: "2018-01-01" "2018-01-01" ...

## $ Sub\_Sector\_Type\_Code: chr "A" "01" "02" "B" ...

## $ Sector\_Type\_Code : chr "A" "A" "A" "B" ...

## $ Main\_Sector\_Flag : num 1 0 0 1 0 1 0 0 0 0 ...

## $ Sector\_Name\_Eng : chr "Agriculture and forestry" "Agriculture, hunting and related service activities" "Forestry, logging and related service activities" "Fishing" ...

## $ Import\_Amount : num 975964 966980 8985 5223 5223 ...

## $ Import\_Year : num 2018 2018 2018 2018 2018 ...

## $ Import\_Year\_Month : chr "2018-01" "2018-01" "2018-01" "2018-01" ...

str(imp\_data\_final)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 94 obs. of 7 variables:

## $ Import\_Date : Date, format: "2010-07-01" "2010-08-01" ...

## $ Import\_Total\_Amount : num 26002098 24474224 24503919 26842444 26584903 ...

## $ Consumer\_Price\_Index\_Yearly\_Change : num 7.58 8.33 9.24 8.62 7.29 6.4 4.9 4.16 3.99 4.26 ...

## $ Consumer\_Price\_Index\_Monthly\_Change: num -0.48 0.4 1.23 1.83 0.03 -0.3 0.41 0.73 0.42 0.87 ...

## $ USD\_Rate : num 1.53 1.5 1.49 1.42 1.43 ...

## $ Import\_Year : num 2010 2010 2010 2010 2010 ...

## $ Import\_Year\_Month : chr "2010-07" "2010-08" "2010-09" "2010-10" ...

Review Export Data Structure

str(exp\_data)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 4452 obs. of 8 variables:

## $ Export\_Date : Date, format: "2018-01-01" "2018-01-01" ...

## $ Sub\_Sector\_Type\_Code: chr "A" "01" "02" "B" ...

## $ Sector\_Type\_Code : chr "A" "A" "A" "B" ...

## $ Main\_Sector\_Flag : num 1 0 0 1 0 1 0 0 0 0 ...

## $ Sector\_Name\_Eng : chr "Agriculture and forestry" "Agriculture, hunting and related service activities" "Forestry, logging and related service activities" "Fishing" ...

## $ Export\_Amount : num 529803 527251 2551 49826 49826 ...

## $ Export\_Year : num 2018 2018 2018 2018 2018 ...

## $ Export\_Year\_Month : chr "2018-01" "2018-01" "2018-01" "2018-01" ...

str(exp\_data\_final)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 94 obs. of 7 variables:

## $ Export\_Date : Date, format: "2010-07-01" "2010-08-01" ...

## $ Export\_Total\_Amount : num 19129365 17046904 17818461 21927173 18764739 ...

## $ Consumer\_Price\_Index\_Yearly\_Change : num 7.58 8.33 9.24 8.62 7.29 6.4 4.9 4.16 3.99 4.26 ...

## $ Consumer\_Price\_Index\_Monthly\_Change: num -0.48 0.4 1.23 1.83 0.03 -0.3 0.41 0.73 0.42 0.87 ...

## $ USD\_Rate : num 1.53 1.5 1.49 1.42 1.43 ...

## $ Export\_Year : num 2010 2010 2010 2010 2010 ...

## $ Export\_Year\_Month : chr "2010-07" "2010-08" "2010-09" "2010-10" ...

Prepare Data for Import&Export Line Graph

imp\_and\_exp\_data <- inner\_join(exp\_data, imp\_data, by=c("Export\_Date" = "Import\_Date","Sub\_Sector\_Type\_Code"="Sub\_Sector\_Type\_Code"))

imp\_and\_exp\_data\_bymonth <- aggregate(cbind(Import\_Amount, Export\_Amount) ~ Export\_Date, data = imp\_and\_exp\_data, sum)

imp\_and\_exp\_data\_bymonth <- gather(imp\_and\_exp\_data\_bymonth,

value = "value",

key = "type",

Export\_Amount, Import\_Amount)

*#Rename column names*

colnames(imp\_and\_exp\_data\_bymonth) <- c("Date","Type","Amount")

*#Remove Empty Dates*

imp\_and\_exp\_data\_bymonth <- imp\_and\_exp\_data\_bymonth %>%

filter(Date<'2018-11-01')

01\_export\_and\_import\_amount

p<-ggplot(imp\_and\_exp\_data\_bymonth,

aes(x=Date,

y=Amount/1000,

color=Type)) +

geom\_line()+

scale\_size\_area("Nitrogen") +

xlab("Import/Export Date") +

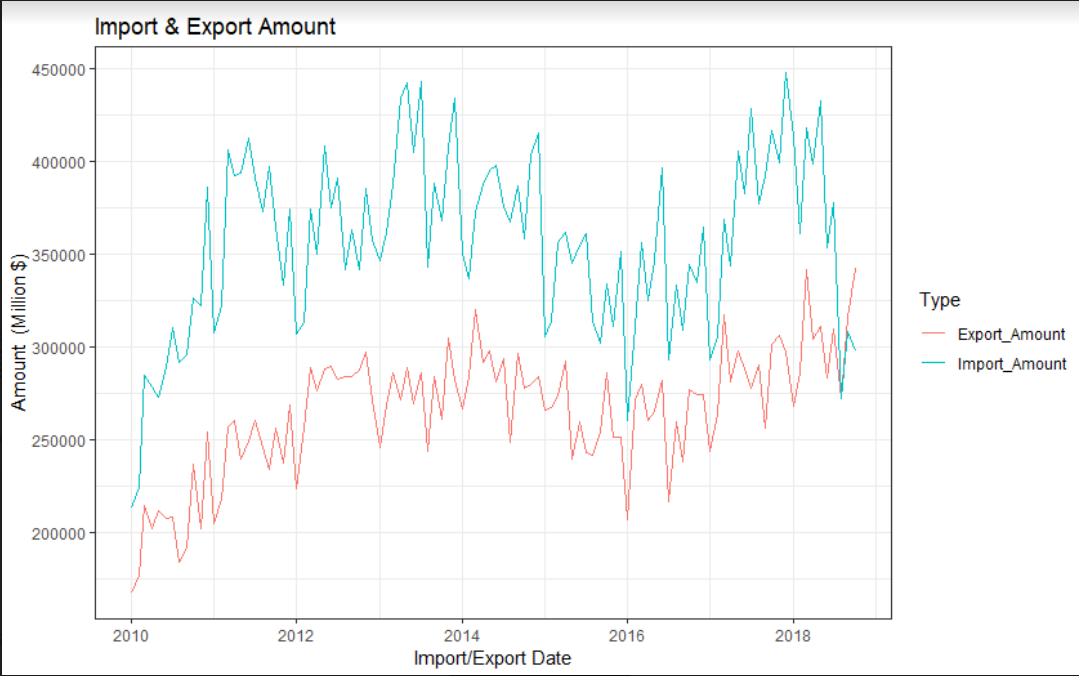
ylab("Amount (Million $)") +

ggtitle("Import & Export Amount")

style(p, text = row.names(imp\_and\_exp\_data\_bymonth))

20102012201420162018200003000040000

Export\_AmountImport\_AmountImport & Export AmountImport/Export DateAmount (Million $)Type



The graph shows the year-based comparison of export and import amount data. A very long period of increases and decreases are moving synchronously. In April 2018, the import amount Line has a major decline, while export amount Line has a major increase.

02\_Export\_Amount\_Based\_on\_Consumer\_Price\_Index\_And\_USD\_Rate

**library**(ggplot2)

**library**(plotly)

**library**(gapminder)

p <- exp\_data\_final %>%

ggplot(aes(USD\_Rate, Export\_Total\_Amount, size = Consumer\_Price\_Index\_Yearly\_Change, color=Export\_Year)) +

geom\_point() +

scale\_x\_log10() +

theme\_bw()+

scale\_size\_area("Nitrogen") +

xlab("USD Rate") +

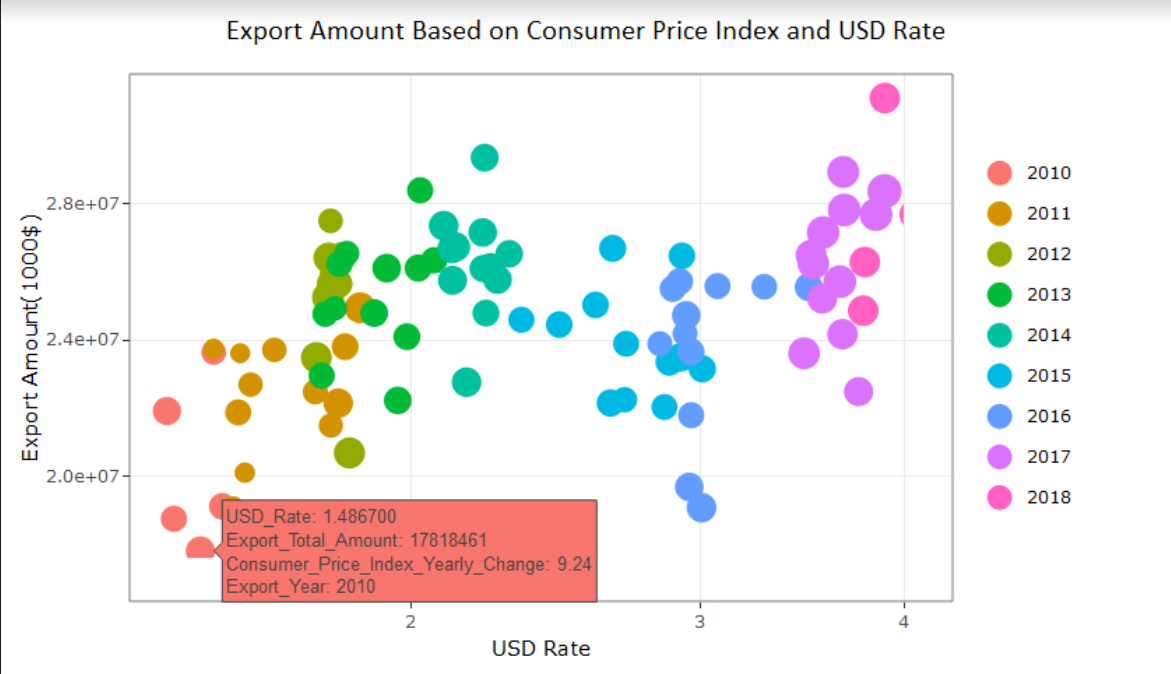
ylab("Export Amount(1000$)") +

ggtitle("Export Amounts and Consumer Price Index")

ggplotly(p)

2342.0e+072.4e+072.8e+07

Export Amounts and Consumer Price Index20102012201420162018Export\_YearUSD RateExport Amount(1000$)



The graph shows export amount based on consumer price index and USD rate on monthly basis. Each year is indicated in different colors and every point shows monthly variables values.

03\_Import\_Amount\_Based\_on\_Consumer\_Price\_Index\_And\_USD\_Rate

p <- imp\_data\_final %>%

ggplot(aes(USD\_Rate, Import\_Total\_Amount, size = Consumer\_Price\_Index\_Yearly\_Change, color=Import\_Year)) +

geom\_point() +

scale\_x\_log10() +

theme\_bw()+

scale\_size\_area("Nitrogen") +

xlab("USD Rate") +

ylab("Import Amount(1000$)") +

ggtitle("Import Amounts and Consumer Price Index")

ggplotly(p)

2342.5e+073.0e+073.5e+074.0e+074.5e+07

Import Amounts and Consumer Price Index20102012201420162018Import\_YearUSD RateImport Amount(1000$)

top\_import\_by\_sector <-

imp\_data %>%

filter(Main\_Sector\_Flag==1)%>% *#& Import\_Year==2018) %>%*

group\_by(Import\_Year,Sector\_Type\_Code,Sector\_Name\_Eng) %>%

summarise(Import\_Total\_Amount=sum(Import\_Amount))%>%

arrange(desc(Import\_Total\_Amount))

top\_export\_by\_sector <-

exp\_data %>%

filter(Main\_Sector\_Flag==1)%>% *#& Export\_Year==2018) %>%*

group\_by(Export\_Year,Sector\_Type\_Code,Sector\_Name\_Eng) %>%

summarise(Export\_Total\_Amount=sum(Export\_Amount))%>%

arrange(desc(Export\_Total\_Amount))

trade\_deficit\_by\_sectors<-sqldf('select Export\_Year as Year, a.Sector\_Type\_Code,

a.Sector\_Name\_Eng, Export\_Total\_Amount, Import\_Total\_Amount,

(Import\_Total\_Amount - Export\_Total\_Amount) as Trade\_Deficit\_Amount

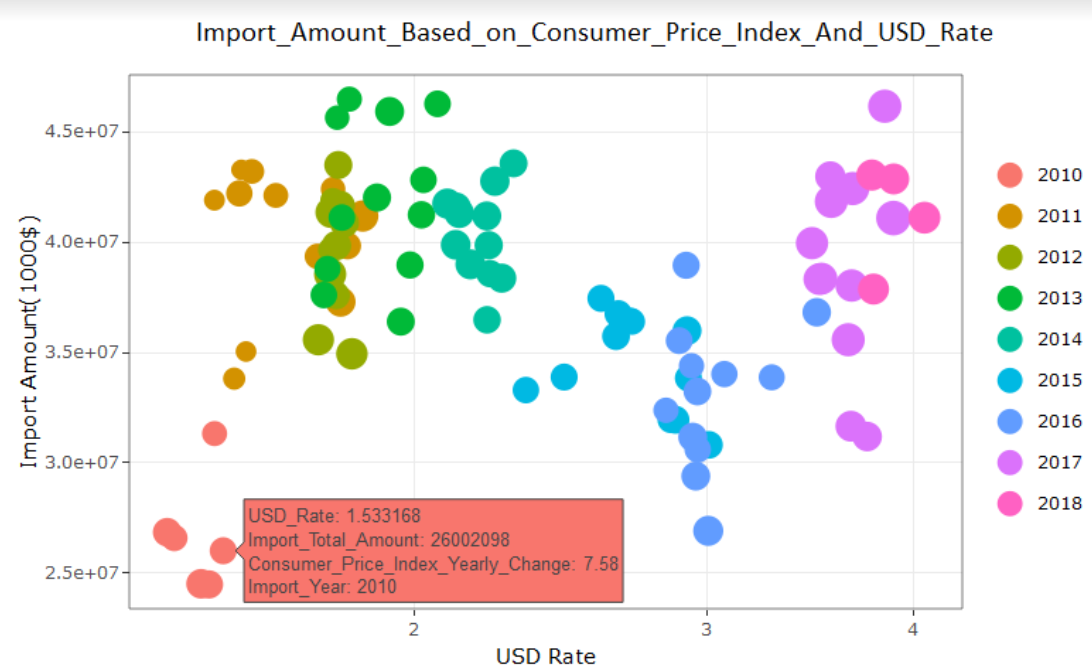
from top\_export\_by\_sector a left join top\_import\_by\_sector b

on Export\_Year = Import\_Year

and a.Sector\_Type\_Code = b.Sector\_Type\_Code')

trade\_deficit\_by\_sectors$Import\_Total\_Amount[is.na(trade\_deficit\_by\_sectors$Import\_Total\_Amount)] <- 0

trade\_deficit\_by\_sectors$Trade\_Deficit\_Amount[is.na(trade\_deficit\_by\_sectors$Trade\_Deficit\_Amount)] <- 0



The graph shows import amount based on consumer price index and USD rate on monthly basis. Each year is indicated in different colors and every point shows monthly variables values.

