Challange_4i

Felipe Freitas da Rocha

18/02/2021

1 Case 1

- Open the TFP.csv file attached.
- The series is composed by TFP (rtfpna variable) at constant national prices (2005 = 1) for three countries: United States (USA), Canada (CAN) and Mexico (MEX).

1. Make an exploratory data analysis;
2. Forecast 10 years of the series (if you are performing the exercise in R, use package “forecast”);
3. Check in the following link pages 2 and 3: https://cran.r-project.org/web/packages/pwt8/pwt8.pdf (https://cran.r-project.org/web/packages/pwt8/pwt8.pdf) to see a list of all variables in the original dataset. Can you think about another feature that could be helpful in explaining TFP series? Explain.

1.1 Load the Packages

```
library(tidyverse)
library(readxl)
library(forecast)
library(pwt8)
library(rmarkdown)
library(stats)
library(lmtest)
library(lubridate)
library(reshape2)
library(knitr)
library(kableExtra)
library(urca)
library(tsDyn)
library(vars)
```

1.2 Open the TFP.csv file attached

```
TFP <- read_csv("TFP.csv")

# we can also import the data by loading the pwt8 package:

  # TFP <- as_tibble(pwt8.0) %>%
  #        filter(isocode %in% c("USA", "CAN", "MEX")) %>%
  #        select(isocode,year,rtfpna) %>%
  #        arrange(isocode, year)
```

1.3 Make an exploratory data analysis

The TFP database presents 186 observations (rows) an 3 variables (columns). The first variable is called **isocode** and represents 3 countries: USA (62 observations), CAN (62 observations) and MEX (62 observations). The second corresponds to **year** (1950-2011). The third is called **rtfpna** and represents Total Factor Productivity (TFP) at constant national prices (2005 = 1). The function `summary`, `str` and `glimpse` summarize some information about the TFP database.

```
glimpse(TFP)
```

```
## Rows: 186
## Columns: 3
## $ isocode <chr> "USA", "USA", "USA", "USA", "USA", "USA", "USA", "USA", "US...
## $ year    <dbl> 1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959,...
## $ rtfpna  <dbl> 0.6171479, 0.6295884, 0.6384513, 0.6518582, 0.6461794, 0.66...
```

```
summary(TFP)
```