04\_yearly\_trade\_deficit\_amount

trade\_deficit\_by\_sectors %>%

*#filter(Main\_Sector\_Flag==1)%>%*

group\_by(Year,Sector\_Name\_Eng) %>%

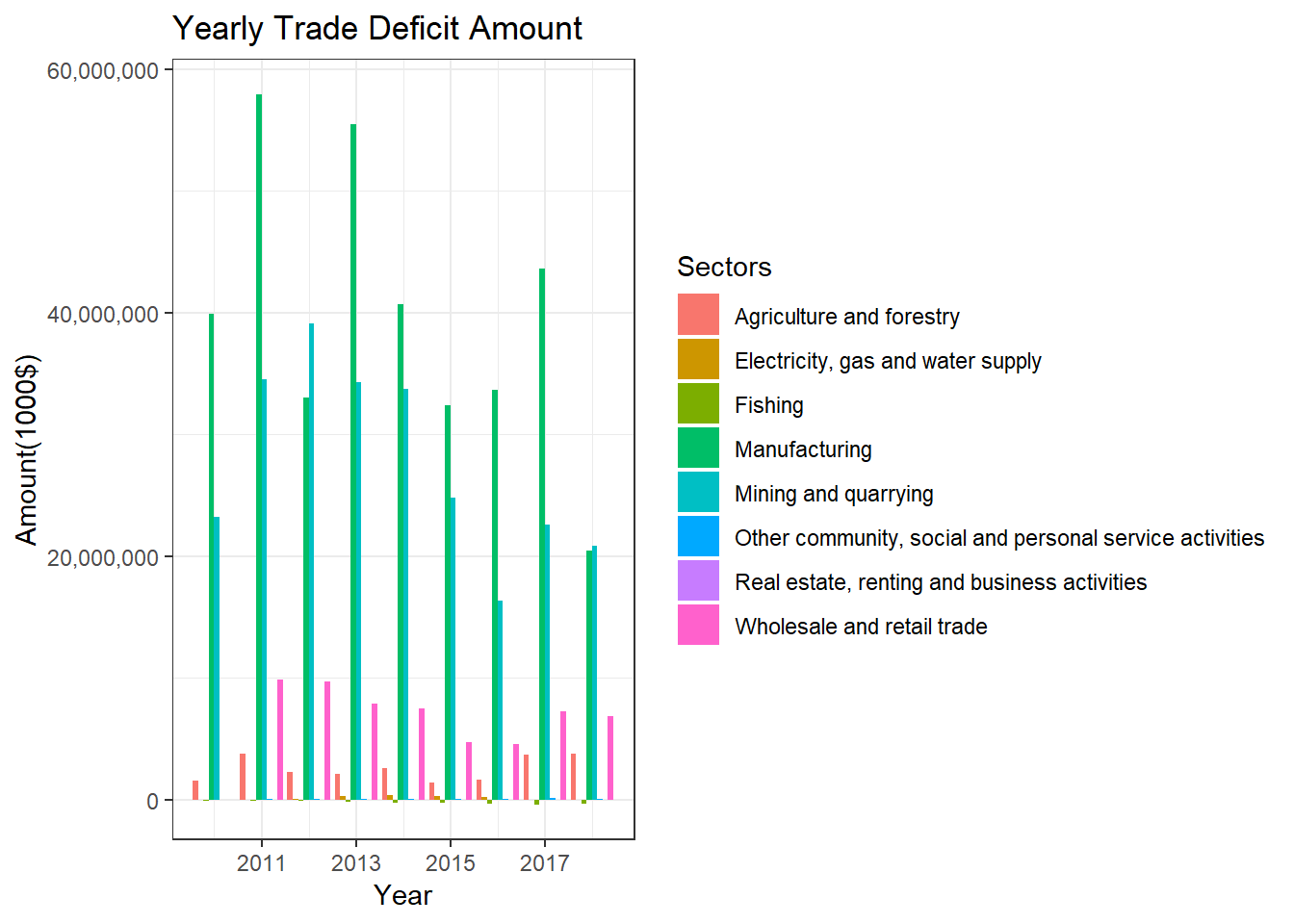
summarise(Yearly\_Total\_Trade\_Deficit\_Amount=sum(Trade\_Deficit\_Amount))%>%

ggplot(data = ., aes(x = Year, y = Yearly\_Total\_Trade\_Deficit\_Amount,

fill = Sector\_Name\_Eng)) + geom\_bar(stat = "identity", position=position\_dodge()) + aes(x = Year, y = Yearly\_Total\_Trade\_Deficit\_Amount) + labs(x = "", y = "", title = "Yearly Trade Deficit Amount") + theme\_bw() + scale\_y\_continuous(labels = scales::comma) + guides(fill=guide\_legend(title="Sectors"))+

xlab("Year") +

ylab("Amount(1000$)")



Yearly trade deficit amount graph was prepared based on the sector. In general, there is a constant decline. Although a clear result can not be seen due to the incomplete year for 2018, we can see that the year will be closed down again for two months due to the close of the year.

Prepare Data for Yearly Average Export Amount and Other Factors

**library**("dplyr")

exp\_data\_total\_amount\_by\_year <-

exp\_data %>%

group\_by(Export\_Date,Export\_Year) %>%

summarise(Yearly\_Export\_Total\_Amount=sum(Export\_Amount))

exp\_data\_amount\_and\_others <- inner\_join(exp\_data\_final,exp\_data\_total\_amount\_by\_year, by=c("Export\_Date" = "Export\_Date"))

str(exp\_data\_amount\_and\_others)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 94 obs. of 9 variables:

## $ Export\_Date : Date, format: "2010-07-01" "2010-08-01" ...

## $ Export\_Total\_Amount : num 19129365 17046904 17818461 21927173 18764739 ...

## $ Consumer\_Price\_Index\_Yearly\_Change : num 7.58 8.33 9.24 8.62 7.29 6.4 4.9 4.16 3.99 4.26 ...

## $ Consumer\_Price\_Index\_Monthly\_Change: num -0.48 0.4 1.23 1.83 0.03 -0.3 0.41 0.73 0.42 0.87 ...

## $ USD\_Rate : num 1.53 1.5 1.49 1.42 1.43 ...

## $ Export\_Year.x : num 2010 2010 2010 2010 2010 ...

## $ Export\_Year\_Month : chr "2010-07" "2010-08" "2010-09" "2010-10" ...

## $ Export\_Year.y : num 2010 2010 2010 2010 2010 ...

## $ Yearly\_Export\_Total\_Amount : num 19129365 17046904 17818461 21927173 18764739 ...

**library**("dplyr")

exp\_data\_amount\_and\_others$Export\_Year.x <- as.numeric(as.character(exp\_data\_amount\_and\_others$Export\_Year.x))

str(exp\_data\_amount\_and\_others)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 94 obs. of 9 variables:

## $ Export\_Date : Date, format: "2010-07-01" "2010-08-01" ...

## $ Export\_Total\_Amount : num 19129365 17046904 17818461 21927173 18764739 ...

## $ Consumer\_Price\_Index\_Yearly\_Change : num 7.58 8.33 9.24 8.62 7.29 6.4 4.9 4.16 3.99 4.26 ...

## $ Consumer\_Price\_Index\_Monthly\_Change: num -0.48 0.4 1.23 1.83 0.03 -0.3 0.41 0.73 0.42 0.87 ...

## $ USD\_Rate : num 1.53 1.5 1.49 1.42 1.43 ...

## $ Export\_Year.x : num 2010 2010 2010 2010 2010 ...

## $ Export\_Year\_Month : chr "2010-07" "2010-08" "2010-09" "2010-10" ...

## $ Export\_Year.y : num 2010 2010 2010 2010 2010 ...

## $ Yearly\_Export\_Total\_Amount : num 19129365 17046904 17818461 21927173 18764739 ...

colnames(exp\_data\_amount\_and\_others)[colnames(exp\_data\_amount\_and\_others) == 'Export\_Year.x'] <- 'Export\_Year'

exp\_data\_amount\_and\_others\_yearly <-

exp\_data\_amount\_and\_others %>%

group\_by(Export\_Year) %>%

summarise(Yearly\_Avg\_Export\_Amount=mean(Yearly\_Export\_Total\_Amount),

Yearly\_Avg\_Consumer\_Price\_Index\_Yearly\_Change = mean(Consumer\_Price\_Index\_Yearly\_Change),

Yearly\_Avg\_Consumer\_Price\_Index\_Monthly\_Change = mean(Consumer\_Price\_Index\_Monthly\_Change),

Yearly\_Avg\_USD\_Rate = mean(USD\_Rate))

str(exp\_data\_amount\_and\_others)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 94 obs. of 9 variables:

## $ Export\_Date : Date, format: "2010-07-01" "2010-08-01" ...

## $ Export\_Total\_Amount : num 19129365 17046904 17818461 21927173 18764739 ...

## $ Consumer\_Price\_Index\_Yearly\_Change : num 7.58 8.33 9.24 8.62 7.29 6.4 4.9 4.16 3.99 4.26 ...

## $ Consumer\_Price\_Index\_Monthly\_Change: num -0.48 0.4 1.23 1.83 0.03 -0.3 0.41 0.73 0.42 0.87 ...

## $ USD\_Rate : num 1.53 1.5 1.49 1.42 1.43 ...

## $ Export\_Year : num 2010 2010 2010 2010 2010 ...

## $ Export\_Year\_Month : chr "2010-07" "2010-08" "2010-09" "2010-10" ...

## $ Export\_Year.y : num 2010 2010 2010 2010 2010 ...

## $ Yearly\_Export\_Total\_Amount : num 19129365 17046904 17818461 21927173 18764739 ...

05-Yearly Average Export Amount and Other Factors

*#Export\_Yearly\_Avg\_Values*

**library**("ggplot2")

**library**("plotly")

p <- exp\_data\_amount\_and\_others\_yearly %>%

ggplot(aes(Yearly\_Avg\_USD\_Rate, Yearly\_Avg\_Export\_Amount, size = Yearly\_Avg\_Consumer\_Price\_Index\_Yearly\_Change, color=Export\_Year)) +

geom\_point() +

scale\_x\_log10() +

theme\_bw()+

xlab("USD Rate") +

ylab("Export Amount(1000$)") +

ggtitle("Yearly Average Export Amount and Other Factors")

ggplotly(p)

2342.0e+072.2e+072.4e+072.6e+07

Yearly Average Export Amount and Other Factors20102012201420162018Export\_YearUSD RateExport Amount(1000$)

exp\_min\_max\_mean\_by\_sectors<- exp\_data %>%

filter(Main\_Sector\_Flag==1)%>%

group\_by(Export\_Year,Sector\_Name\_Eng) %>%

summarise\_each(funs(min(.,na.rm=TRUE), round(mean(.,na.rm=TRUE),digits = 3), max(.,na.rm=TRUE),sum(.,na.rm=TRUE)),Export\_Amount)

## `summarise\_each()` is deprecated.

## Use `summarise\_all()`, `summarise\_at()` or `summarise\_if()` instead.

## To map `funs` over a selection of variables, use `summarise\_at()`

head(exp\_min\_max\_mean\_by\_sectors)

## # A tibble: 6 x 6

## # Groups: Export\_Year [1]

## Export\_Year Sector\_Name\_Eng min round max sum

## <dbl> <chr> <dbl> <dbl> <dbl> <dbl>

## 1 2010 Agriculture and forestry 2.87e5 4.11e5 5.70e5 4.93e6

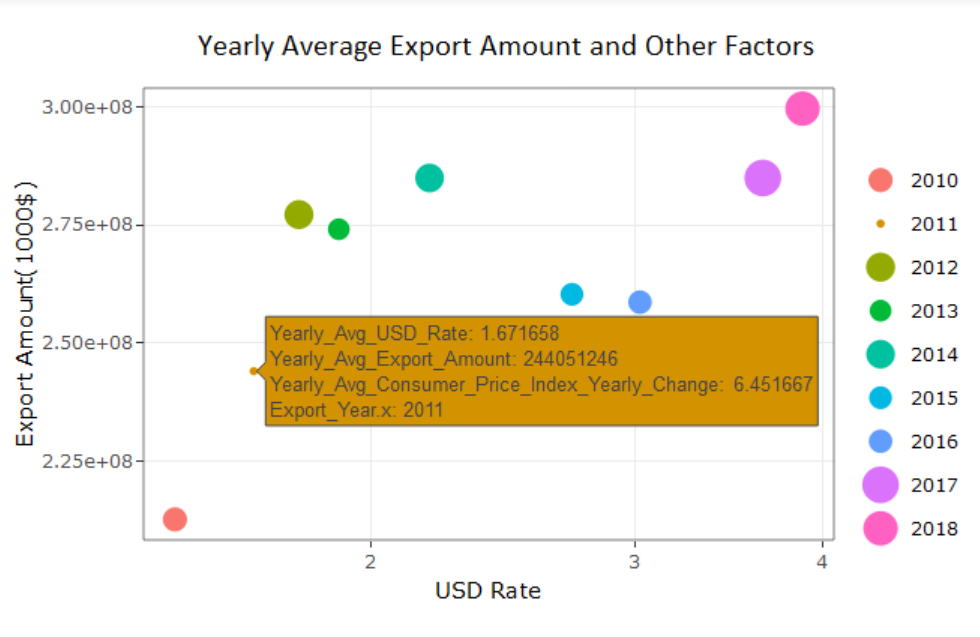
## 2 2010 Electricity, gas and water s~ 7.61e3 1.51e4 2.63e4 1.81e5

## 3 2010 Fishing 8.28e3 1.30e4 2.57e4 1.56e5

## 4 2010 Manufacturing 7.18e6 8.79e6 1.09e7 1.05e8

## 5 2010 Mining and quarrying 1.48e5 2.24e5 2.60e5 2.69e6

## 6 2010 Other community, social and ~ 4.18e1 3.01e2 5.89e2 3.62e3



Yearly average export amount and other factors show the inflation rate of the points in the chart. At every point can be seen yearly average USD rate, yearly average export amount, yearly average consumer price index yearly change.