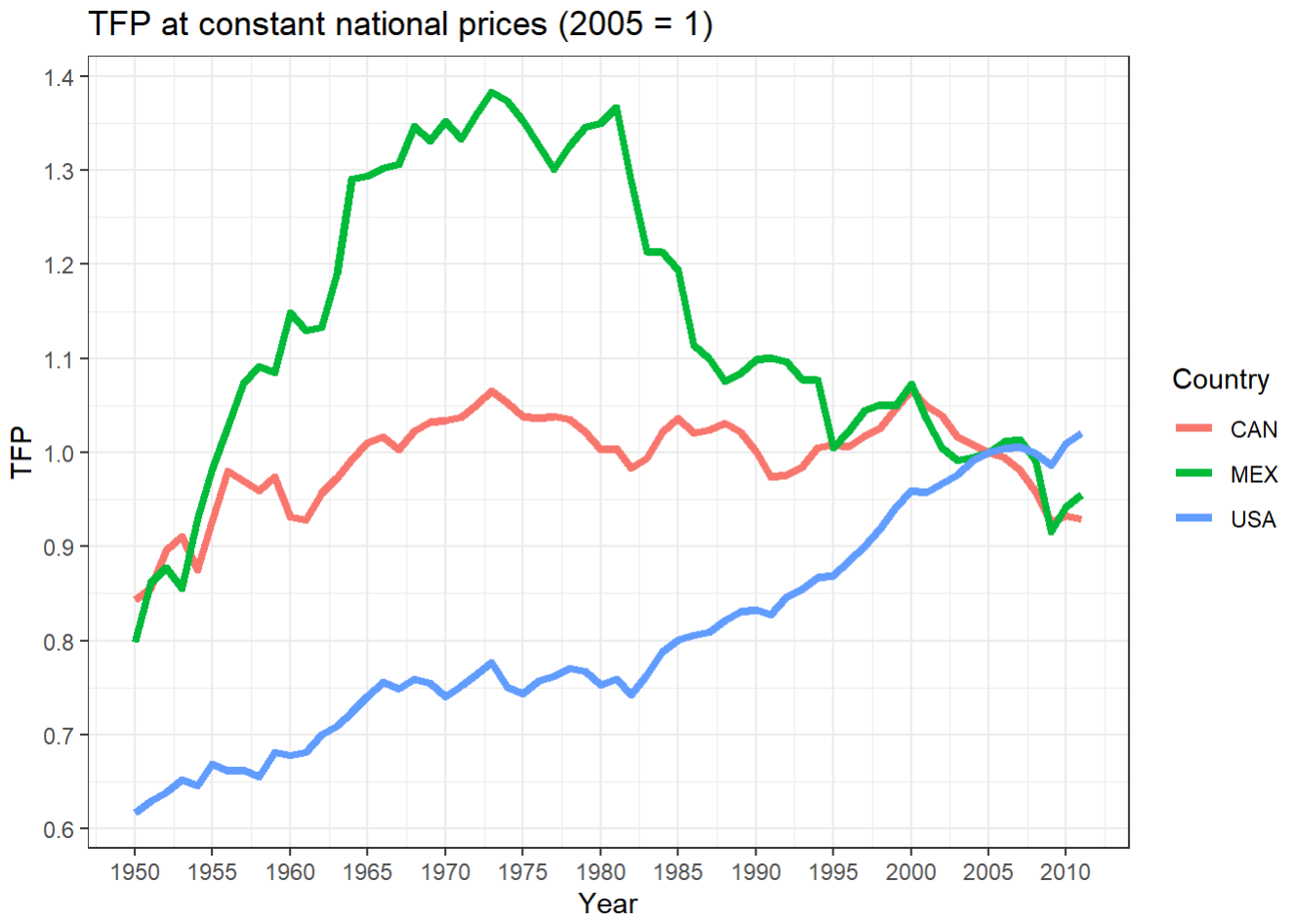
```
##    isocode               year          rtfpna      
##  Length:186         Min.   :1950   Min.   :0.6171  
##  Class :character   1st Qu.:1965   1st Qu.:0.8551  
##  Mode  :character   Median :1980   Median :0.9950  
##                     Mean   :1980   Mean   :0.9756  
##                     3rd Qu.:1996   3rd Qu.:1.0463  
##                     Max.   :2011   Max.   :1.3837
```

```
str(TFP)
```

```
## spec_tbl_df [186 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ isocode: chr [1:186] "USA" "USA" "USA" "USA" ...
##  $ year   : num [1:186] 1950 1951 1952 1953 1954 ...
##  $ rtfpna : num [1:186] 0.617 0.63 0.638 0.652 0.646 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   isocode = col_character(),
##   ..   year = col_double(),
##   ..   rtfpna = col_double()
##   .. )
```

Visual inspection of the data also helps us in the initial exploration. Using `ggplot` to plot the data, we have:

```
#Plotting the TFP data

ggplot(TFP) +
  aes(x = year, y = rtfpna, colour = isocode) +
  geom_line(size = 1.5) +
  scale_color_hue() +
  scale_x_continuous(breaks = seq(min(TFP$year),max(TFP$year),5)) +
  scale_y_continuous(breaks = seq(0.6,1.4,0.1)) +
  labs(x = "Year", y = "TFP", title = "TFP at constant national prices (2005 = 1)", color = "Cou
ntry") +
  theme_bw()
```



The graph above shows that, between 1950-2011, the United States (USA) showed considerable TFP growth, jumping from 0.62 in 1950 to 1.02 in 2011. On the other hand, in same period, Canada's TFP was relatively constant, ranging from 0.84 in 1950 to 0.93 in 2011. The Mexican case shows that the TFP grew rapidly between 1950 and the early 1970s, leaving 0.8 in 1950 to 1.38 in 1973. Since then, Mexico's TFP has been steadily falling, reaching 0.96 in 2011.

In addition, the graph also shows evidence that all three time series are non-stationary. In order to test whether the data are stationary, we will create the time series.

```
ts.USA <- TFP %>%                                  # data
  filter(isocode == "USA") %>%                     # filter the country
  dplyr::select(rtfpna) %>%                              # selecting TFP variable
  ts(start = min(TFP$year), end = max(TFP $ year))   # creating the time series

ts.MEX <- TFP %>%                                  # data
  filter(isocode == "MEX") %>%                     # filter the country
  dplyr::select(rtfpna) %>%                              # selecting TFP variable
  ts(start = min(TFP$year), end = max(TFP $ year))   # creating the time series

ts.CAN <- TFP %>%                                  # data
  filter(isocode == "CAN") %>%                     # filter the country
  dplyr::select(rtfpna) %>%                              # selecting TFP variable
  ts(start = min(TFP$year), end = max(TFP $ year))   # creating the time series
```

Using Augmented Dickey–Fuller (ADF) test in the `ndiffs` function, it is possible to see that it is necessary to take the first difference for the time series to become stationary.

```
ndiffs(ts.USA, test = "adf", type = "trend")   # adf_test - number of diff necessary to became s
tationary
```

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

```
ndiffs(ts.CAN, test = "adf", type = "trend")   # adf_test - number of diff necessary to became s
tationary
```

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

```
ndiffs(ts.MEX, test = "adf", type = "trend")   # adf_test - number of diff necessary to became s
tationary
```