06\_Export\_Sector\_Share

exp\_share\_sectors <-

exp\_data %>%

filter(Main\_Sector\_Flag==1 & Export\_Date<'2018-11-01')%>%

group\_by(Sector\_Name\_Eng) %>%

summarize(Export\_Amount\_Share=sum(Export\_Amount)) %>%

mutate (Export\_Amount\_Share=round((Export\_Amount\_Share/sum(Export\_Amount\_Share)),4))

exp\_share\_sectors$share\_z <- round((exp\_share\_sectors$Export\_Amount\_Share - mean(exp\_share\_sectors$Export\_Amount\_Share))/sd(exp\_share\_sectors$Export\_Amount\_Share), 2)

exp\_share\_sectors$above\_or\_below <- ifelse(exp\_share\_sectors$share\_z < 0, "Below", "Above")

exp\_share\_sectors <- exp\_share\_sectors[order(exp\_share\_sectors$share\_z), ]

exp\_share\_sectors$Sector\_Name\_Eng <- factor(exp\_share\_sectors$Sector\_Name\_Eng, levels = exp\_share\_sectors$Sector\_Name\_Eng)

theme\_set(theme\_bw())

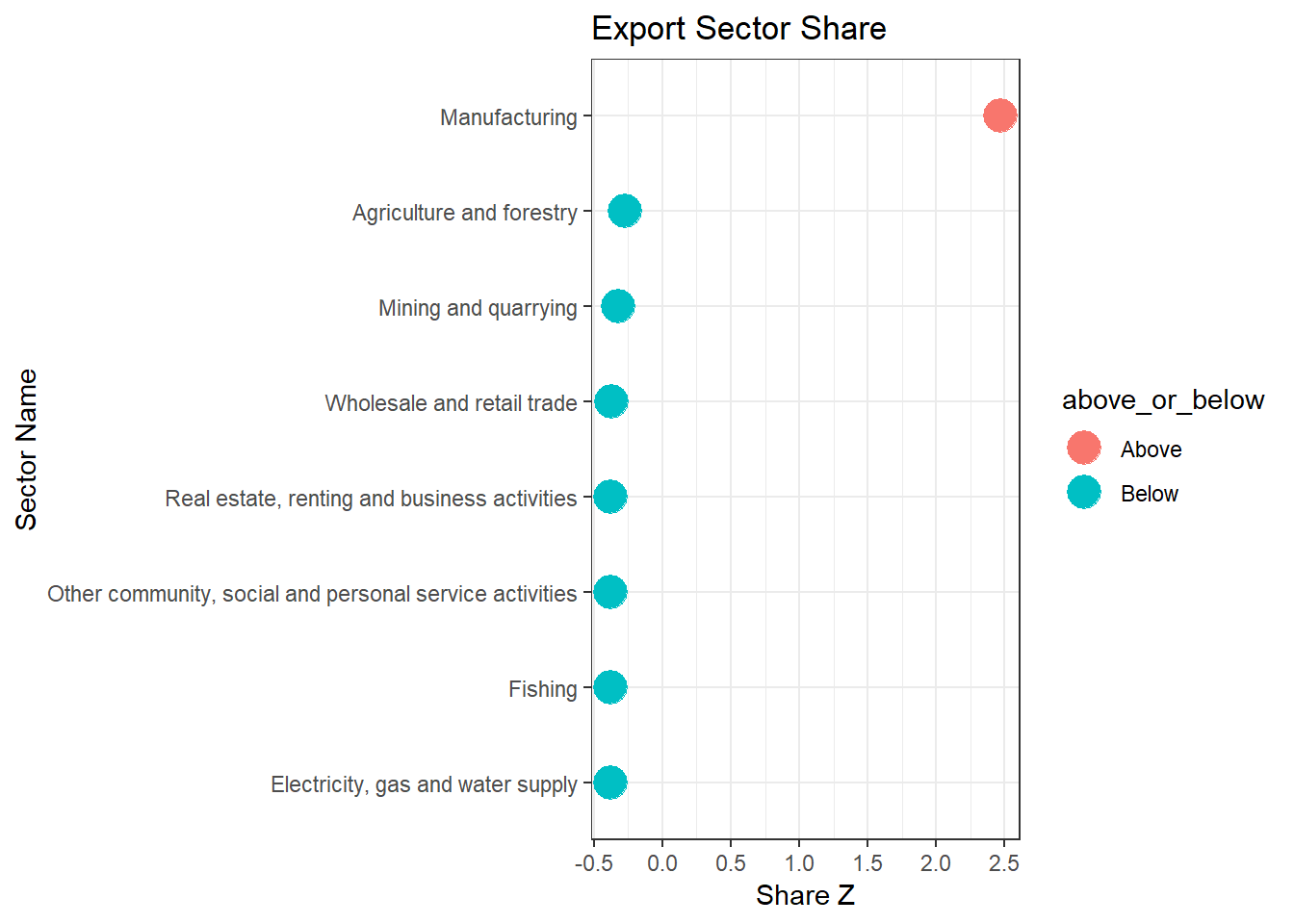
ggplot(exp\_share\_sectors, aes(x= share\_z , y= Sector\_Name\_Eng, label=share\_z)) +

xlab("Share Z") +

ylab("Sector Name") +

ggtitle("Export Sector Share")+

geom\_point(stat='identity', aes(col=above\_or\_below), size=6) *#+*



06\_Import\_Sector\_Share

imp\_share\_sectors <-

imp\_data %>%

filter(Main\_Sector\_Flag==1 & Import\_Date<'2018-11-01')%>%

group\_by(Sector\_Name\_Eng) %>%

summarize(Import\_Amount\_Share=sum(Import\_Amount)) %>%

mutate (Import\_Amount\_Share=round((Import\_Amount\_Share/sum(Import\_Amount\_Share)),4))

imp\_share\_sectors$share\_z <- round((imp\_share\_sectors$Import\_Amount\_Share - mean(imp\_share\_sectors$Import\_Amount\_Share))/sd(imp\_share\_sectors$Import\_Amount\_Share), 2)

imp\_share\_sectors$above\_or\_below <- ifelse(imp\_share\_sectors$share\_z < 0, "Below", "Above")

imp\_share\_sectors <- imp\_share\_sectors[order(imp\_share\_sectors$share\_z), ]

imp\_share\_sectors$Sector\_Name\_Eng <- factor(imp\_share\_sectors$Sector\_Name\_Eng, levels = imp\_share\_sectors$Sector\_Name\_Eng)

theme\_set(theme\_bw())

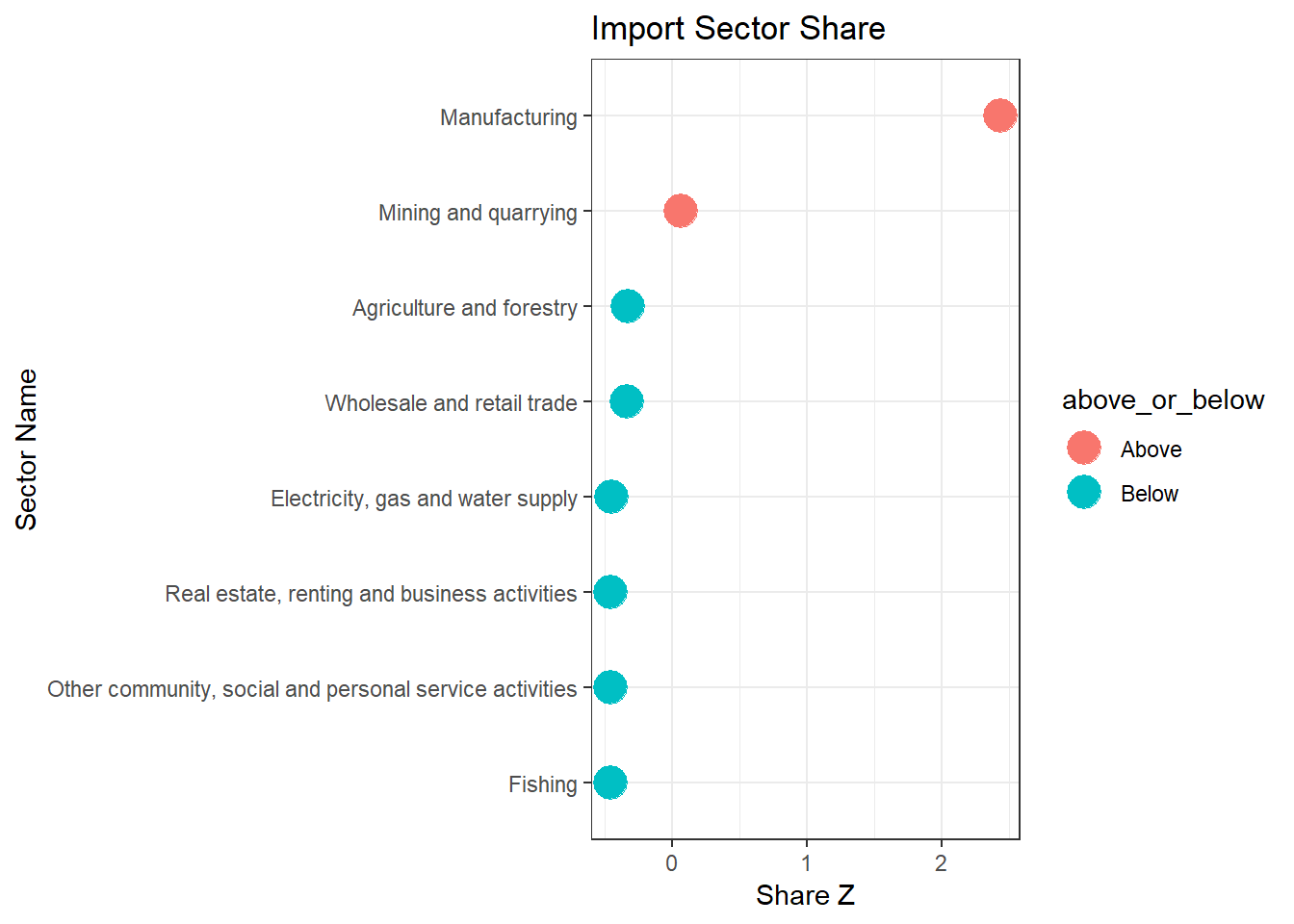
ggplot(imp\_share\_sectors, aes(x= share\_z , y= Sector\_Name\_Eng, label=share\_z)) +

xlab("Share Z") +

ylab("Sector Name") +

ggtitle("Import Sector Share")+

geom\_point(stat='identity', aes(col=above\_or\_below), size=6)



The graph shows the sector shares in the total import amount. Only two sector is above the sector average import amount because of that manufacturing is extremely higher than other sectors. Mining is also important but manufacturing import amount is higher than the total of the other sector import amounts.

07\_Export\_Amount\_by\_Sectors\_And\_Year

exp\_agg\_by\_sectors<- exp\_data %>%

filter(Main\_Sector\_Flag==1)%>%

group\_by(Export\_Year,Sector\_Name\_Eng) %>%

summarise\_each(funs(min(.,na.rm=TRUE), round(mean(.,na.rm=TRUE),digits = 3), max(.,na.rm=TRUE),sum(.,na.rm=TRUE)),Export\_Amount)

## `summarise\_each()` is deprecated.

## Use `summarise\_all()`, `summarise\_at()` or `summarise\_if()` instead.

## To map `funs` over a selection of variables, use `summarise\_at()`

**library**(ggplot2)

**library**(plotly)

**library**(gapminder)

colnames(exp\_agg\_by\_sectors)<- c("Export\_Year", "Sector\_Name\_Eng", "Min\_Amount", "Avg\_Amount", "Max\_Amount", "Total\_Amount" )

p <- exp\_agg\_by\_sectors %>%

ggplot(aes(x = Export\_Year, y= Total\_Amount, group=Min\_Amount, group2 = Max\_Amount, size = Avg\_Amount, color=Sector\_Name\_Eng)) +

geom\_point() +

scale\_x\_log10() +

theme\_bw()+

scale\_size\_area("Nitrogen") +

xlab("Export Year") +

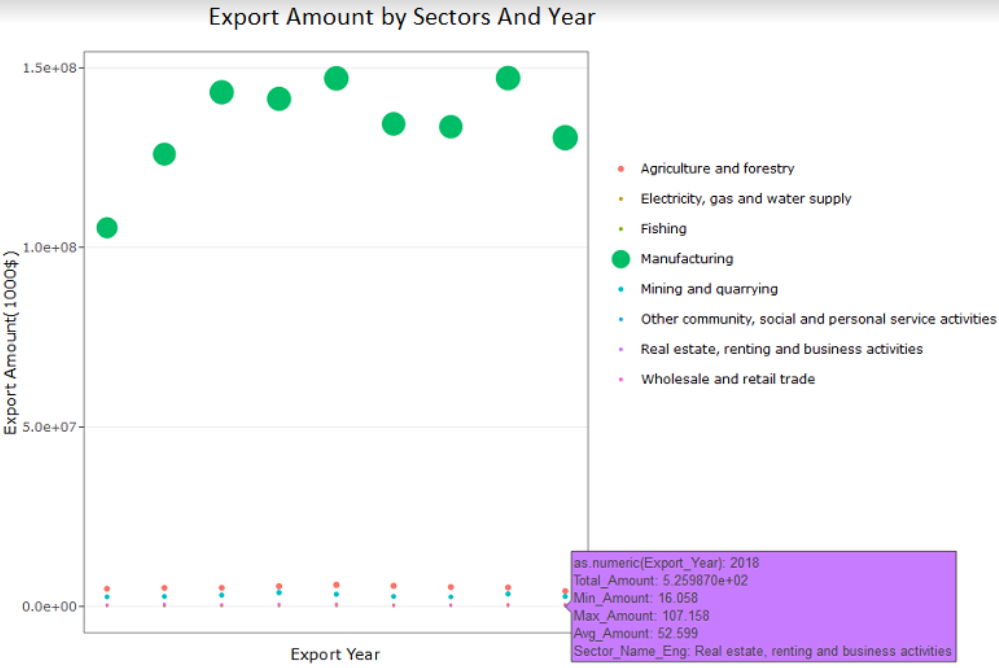
ylab("Export Amount(1000$)") +

ggtitle("Export Amount by Sectors and Years")

ggplotly(p)

0.0e+005.0e+071.0e+081.5e+08

Agriculture and forestryElectricity, gas and water supplyFishingManufacturingMining and quarryingOther community, social and personal service activitiesReal estate, renting and business activitiesWholesale and retail tradeExport Amount by Sectors and YearsExport YearExport Amount(1000$)NitrogenSector\_Name\_Eng



The graph shows the export amounts by sectors and years. We can see that manufacturing is the main part of the export amount. Manufacturing export amount is sharply increased between 2010-2013 and it is in a cycle of increase-decrease-stable for every 3 year between 2013-2018.