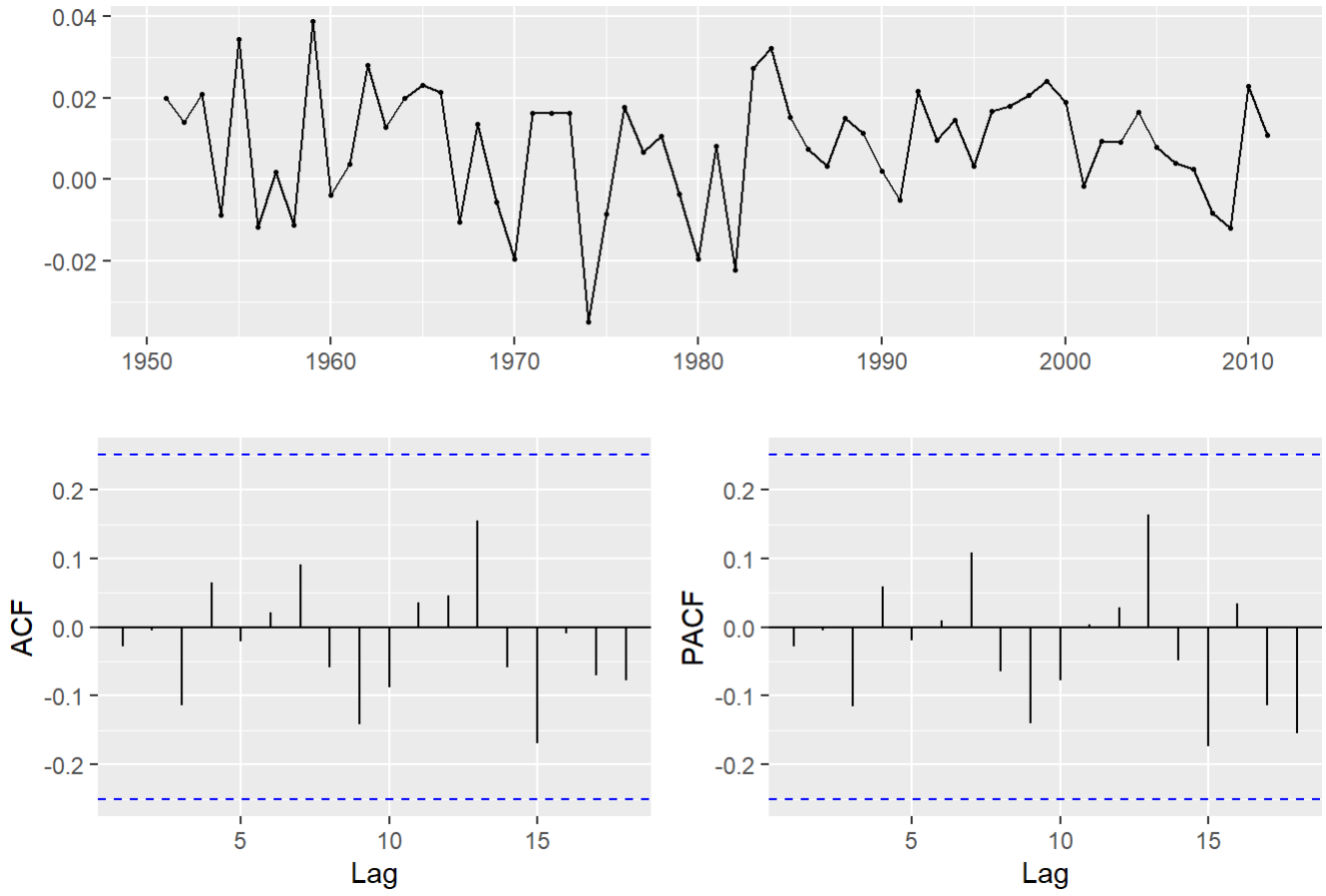
```
## [1] 1
```

1.3.1 First difference

By plotting the time series in `log` transformation and in first difference with their, respective, ACF and PACF, we see that the problem of non-stationary has been solved.

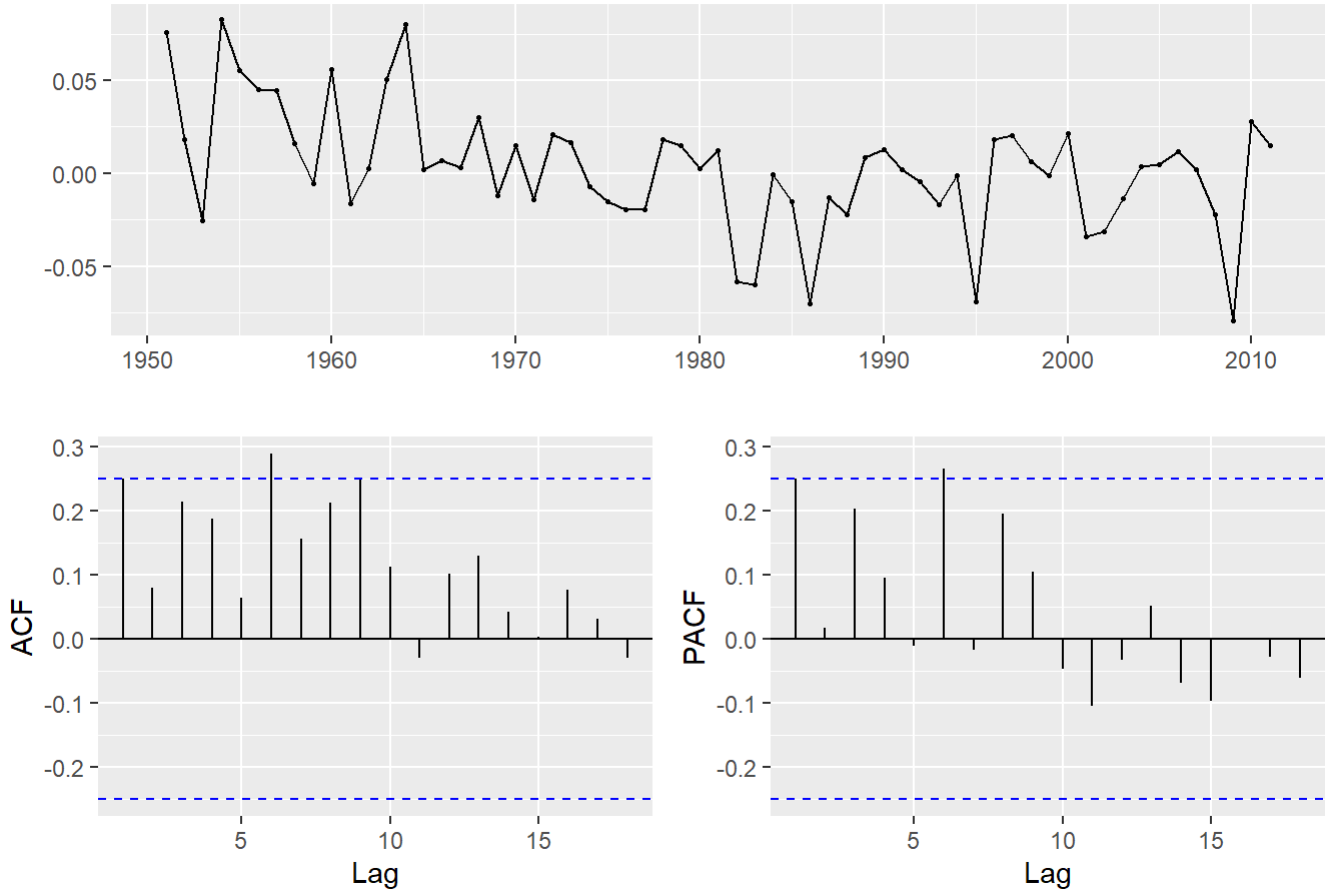
1.3.1.1 USA

```
ts.USA %>%        # data
  log() %>%       # log transformation
  diff() %>%      # first difference
  ggtsdisplay()   # plotting
```