08\_Import\_Amount\_by\_Sectors\_And\_Year

imp\_agg\_by\_sectors<- imp\_data %>%

filter(Main\_Sector\_Flag==1)%>%

group\_by(Import\_Year,Sector\_Name\_Eng) %>%

summarise\_each(funs(min(.,na.rm=TRUE), round(mean(.,na.rm=TRUE),digits = 3), max(.,na.rm=TRUE),sum(.,na.rm=TRUE)),Import\_Amount)

## `summarise\_each()` is deprecated.

## Use `summarise\_all()`, `summarise\_at()` or `summarise\_if()` instead.

## To map `funs` over a selection of variables, use `summarise\_at()`

colnames(imp\_agg\_by\_sectors)<- c("Import\_Year", "Sector\_Name\_Eng", "Min\_Amount", "Avg\_Amount", "Max\_Amount", "Total\_Amount" )

p <- imp\_agg\_by\_sectors %>%

ggplot(aes(x= as.numeric(Import\_Year), y= Total\_Amount, group=Min\_Amount, group2 = Max\_Amount, size = Avg\_Amount, color=Sector\_Name\_Eng)) +

geom\_point() +

scale\_x\_log10() +

theme\_bw()+

scale\_size\_area("Nitrogen") +

xlab("Import Year") +

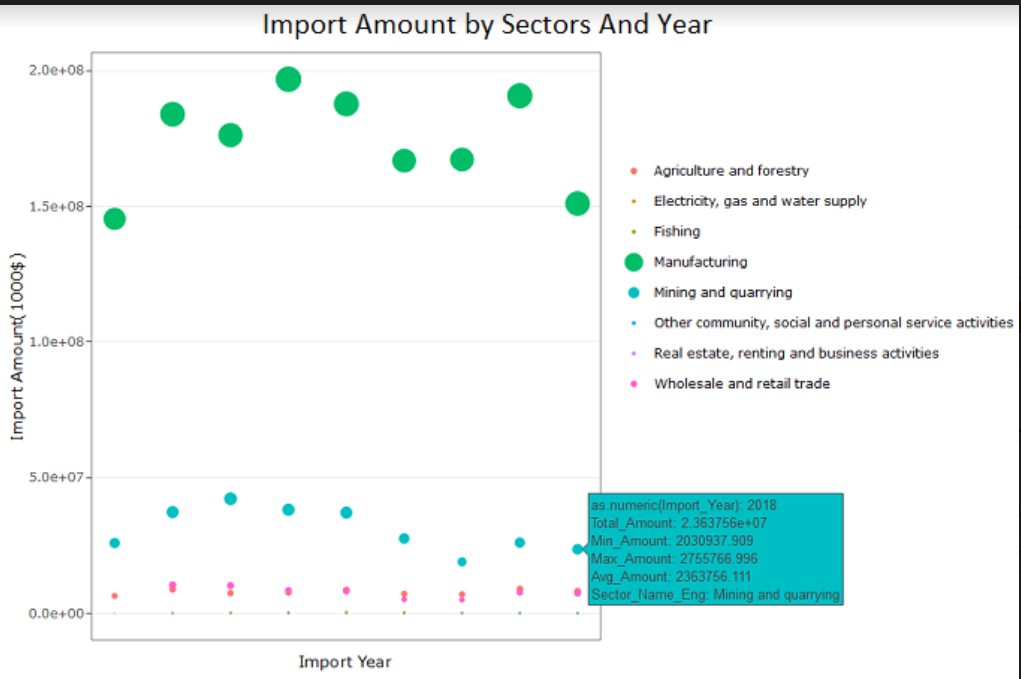
ylab("Import Amount(1000$)") +

ggtitle("Import Amount by Sectors and Years")

ggplotly(p)

0.0e+005.0e+071.0e+081.5e+082.0e+08

Agriculture and forestryElectricity, gas and water supplyFishingManufacturingMining and quarryingOther community, social and personal service activitiesReal estate, renting and business activitiesWholesale and retail tradeImport Amount by Sectors and YearsImport YearImport Amount(1000$)NitrogenSector\_Name\_Eng



The graph shows import amounts by sectors and years. Manufacturing is the main port of our import. Mining amounts gives us a chance to compare manufacturing amounts and mining amount trends. We can see that there is almost a synchronization between manufacturing and mining trends which may mean overall factors affects these two sectors in the same way and with a nearly equal rate.

09\_Export\_Amount\_USD\_Rate\_Inflation

colors = c("red", "blue", "green")

*# Set the margins of the plot wider*

par(oma = c(0, 2, 2, 3))

plot(exp\_data\_final$Export\_Date, exp\_data\_final$Export\_Total\_Amount, yaxt = "n", xlab = "Export Date", main = "Export Amount & USD Rate & Inflation",

ylab = "")

lines(exp\_data\_final$Export\_Date, exp\_data\_final$Export\_Total\_Amount)

*# We use the "pretty" function go generate nice axes*

axis(at = pretty(exp\_data\_final$Export\_Total\_Amount), side = 2)

**library**("tidyverse")

exp\_data\_final <- exp\_data\_final %>%

select(Export\_Date,Export\_Total\_Amount,USD\_Rate,Consumer\_Price\_Index\_Yearly\_Change

,Consumer\_Price\_Index\_Monthly\_Change,Export\_Year,Export\_Year\_Month)

str(exp\_data\_final)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 94 obs. of 7 variables:

## $ Export\_Date : Date, format: "2010-07-01" "2010-08-01" ...

## $ Export\_Total\_Amount : num 19129365 17046904 17818461 21927173 18764739 ...

## $ USD\_Rate : num 1.53 1.5 1.49 1.42 1.43 ...

## $ Consumer\_Price\_Index\_Yearly\_Change : num 7.58 8.33 9.24 8.62 7.29 6.4 4.9 4.16 3.99 4.26 ...

## $ Consumer\_Price\_Index\_Monthly\_Change: num -0.48 0.4 1.23 1.83 0.03 -0.3 0.41 0.73 0.42 0.87 ...

## $ Export\_Year : num 2010 2010 2010 2010 2010 ...

## $ Export\_Year\_Month : chr "2010-07" "2010-08" "2010-09" "2010-10" ...

*# The side for the axes. The next one will go on*

*# the left, the following two on the right side*

sides <- list(2, 4, 4)

*# The number of "lines" into the margin the axes will be*

lines <- list(2, NA, 2)

**for**(i **in** 3:5) {

par(new = TRUE)

plot(exp\_data\_final$Export\_Date, exp\_data\_final[[i]], axes = FALSE, col = colors[i - 1], xlab = "", ylab = "")

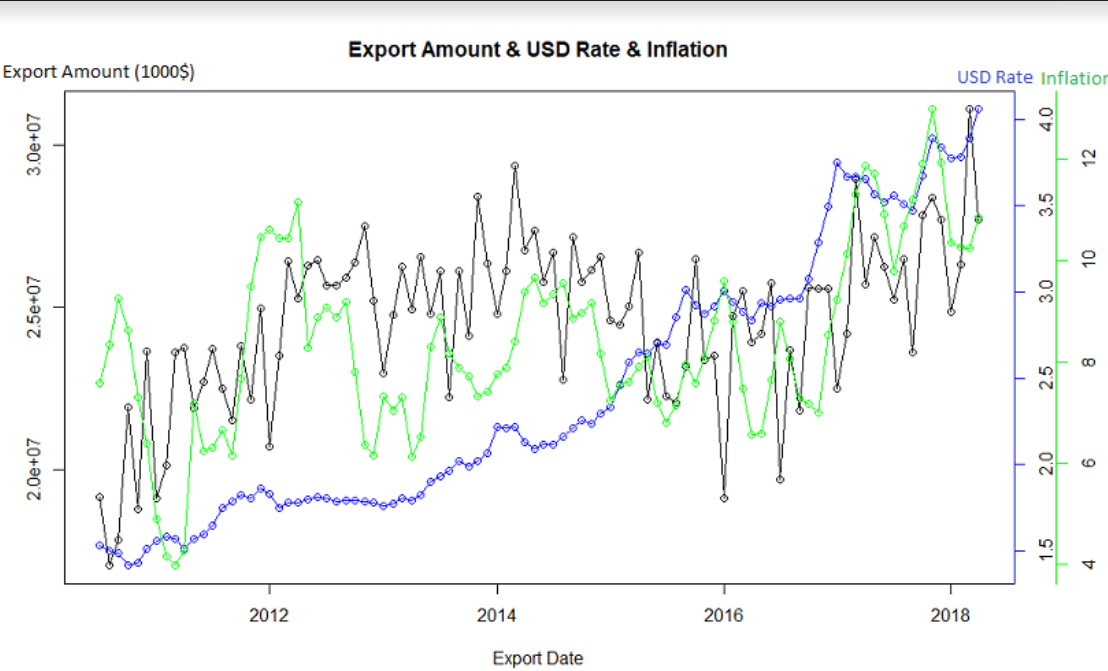
axis(at = pretty(exp\_data\_final[[i]]), side = sides[[i-2]], line = lines[[i-2]],

col = colors[i - 1])

*#mtext(2,text=colnames(exp\_data\_final)[i],line=2)*

lines(exp\_data\_final$Export\_Date, exp\_data\_final[[i]], col = colors[i - 1])

}



The graph shows export amounts, USD rate and inflation. First thing that take attention is USD rate is increasing slowly or fast but it is always increasing however export amount and inflation is fluctuated by years. Export amount and inflation is nearly synchronized. If we look at 2014-2016 USD rate is increasing and export amount is decreased significantly however in 2016-2018 USD rate is increasing again and export amount is increased significantly which means that only USD rate is not enough to explain direction and fluctuations in export amounts.

10\_Trade\_Deficit\_Amount\_by\_Sub\_Sectors

```{r, warning=FALSE, echo=FALSE}

emp\_by\_sectors <- exp\_data %>%

filter(Main\_Sector\_Flag == 0) %>%

group\_by(Sector\_Name\_Eng,Sub\_Sector\_Type\_Code) %>%

summarize(Total\_Export\_Amount = sum(Export\_Amount))

imp\_by\_sectors <- imp\_data %>%

filter(Main\_Sector\_Flag == 0) %>%

group\_by(Sector\_Name\_Eng,Sub\_Sector\_Type\_Code) %>%

summarize(Total\_Import\_Amount = sum(Import\_Amount))

trade\_deficit\_sub\_sector <- inner\_join(imp\_by\_sectors, emp\_by\_sectors, by =c("Sub\_Sector\_Type\_Code", "Sub\_Sector\_Type\_Code")) %>%

mutate(Trade\_Deficit\_Amount = Total\_Import\_Amount - Total\_Export\_Amount)%>%

arrange(desc(Trade\_Deficit\_Amount))

trade\_deficit\_sub\_sector %>%

ggplot(aes(x=reorder(Sector\_Name\_Eng.x, Trade\_Deficit\_Amount),

y=Trade\_Deficit\_Amount, color =Sector\_Name\_Eng.x, fill=Sector\_Name\_Eng.x)) +

geom\_bar(stat = "identity") +

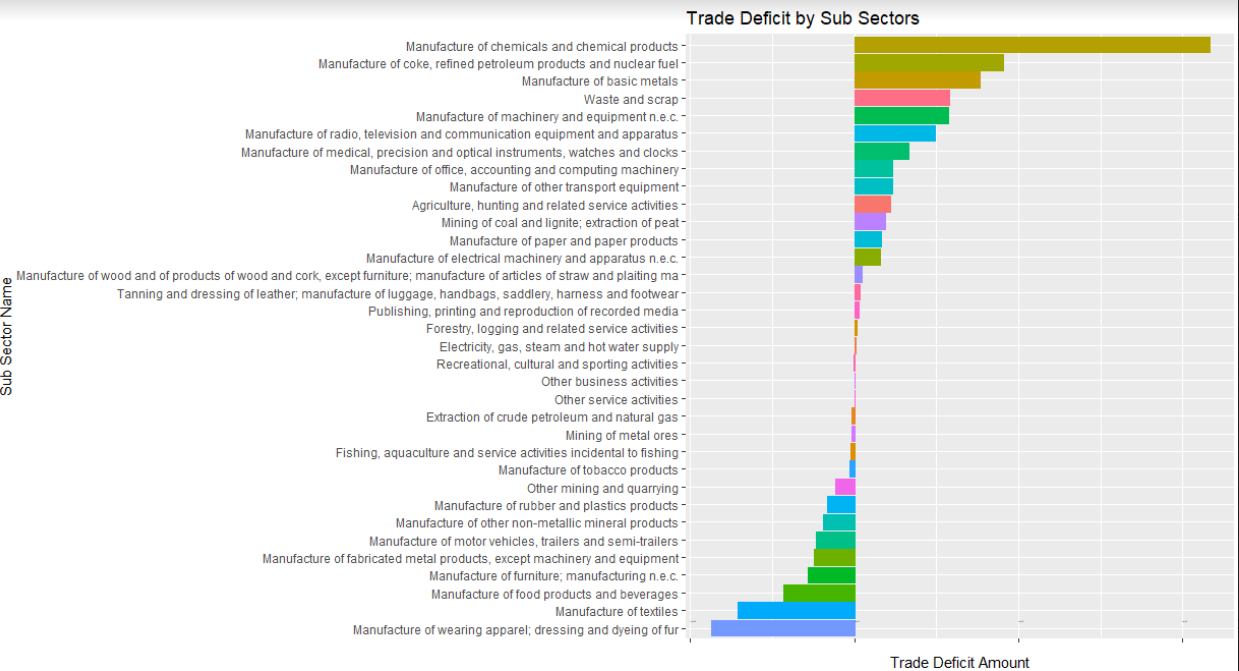
coord\_flip() +

labs(x = "Sub Sector Name", y = "Trade Deficit Amount") +

ggtitle("Trade Deficit by Sub Sectors") +

theme(legend.position = "none", axis.text.x = element\_text(angle = 0, vjust = 10.0, hjust = 0.0, size = 1))