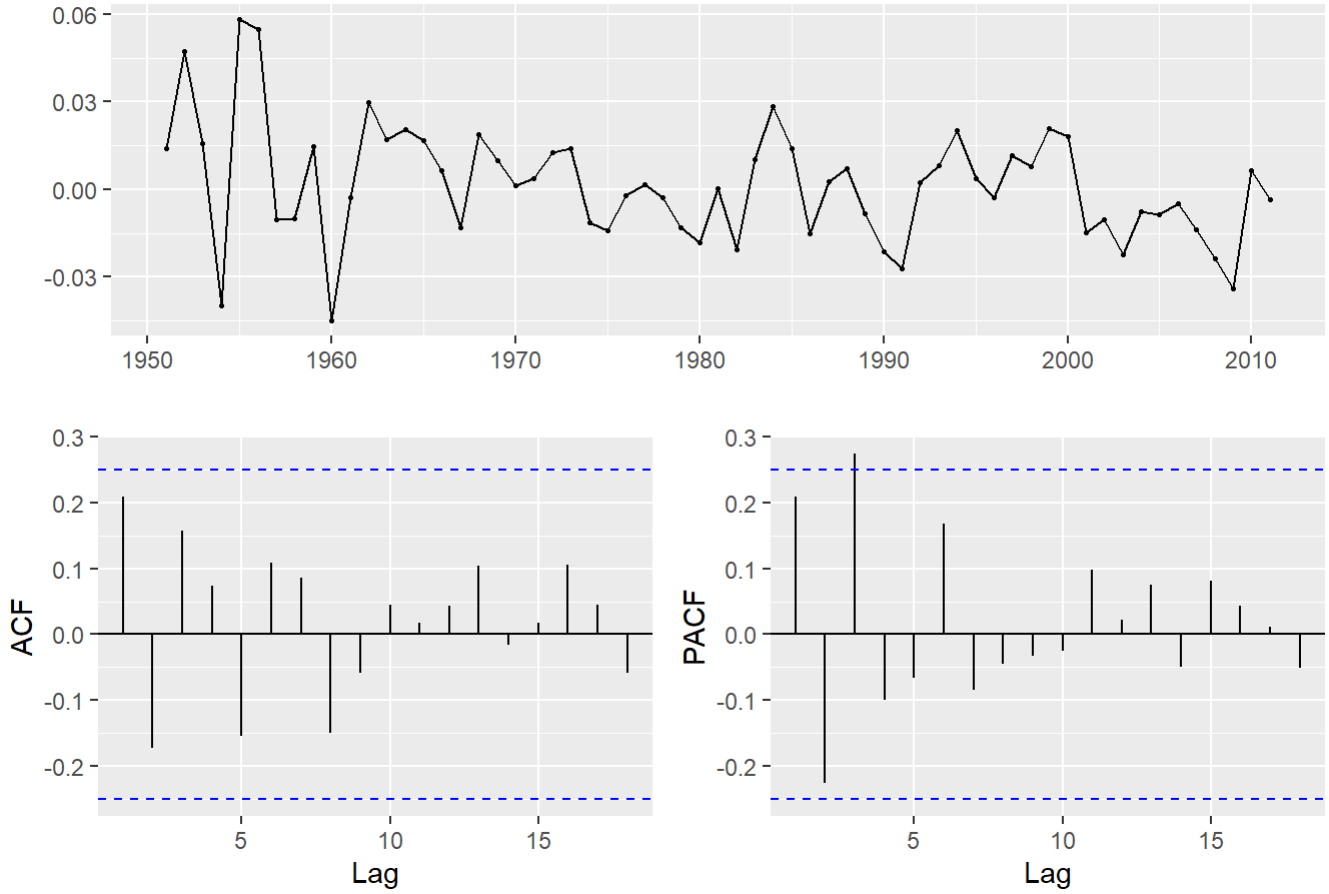
1.3.1.2 MEX

```
ts.MEX %>%          # data
  log() %>%         # log transformation
  diff() %>%        # first difference
  ggtsdisplay()     # plotting
```



1.3.1.3 CAN

```
ts.CAN %>%          # data
  log() %>%         # log transformation
  diff() %>%        # first difference
  ggtsdisplay()     # plotting
```



1.4 Forecast 10 years of the series (if you are performing the exercise in R, use package “forecast”)

1.4.1 Choosing the model

Before making the forecast we have to choose the ARIMA model that fits the data. To do this, we will use the function `auto.arima` , using the time series in `log` transformation (i.e. **lambda = 0**) and in first difference (i.e. **d=1**).

1.4.1.1 USA

In the case of USA, an ARIMA (0.1.0) with drift was selected. This indicates that the series **log(ts.USA)** is a Random Walk with drift. It should be noted that the choice of the model does not depend on the information criterion (i.e. both *aic* and *bic* choose the same model).

```
arima.bic.USA <- auto.arima(ts.USA,                    # data
                            d=1,                       # number of diff
                            ic = "bic",                # information criterion
                            lambda = 0,                # log transformation
                            seasonal = FALSE,          # without seasonality
                            stepwise = FALSE,
                            approximation = FALSE)
arima.bic.USA
```

```
## Series: ts.USA
## ARIMA(0,1,0) with drift
## Box Cox transformation: lambda= 0
##
## Coefficients:
##        drift
##       0.0082
## s.e.  0.0019
##
## sigma^2 estimated as 0.0002173:  log likelihood=171.19
## AIC=-338.38   AICc=-338.17   BIC=-334.16
```

```
arima.aic.USA <- auto.arima(ts.USA,                    # data
                            d=1,                       # number of diff
                            ic = "aic",                # information criterion
                            lambda = 0,                # log transformation
                            seasonal = FALSE,          # without seasonality
                            stepwise = FALSE,
                            approximation = FALSE)

arima.aic.USA
```