```

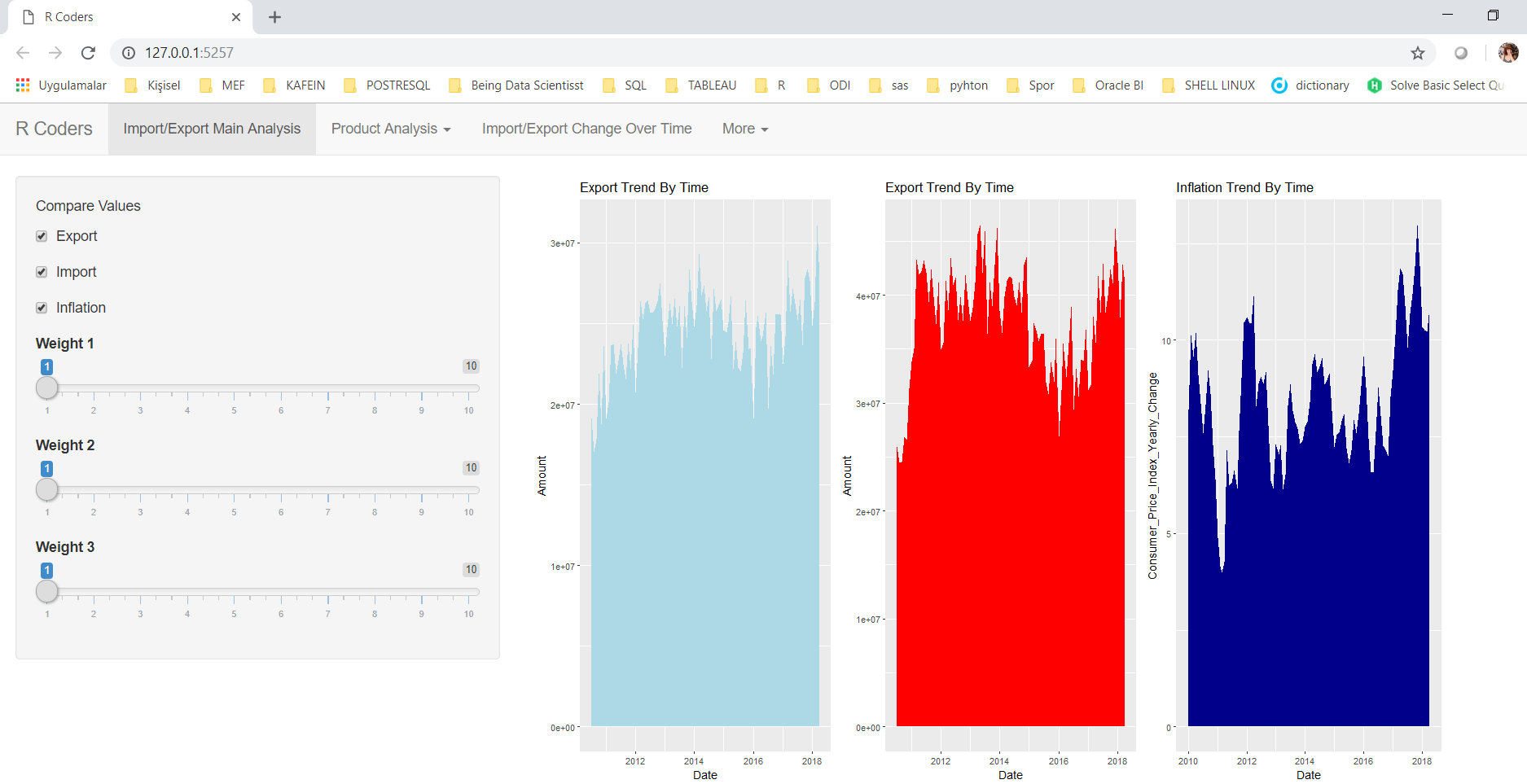


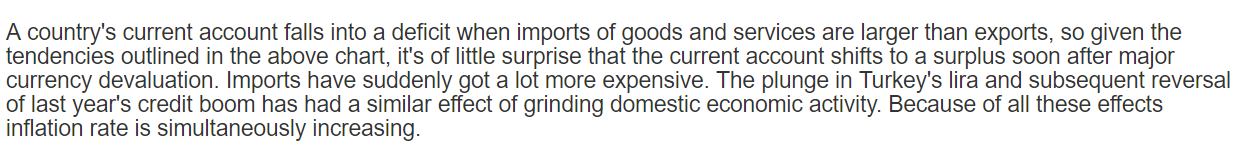
* When export sub-sectors are investigated for last 3 years, we saw that top 3 export subsectors are manufacture of motor vehicles and trailers, manufacture of basic metals and manufacture of textiles.
* When import subsectors are investigated for last 3 years, we saw that top 3 import subsectors are manufacture of basic metals, manufacture of chemicals and chemical products, mining and quarrying.

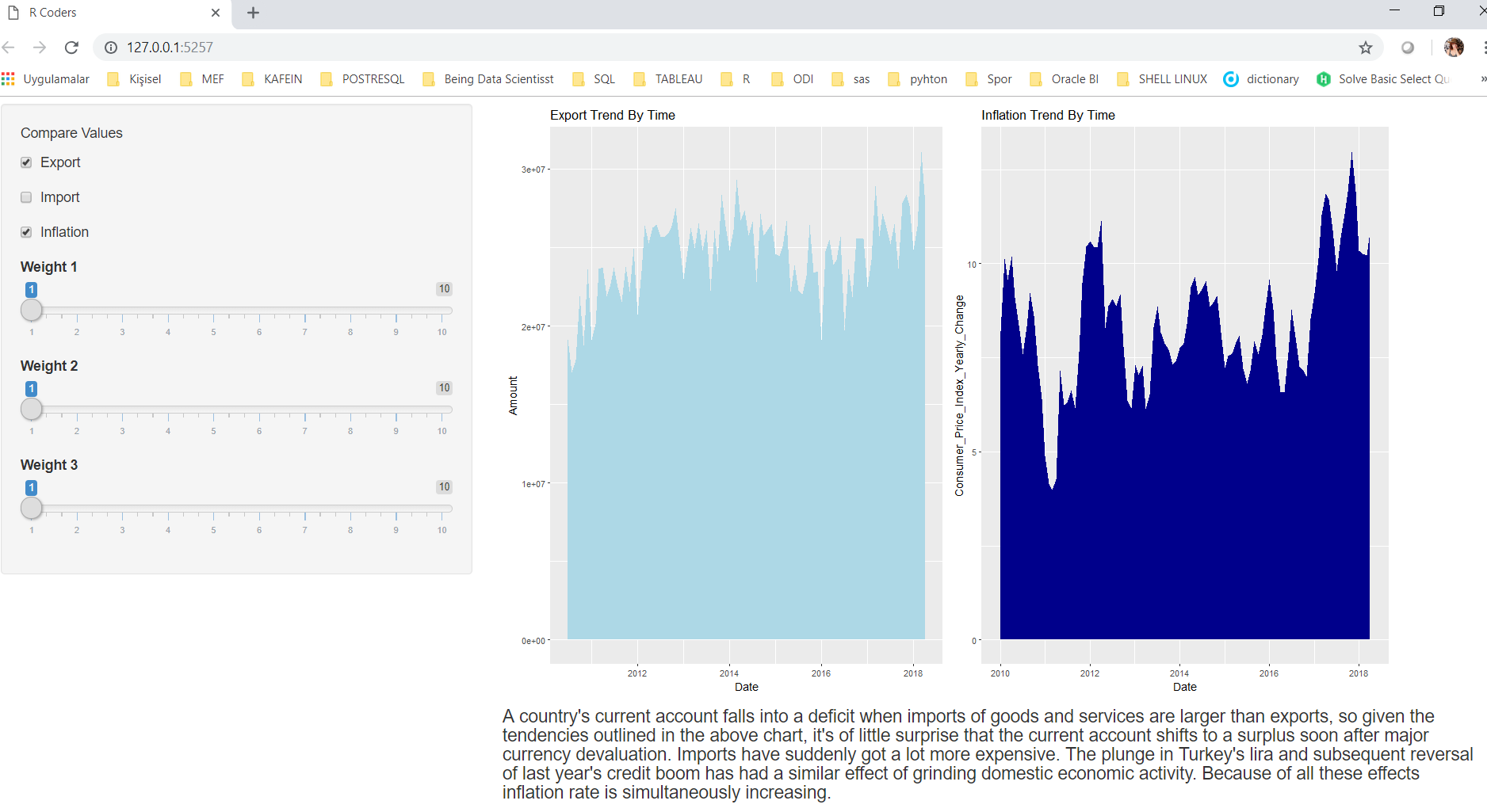
SHINY APP

The detail version of shiny app can be seen from this link: <https://mef-bda503.github.io/gpj18-r_coders/>

And also screenshots and analysis of app is like below.



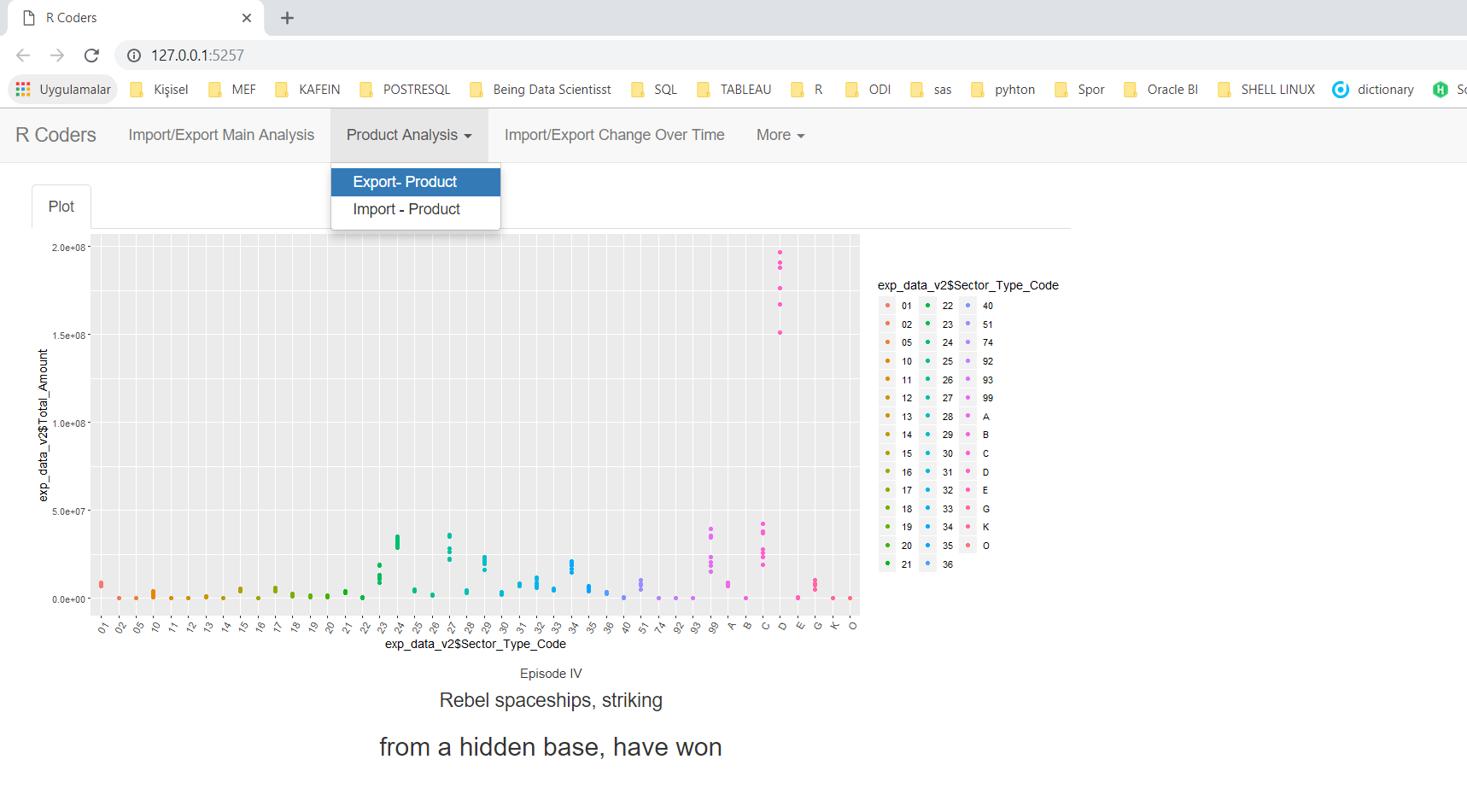




This dynamic app is giving the ability to choose which analysis to be shown to the user.

And also the weight of graphs can be determined by the user.