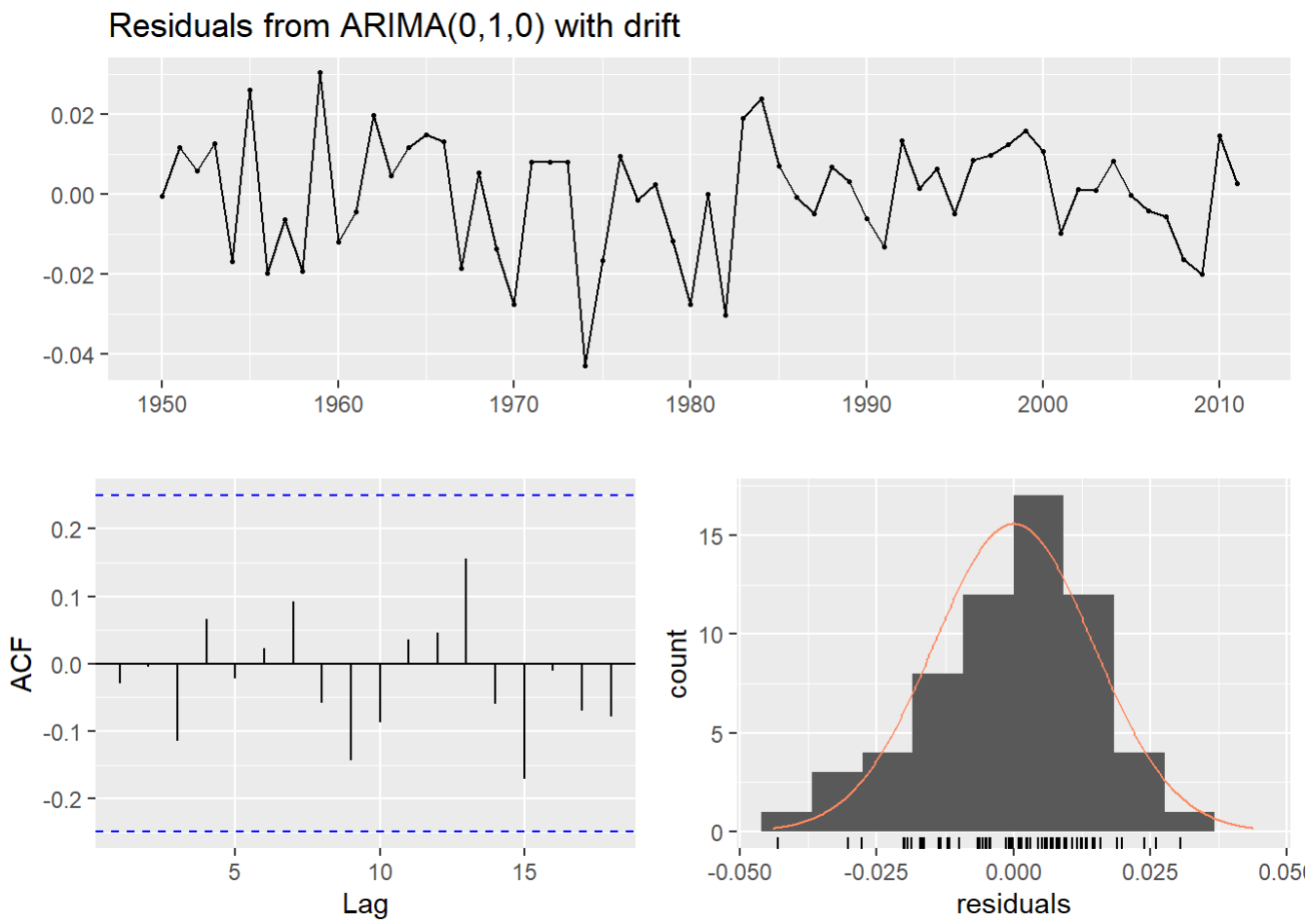
```
## Series: ts.USA
## ARIMA(0,1,0) with drift
## Box Cox transformation: lambda= 0
##
## Coefficients:
##        drift
##       0.0082
## s.e.  0.0019
##
## sigma^2 estimated as 0.0002173:  log likelihood=171.19
## AIC=-338.38   AICc=-338.17   BIC=-334.16
```

In addition, the estimated value for the drift is significant with 0.01 significance. We must check the residues using the `checkresiduals` function. There is no evidence of autocorrelation in the residuals and they have a white noise shape.

```
coeftest(arima.bic.USA)        # testing the coefficients
```

```
##
## z test of coefficients:
##
##        Estimate Std. Error z value  Pr(>|z|)
## drift 0.0082432  0.0018764  4.3931 1.117e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
checkresiduals(arima.bic.USA)  # checking the residuals
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0) with drift
## Q* = 4.2628, df = 9, p-value = 0.8933
##
## Model df: 1.   Total lags used: 10
```

1.4.1.2 MEX

In the case of MEX, the *bic* information criterion selects an ARIMA(1,1,0), while the *aic* selects an ARIMA(1,1,1). In both cases, there is no evidence of autocorrelation in the residuals and all parameters are significant.