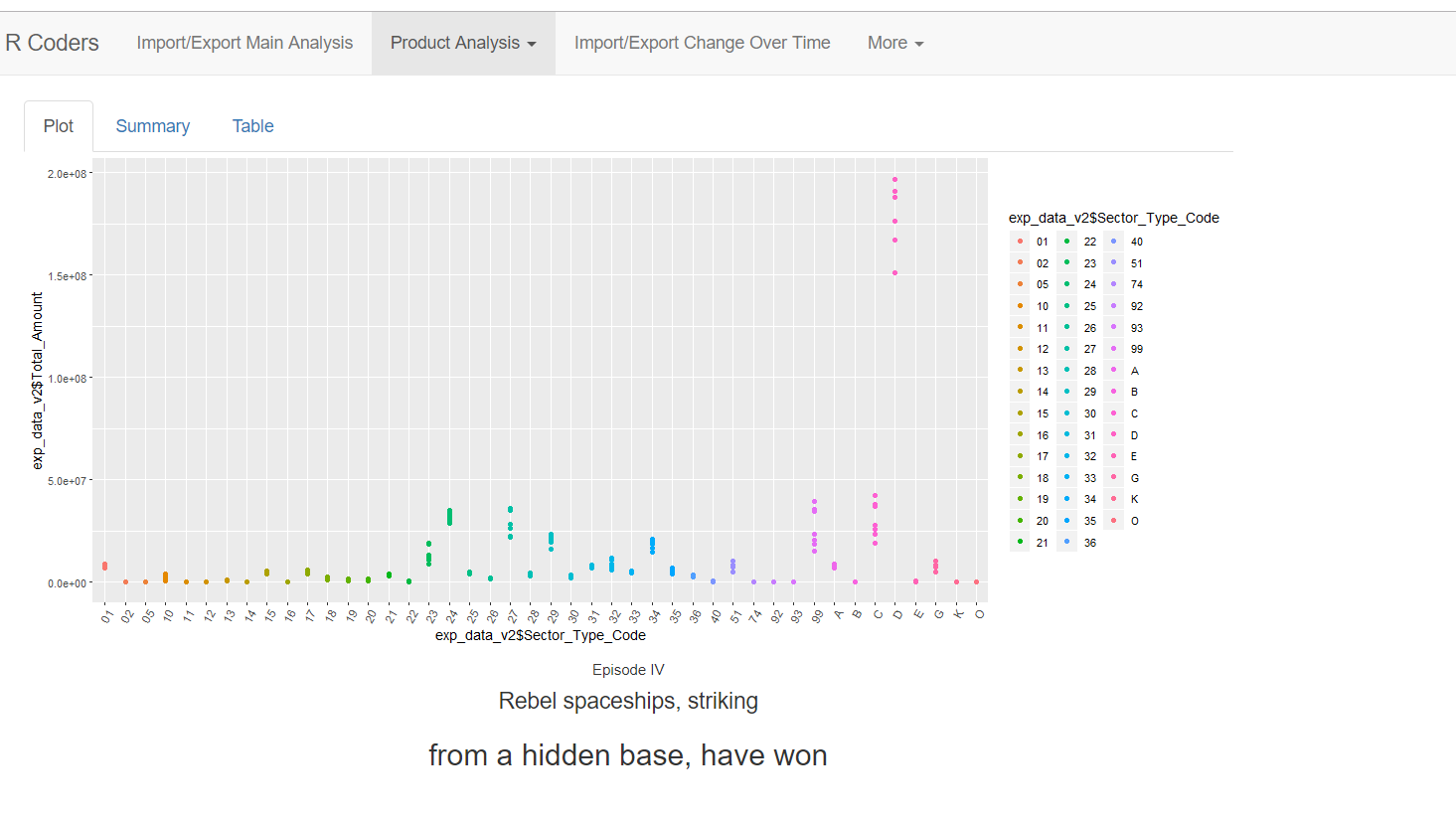
Shiny also gives the opportunity of arrange many different custom manupulations on diagrams like color, size, being smooth. This abilities creates a difference against other visualization tools.

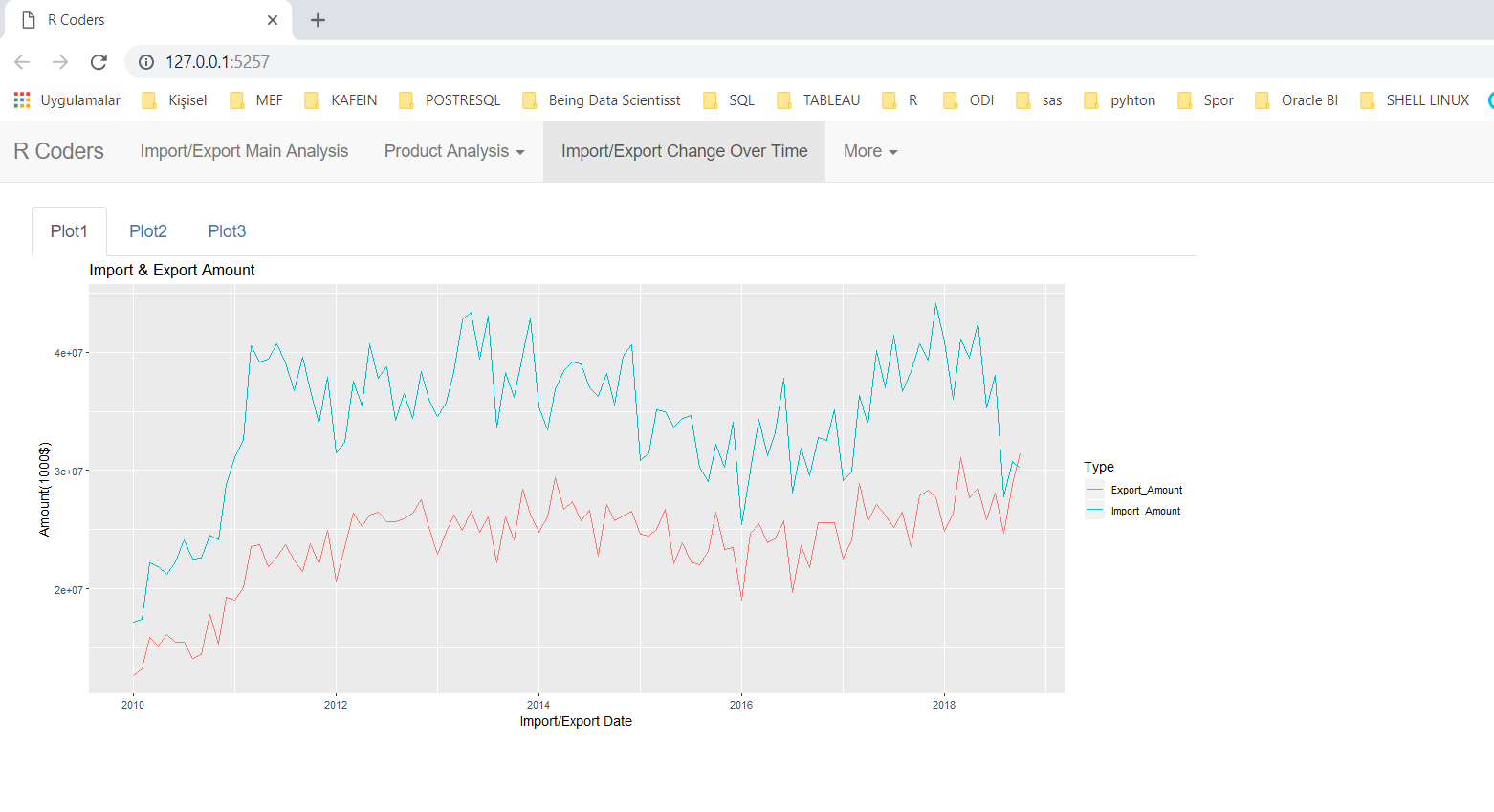


On the Shiny app, dashboard view( As R CODERS we have stole this idea from TABLEAU dashboard structure)

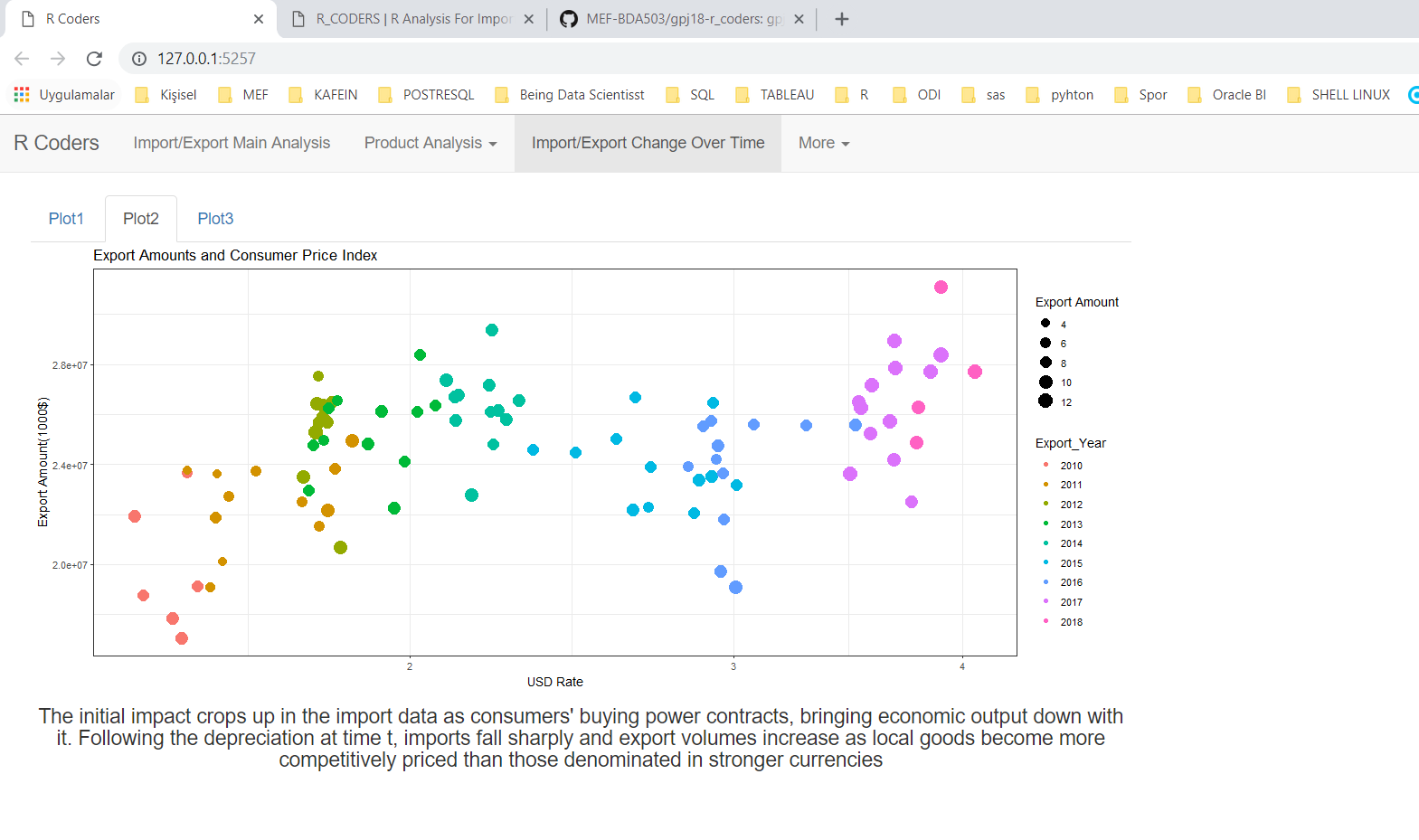
From the perfective of Tableau, R\_CODER’s second worksheet is the Product analysis tab. The user or analyzer can choose either Export products or import products.

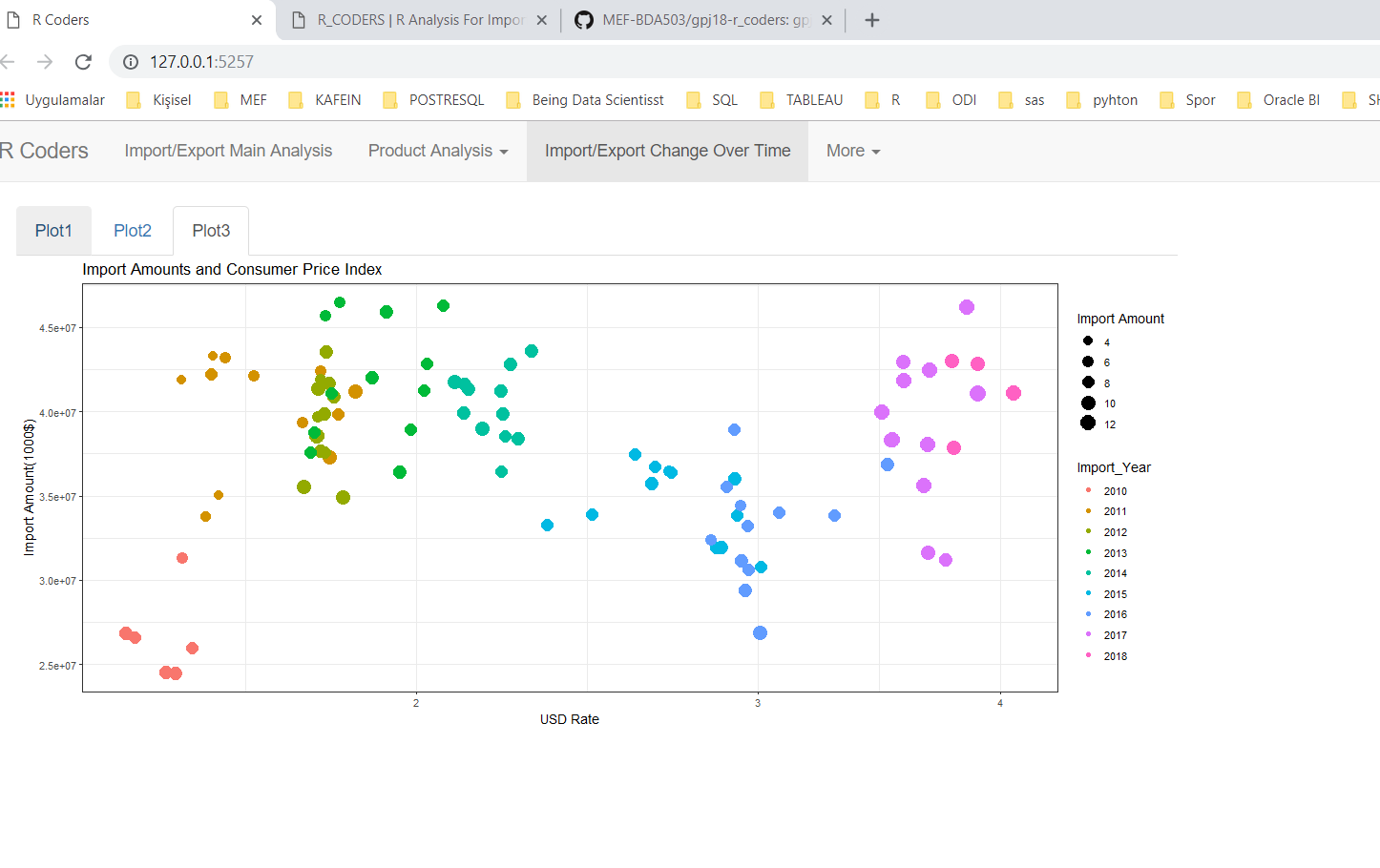
On this diagram product range try to be analyzed.



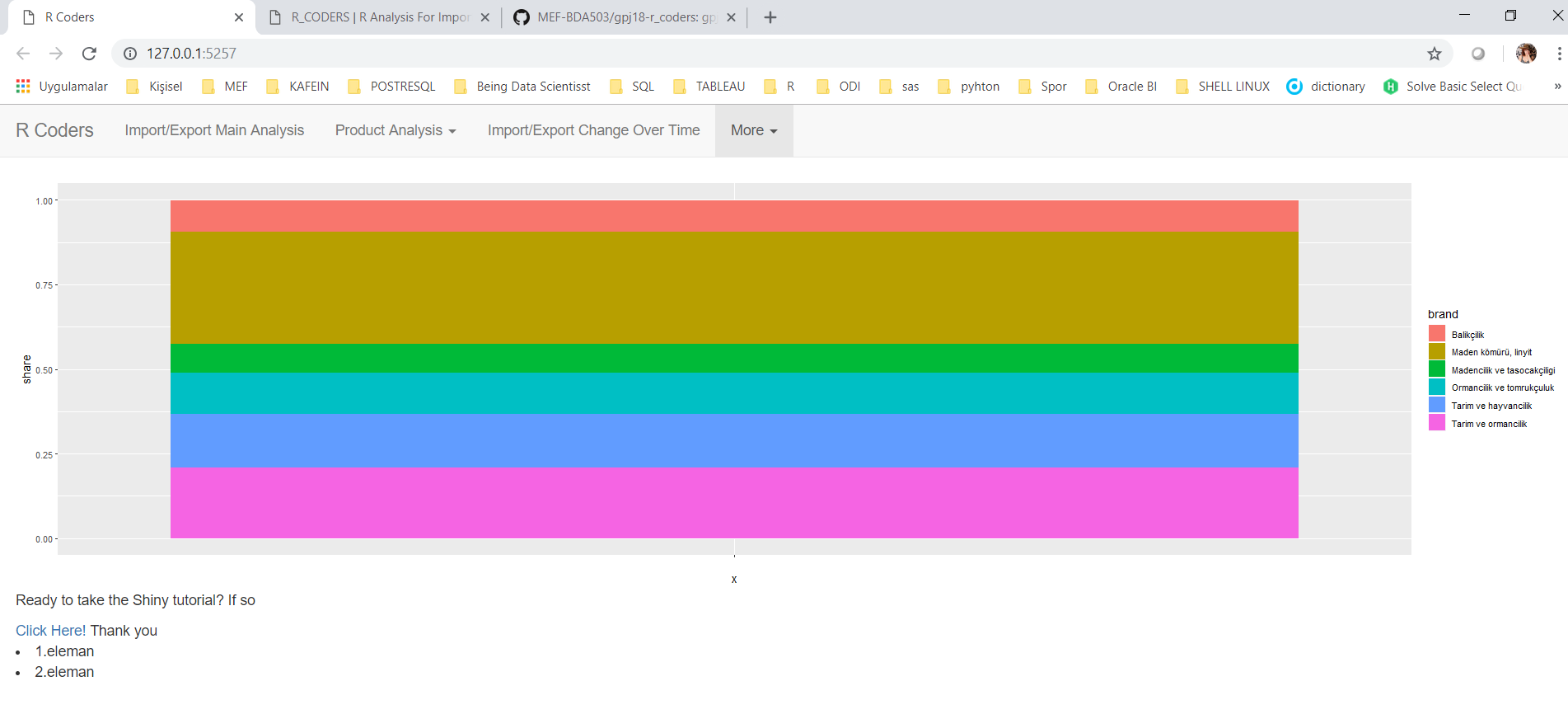


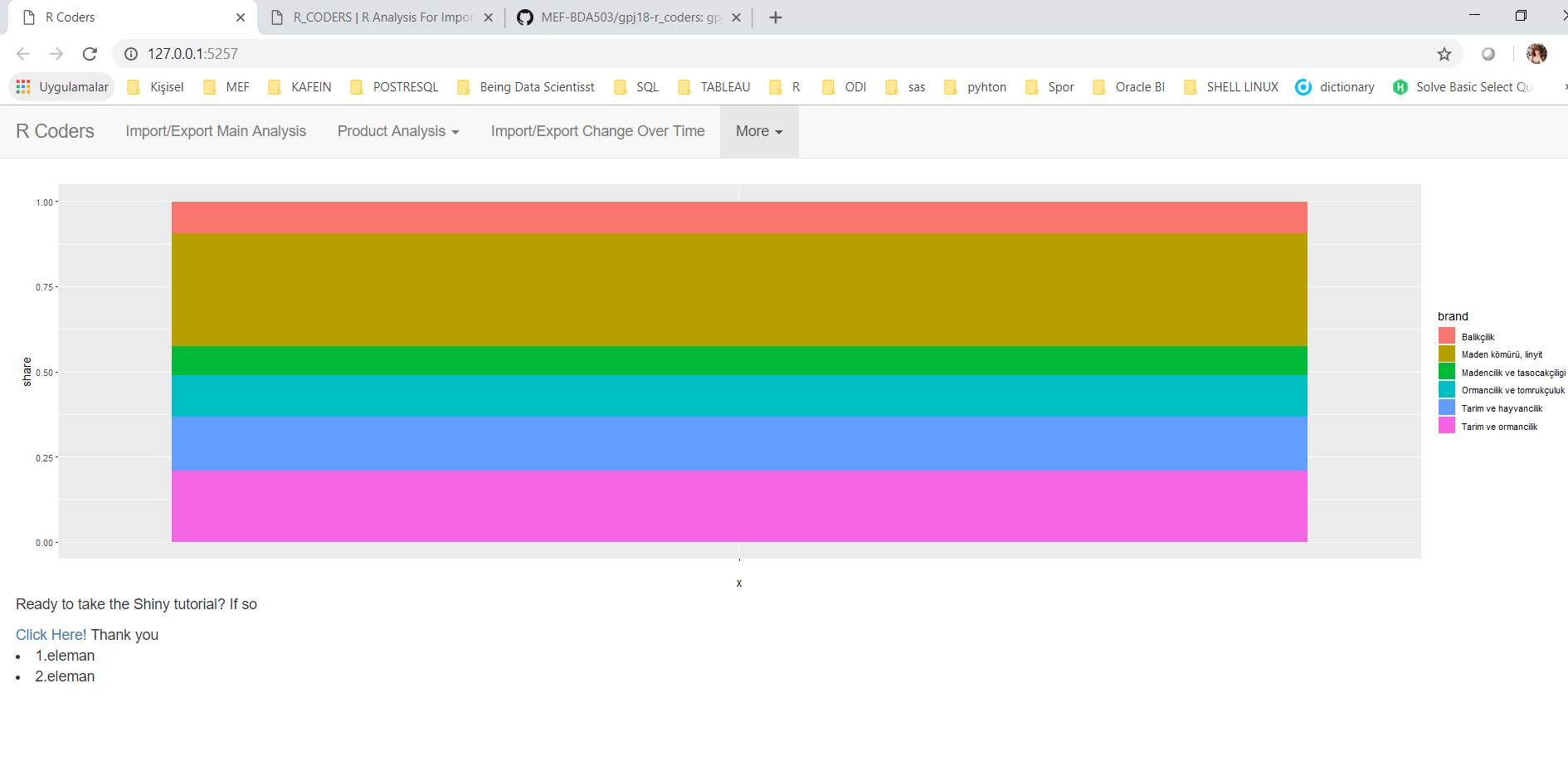
In this diagram Turkey’s foreign trade deficit try to be dedicated. So the conclusion as we saw is that while deficit is decreasing, economic indicators of Turkey is decreasing also such as inflation goes up.





These analysis above Show that the scale of products amount in USD. By the time passes the import amount of Turkish economy is increasing.





Above graps are product groups percentage in import and export total amount. These analysis indicate that , As an import product energy has a great percentage

Here is R\_CODER’s shiny app code:

library(shiny)

library(readxl)

library(tidyverse)

library(ggplot2)

library(dplyr)

library(stringr)

library(rsconnect)

library(plotly)

library(gapminder)

library(gridExtra)

## Import Analysis

tmp<-tempfile(fileext=".xls")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Excel/import\_1996\_2018.xls?raw=true",mode = 'wb',destfile=tmp)

import\_data<-readxl::read\_excel(tmp,skip=7,col\_names=FALSE)

file.remove(tmp)

#Define Colnames

colnames(import\_data) <- c("Year","Sector\_Type\_Code","Sector\_Name", "Total\_Amount", "January", "February", "March", "April", "May", "June", "July","August", "September", "October" ,"November","December")

cols = c(4:15);

import\_data[,cols] = suppressWarnings(apply(import\_data[,cols], 2, function(x) as.numeric(as.character(x))));

str(import\_data)

print(import\_data %>% select(Sector\_Name,January,February,March)) %>% mutate(VATotal = import\_data$January + import\_data$February + import\_data$March) %>% filter(VATotal > 3000000)

## Print No Import Sectors

import\_data %>% select(Sector\_Name) %>% mutate(VADiff = import\_data$January + import\_data$February + import\_data$March ) %>% filter(is.na(VADiff)) %>% distinct()

print(import\_data %>% select(Sector\_Name) %>% mutate(VADiff = import\_data$January + import\_data$February + import\_data$March ) %>% filter(is.na(VADiff)) %>% filter(!(is.na(Sector\_Name))) %>% distinct())

tmp<-tempfile(fileext=".xls")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Excel/import\_1996\_2018.xls?raw=true",mode = 'wb',destfile=tmp)

raw\_data<-readxl::read\_excel(tmp,skip=7,col\_names=FALSE)

file.remove(tmp)

colnames(raw\_data) <- c("Year","Sector\_Type\_Code","Sector\_Name", "Total\_Amount", "January", "February", "March", "April", "May", "June", "July","August", "September", "October" ,"November","December")

cols = c(4:15);

raw\_data[,cols] = suppressWarnings(apply(raw\_data[,cols], 2, function(x) as.numeric(as.character(x))));

raw\_data %>% select(Sector\_Name) %>% mutate(VADiff = raw\_data$January + raw\_data$February)

for (row in 1:nrow(raw\_data)) {

year <- raw\_data[row, "Year"]

if(!is.na(year) & year == 2017){

break

}

raw\_data[row, "Year"] <- 2018

}

v\_year <- 2017

for (row in 1:nrow(raw\_data)) {

year <- raw\_data[row, "Year"]

if(!is.na(year) & year == v\_year){

v\_year <- v\_year - 1

}

raw\_data[row, "Year"] <- v\_year + 1

if (v\_year==2008){

break

}

}

exp\_data\_v2 <- raw\_data %>%

slice(6:391)%>% filter(Sector\_Name != "Toplam -Total")

Months <- c("January","February","March","April","May","June","July","August","September","October","November","December")

Values <- c(1000,1200,1100,1600,1800,1000,1200,1300,2000,1300,1200,1100)

Randoms <- c(1020,1300,1130,1500,1080,2000,2200,1350,2500,1350,1220,1101)

## Import/ Export Union Part

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/imp\_data\_final.rds?raw=true?raw=true",destfile=tmp,mode = 'wb')

imp\_data\_final <- read\_rds(tmp)

file.remove(tmp)

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/exp\_data\_final.rds?raw=true?raw=true",destfile=tmp,mode = 'wb')

exp\_data\_final<-read\_rds(tmp)

file.remove(tmp)

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/imp\_data.rds?raw=true?raw=true",destfile=tmp,mode = 'wb')

imp\_data<-read\_rds(tmp)

file.remove(tmp)

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/exp\_data.rds?raw=true?raw=true",destfile=tmp,mode = 'wb')

exp\_data<-read\_rds(tmp)

file.remove(tmp)

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/Producer\_Inflation.rds?raw=true?raw=true",destfile=tmp,mode = 'wb')

producer\_inf<-read\_rds(tmp)

file.remove(tmp)

# Create a temporary file

tmp=tempfile(fileext=".xls")

# Download file from repository to the temp file

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Excel/export\_import\_sectors.xls?raw=true",destfile=tmp,mode='wb')

# Read that excel file.

sectors <- read\_excel(tmp)

# Remove the temp file

file.remove(tmp)

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/US\_Dollar\_Montly\_Rate.rds?raw=true?raw=true",destfile=tmp,mode = 'wb')

usd\_rate<-read\_rds(tmp)

file.remove(tmp)

names(exp\_data\_final)[names(exp\_data\_final) == 'Date'] <- 'Export\_Date'

names(exp\_data)[names(exp\_data) == 'Date'] <- 'Export\_Date'

names(imp\_data\_final)[names(imp\_data\_final) == 'Date'] <- 'Import\_Date'

names(imp\_data\_final)[names(imp\_data\_final) == 'Export\_Total\_Amount'] <- 'Import\_Total\_Amount' #fix

names(imp\_data)[names(imp\_data) == 'Date'] <- 'Import\_Date'

exp\_data <- inner\_join(exp\_data,sectors, by=c("Sector\_Type\_Code"="Sub\_Sector\_Type\_Code"))

imp\_data <- inner\_join(imp\_data,sectors, by=c("Sector\_Type\_Code"="Sub\_Sector\_Type\_Code"))

exp\_data$Export\_Year<-format(exp\_data$Export\_Date,"%Y")

exp\_data$Export\_Year\_Month<-format(exp\_data$Export\_Date,"%Y-%m")

exp\_data\_final$Export\_Year<-format(exp\_data\_final$Export\_Date,"%Y")

exp\_data\_final$Export\_Year\_Month<-format(exp\_data\_final$Export\_Date,"%Y-%m")

imp\_data$Import\_Year<-format(imp\_data$Import\_Date,"%Y")

imp\_data$Import\_Year\_Month<-format(imp\_data$Import\_Date,"%Y-%m")

imp\_data\_final$Import\_Year<-format(imp\_data\_final$Import\_Date,"%Y")

imp\_data\_final$Import\_Year\_Month<-format(imp\_data\_final$Import\_Date,"%Y-%m")

imp\_data<- imp\_data %>%

select (Import\_Date,Sector\_Type\_Code,Sector\_Type\_Code.y,Main\_Sector\_Flag,Sector\_Name\_Eng,Amount,Import\_Year,Import\_Year\_Month)

exp\_data<- exp\_data %>%

select (Export\_Date,Sector\_Type\_Code,Sector\_Type\_Code.y,Main\_Sector\_Flag,Sector\_Name\_Eng,Amount,Export\_Year,Export\_Year\_Month)

colnames(imp\_data)[colnames(imp\_data) == 'Amount'] <- 'Import\_Amount'

colnames(exp\_data)[colnames(exp\_data) == 'Amount'] <- 'Export\_Amount'

colnames(imp\_data)[colnames(imp\_data) == 'Sector\_Type\_Code'] <- 'Sub\_Sector\_Type\_Code'

colnames(exp\_data)[colnames(exp\_data) == 'Sector\_Type\_Code'] <- 'Sub\_Sector\_Type\_Code'

colnames(imp\_data)[colnames(imp\_data) == 'Sector\_Type\_Code.y'] <- 'Sector\_Type\_Code'

colnames(exp\_data)[colnames(exp\_data) == 'Sector\_Type\_Code.y'] <- 'Sector\_Type\_Code'

imp\_data$Import\_Amount[is.na(imp\_data$Import\_Amount)] <- 0

imp\_data\_final$Import\_Total\_Amount[is.na(imp\_data\_final$Import\_Total\_Amount)] <- 0

exp\_data$Export\_Amount[is.na(exp\_data$Export\_Amount)] <- 0

exp\_data\_final$Export\_Total\_Amount[is.na(exp\_data\_final$Export\_Total\_Amount)] <- 0

exp\_data\_final <- exp\_data\_final %>%

filter(Export\_Date<'2018-11-01')

exp\_data <- exp\_data %>%

filter(Export\_Date<'2018-11-01')

imp\_data\_final <- imp\_data\_final %>%

filter(Import\_Date<'2018-11-01')

imp\_data <- imp\_data %>%

filter(Import\_Date<'2018-11-01')

saveRDS(imp\_data,file="imp\_data\_v2.rds")

saveRDS(imp\_data\_final,file="imp\_data\_final\_v2.rds")

saveRDS(exp\_data,file="exp\_data\_v2.rds")

saveRDS(exp\_data\_final,file="exp\_data\_final\_v2.rds")

imp\_and\_exp\_data <- inner\_join(exp\_data, imp\_data, by=c("Export\_Date" = "Import\_Date","Sub\_Sector\_Type\_Code"="Sub\_Sector\_Type\_Code"))

imp\_and\_exp\_data\_bymonth <- aggregate(cbind(Import\_Amount, Export\_Amount) ~ Export\_Date, data = imp\_and\_exp\_data, sum)

imp\_and\_exp\_data\_bymonth <- gather(imp\_and\_exp\_data\_bymonth,

value = "value",

key = "type",

Export\_Amount, Import\_Amount)

exp\_data\_final\_2 <- exp\_data\_final

imd\_data\_final\_2 <- imp\_data\_final

#Rename column names

colnames(imp\_and\_exp\_data\_bymonth) <- c("Date","Type","Amount")

#Remove Empty Dates

imp\_and\_exp\_data\_bymonth <- imp\_and\_exp\_data\_bymonth %>%

filter(Date<'2018-11-01')

df = data.frame("brand" = c("Tarım ve ormancılık","Tarım ve hayvancılık","Ormancılık ve tomrukçuluk","Balıkçılık","Madencilik ve taşocakçılığı","Maden kömürü, linyit"),

"share" = c(.2090,.1580,.1210,.0930,.0860,.3320))

##--------------##

#Download rds files

#get import data

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/imp\_data\_final.rds?raw=true",destfile=tmp,mode = 'wb')

imp\_data\_final<-read\_rds(tmp)

file.remove(tmp)

imp\_data\_final

#get export data

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/exp\_data\_final.rds?raw=true",destfile=tmp,mode = 'wb')

exp\_data\_final<-read\_rds(tmp)

file.remove(tmp)

exp\_data\_final

#Get export data

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/exp\_data.rds?raw=true",destfile=tmp,mode = 'wb')

exp\_data<-read\_rds(tmp)

file.remove(tmp)

#Get inflation data

#Download rds files

#get import data

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/imp\_data\_final.rds?raw=true",destfile=tmp,mode = 'wb')

imp\_data\_final<-read\_rds(tmp)

file.remove(tmp)

imp\_data\_final

#get export data

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/exp\_data\_final.rds?raw=true",destfile=tmp,mode = 'wb')

exp\_data\_final<-read\_rds(tmp)

file.remove(tmp)

exp\_data\_final

#get export data

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/exp\_data.rds?raw=true",destfile=tmp,mode = 'wb')

exp\_data<-read\_rds(tmp)

file.remove(tmp)

#get inflation data

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/Consumer\_Inflation.rds?raw=true",destfile=tmp,mode = 'wb')

Inflation\_data<-read\_rds(tmp)

file.remove(tmp)

#Get $ data

tmp<-tempfile(fileext=".rds")

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Rds/US\_Dollar\_Montly\_Rate.rds?raw=true",destfile=tmp,mode = 'wb')

US\_Dollar\_data<-read\_rds(tmp)

file.remove(tmp)

#US\_Dollar\_Montly\_Rate

#Download Raw Data

# Create a temporary file

tmp<-tempfile(fileext=".xlsx")

# Download file from repository to the temp file

download.file("https://github.com/MEF-BDA503/gpj18-r\_coders/blob/master/Data\_Sources\_Excel/US\_Dollar\_Montly\_Rate.xlsx?raw=true",mode="wb",destfile=tmp)

# Read that excel file using readxl package's read\_excel function. You might need to adjust the parameters (skip, col\_names) according to your raw file's format.

raw\_data<-readxl::read\_excel(tmp,skip=7,col\_names=FALSE)

# Remove the temp file

file.remove(tmp)

colnames(raw\_data) <- c("Date","Dollar")

US\_Dollar\_Montly\_Rate<- raw\_data

saveRDS(US\_Dollar\_Montly\_Rate, file = "US\_Dollar\_Montly\_Rate.rds")

US\_Dollar\_Montly\_Rate

colnames(imp\_data)[which(colnames(imp\_data) %in% c("Date") )] <- c("Import\_Date")

colnames(exp\_data)[which(colnames(exp\_data) %in% c("Date") )] <- c("Export\_Date")

(imp\_data)

head(exp\_data)

Inflation\_data

US\_Dollar\_data

imp\_data\_final

exp\_data\_final

Export\_Import\_union\_sektor\_data

#a nes column type

imp\_data\_final<- mutate(imp\_data\_final,Type="Import")

exp\_data\_final<- mutate(exp\_data\_final,Type="Export")

Export\_Import\_union\_data <- rbind.fill(imp\_data\_final,exp\_data\_final)

print.data.frame(Export\_Import\_union\_sektor\_data)

#change column name as amount

Export\_Total\_Amount

names(Export\_Import\_union\_data)[names(Export\_Import\_union\_data) == "Export\_Total\_Amount"] <- "Total\_Amount"

names(imp\_data\_final)[names(imp\_data\_final) == "Export\_Total\_Amount"] <- "Total\_Amount"

names(exp\_data\_final)[names(exp\_data\_final) == "Export\_Total\_Amount"] <- "Total\_Amount"

names(imp\_data\_final)[names(imp\_data\_final) == "Date"] <- "datadate"

names(exp\_data\_final)[names(exp\_data\_final) == "Date"] <- "datadate"

names(Inflation\_data)[names(Inflation\_data) == "Consumer\_Price\_Index\_Montly\_Change\_%"] <- "Consumer\_Price\_Index"

names(Inflation\_data)[names(Inflation\_data) == "Consumer\_Price\_Index\_Yearly\_Change\_%"] <- "Consumer\_Price\_Index\_Yearly\_Change"

Export\_Import\_union\_data

imp\_data\_final

exp\_data\_final

str(Inflation\_data)

Inflation\_data

## UI Part ##

ui <- navbarPage("R Coders",

tabPanel("Import/Export Main Analysis",

sidebarLayout(position = "left",

sidebarPanel("Compare Values",

checkboxInput("donum1", "Export", value = T),

checkboxInput("donum2", "Import", value = F),

checkboxInput("donum3", "Inflatıon", value = F),

sliderInput("wt1","Weight 1",min=1,max=10,value=1),

sliderInput("wt2","Weight 2",min=1,max=10,value=1),

sliderInput("wt3","Weight 3",min=1,max=10,value=1)

),

mainPanel((plotOutput(outputId="plotgraph", width="900",height="600px")),

h4("A country's current account falls into a deficit when imports of goods and services are larger than exports, so given the tendencies outlined in the above chart, it's of little surprise that the current account shifts to a surplus soon after major currency devaluation. Imports have suddenly got a lot more expensive.

The plunge in Turkey's lira and subsequent reversal of last year's credit boom has had a similar effect of grinding domestic economic activity.

Because of all these effects inflation rate is simultaneously increasing."

)))

),

navbarMenu("Product Analysis",tabPanel("Export- Product",mainPanel(

tabsetPanel(

tabPanel("Plot", plotOutput("distPlot"),h6("Episode IV", align = "center"),

h4("Rebel spaceships, striking", align = "center"),

h3("from a hidden base, have won", align = "center"))

)

)),

tabPanel("Import - Product",mainPanel(

tabsetPanel(

tabPanel("Plot", plotOutput("distPlot\_1"),h6("Episode IV", align = "center"),

h4("Rebel spaceships, striking", align = "center"),

h3("from a hidden base, have won", align = "center")),

tabPanel("Summary", verbatimTextOutput("selected\_var"),verbatimTextOutput("summary")),

tabPanel("Table", tableOutput("table"))

)

))

),

tabPanel("Import/Export Change Over Time",mainPanel(tabsetPanel(tabPanel("Plot1",plotOutput("importExportPlot")),

tabPanel("Plot2",plotOutput("ExpoloratoryPlot"),

h4("The initial impact crops up in the import data as consumers' buying power contracts, bringing economic output down with it. Following the depreciation at time t, imports fall sharply and export volumes increase as local goods become more competitively priced than those denominated in stronger currencies",align = "center")),

tabPanel("Plot3",plotOutput("UsdRatePlot"))))),

navbarMenu("More",

tabPanel("Export- Details",plotOutput("pieChart"),tags$div(class="header", checked=NA,

list(

tags$p("Ready to take the Shiny tutorial? If so"),

tags$a(href="shiny.rstudio.com/tutorial", "Click Here!"),

"Thank you",

tags$li("1.eleman"),

tags$li("2.eleman")

))),

tabPanel("Import- Details",plotOutput("pieChart1"),tags$div(class="header", checked=NA,

list(

tags$p("Ready to take the Shiny tutorial? If so"),

tags$a(href="shiny.rstudio.com/tutorial", "Click Here!"),

"Thank you",

tags$li("1.eleman"),

tags$li("2.eleman")

))))

)

## Server Part ##

server <- function(input, output) {

output$distPlot <- renderPlot({

ggplot(exp\_data\_v2,aes(x=exp\_data\_v2$Sector\_Type\_Code,y=exp\_data\_v2$Total\_Amount,color = exp\_data\_v2$Sector\_Type\_Code))+geom\_point()+theme(axis.text.x = element\_text(angle = 60, hjust = 1))

})

output$distPlot\_1 <- renderPlot({

ggplot(exp\_data\_v2,aes(x=exp\_data\_v2$Sector\_Type\_Code,y=exp\_data\_v2$Total\_Amount,color = exp\_data\_v2$Sector\_Type\_Code))+geom\_point()+theme(axis.text.x = element\_text(angle = 60, hjust = 1))

})

output$importExportPlot <- renderPlot({

ggplot(imp\_and\_exp\_data\_bymonth,

aes(x=Date,

y=Amount,

color=Type)) +

geom\_line()+

scale\_size\_area("Export Amount") +

xlab("Import/Export Date") +

ylab("Amount(1000$)") +

ggtitle("Import & Export Amount")

})

output$ExpoloratoryPlot <- renderPlot({

ggplot(exp\_data\_final\_2,aes(x=USD\_Rate, y = Export\_Total\_Amount, size = Consumer\_Price\_Index\_Yearly\_Change, color=Export\_Year)) +

geom\_point() +

scale\_x\_log10() +

theme\_bw()+

scale\_size\_area("Export Amount") +

xlab("USD Rate") +

ylab("Export Amount(1000$)") +

ggtitle("Export Amounts and Consumer Price Index")

})

output$pieChart <- renderPlot({

ggplot(df, aes(x="", y=share, fill=brand)) + geom\_bar(stat="identity", width=1)

})

output$pieChart1 <- renderPlot({

ggplot(df, aes(x="", y=share, fill=brand)) + geom\_bar(stat="identity", width=1)

})

##pie = ggplot(df, aes(x="", y=share, fill=brand)) + geom\_bar(stat="identity", width=1)

output$UsdRatePlot <- renderPlot({

ggplot(imd\_data\_final\_2,aes(x = USD\_Rate, y = Import\_Total\_Amount, size = Consumer\_Price\_Index\_Yearly\_Change, color=Import\_Year)) +

geom\_point() +

scale\_x\_log10() +

theme\_bw()+

scale\_size\_area("Import Amount") +

xlab("USD Rate") +

ylab("Import Amount(1000$)") +

ggtitle("Import Amounts and Consumer Price Index")

})

output$selected\_var <- renderText({

paste("You have selected",input$Number)

})

output$table <- renderTable({

head(import\_data %>% select(Sector\_Name,Sector\_Type\_Code), 100) %>% filter(!is.na(Sector\_Name) & Sector\_Name != 'Toplam -Total') %>% distinct()

})

output$summary <- renderPrint({

dataset <- import\_data %>% select(Sector\_Name,Sector\_Type\_Code)

summary(dataset)

})

output$table\_import <- renderTable({

head(import\_data %>% select(Sector\_Name) %>% mutate(VADiff = import\_data$January + import\_data$February + import\_data$March ) %>% filter(is.na(VADiff)) %>% distinct() %>% filter(!(is.na(Sector\_Name))), 10)

})

output$table\_export <- renderTable({

head(import\_data %>% select(Sector\_Name) %>% mutate(VADiff = import\_data$January + import\_data$February) %>% filter(VADiff>1000000 & Sector\_Name != 'Toplam -Total') %>% distinct(Sector\_Name), 10)

})

set.seed(600)

pt1 <- reactive({

if (!input$donum1) return(NULL)

qplot(datadate, Total\_Amount, data=exp\_data\_final, geom="area",fill=I("lightblue"),binwidth=0.2,main="Export Trend By Time",xlab="Date", ylab='Amount')

})

pt2 <- reactive({

if (!input$donum2) return(NULL)

qplot(datadate, Total\_Amount, data=imp\_data\_final, geom="area",fill=I("red"),binwidth=0.2,main="Export Trend By Time",xlab="Date", ylab='Amount')

})

pt3 <- reactive({

if (!input$donum3) return(NULL)

qplot(Date, Consumer\_Price\_Index\_Yearly\_Change, data=Inflation\_data, geom="area",fill=I("darkblue"),binwidth=0.2,main="Inflation Trend By Time",xlab="Date", ylab='Consumer\_Price\_Index\_Yearly\_Change')

})

output$plotgraph = renderPlot({

ptlist <- list(pt1(),pt2(),pt3())

wtlist <- c(input$wt1,input$wt2,input$wt3)

# remove the null plots from ptlist and wtlist

to\_delete <- !sapply(ptlist,is.null)

ptlist <- ptlist[to\_delete]

wtlist <- wtlist[to\_delete]

if (length(ptlist)==0) return(NULL)

grid.arrange(grobs=ptlist,widths=wtlist,ncol=length(ptlist))

})

}

# Create Shiny app ----

shinyApp(ui, server)