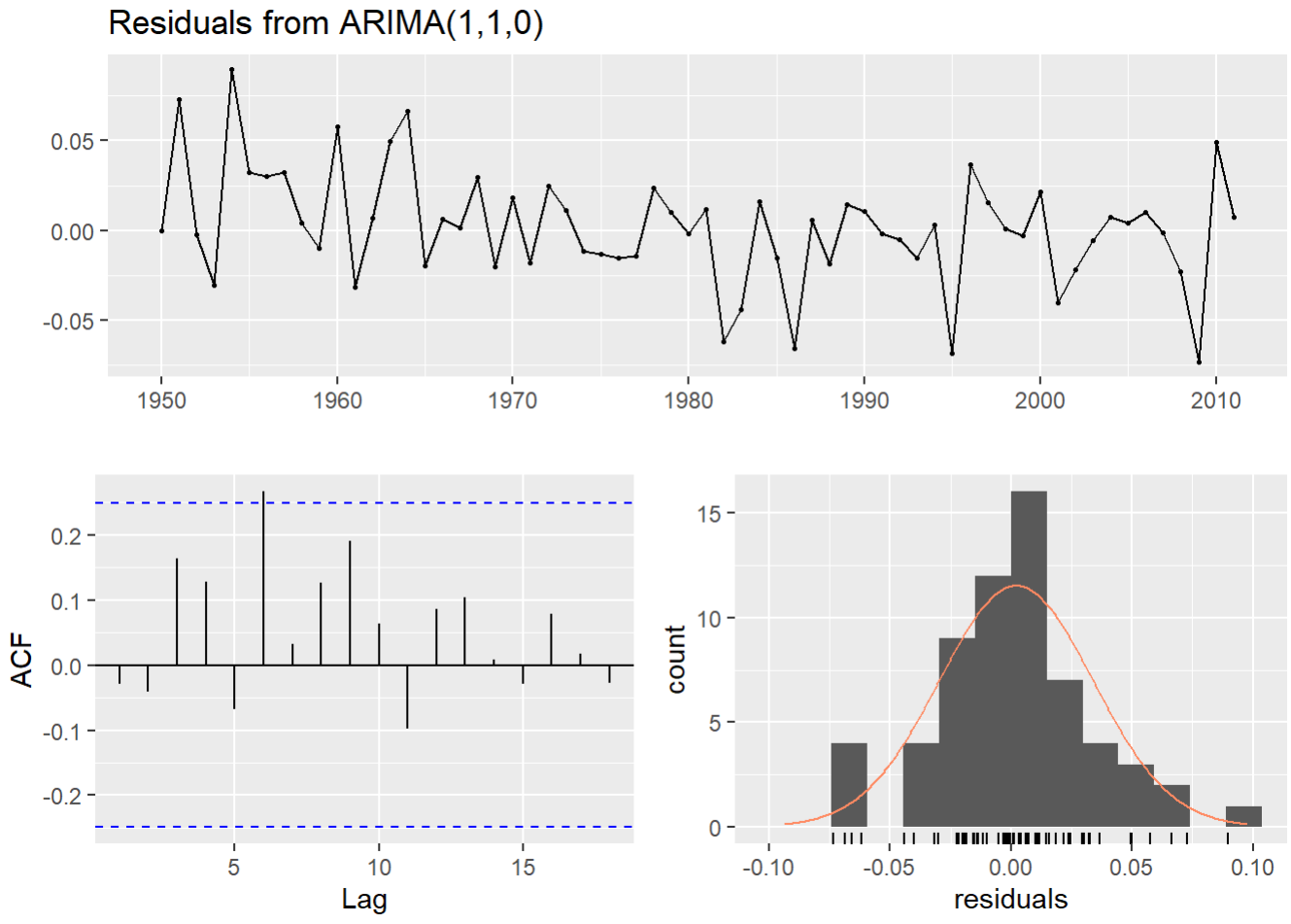
```
arima.bic.MEX <- auto.arima(ts.MEX,                    # data
                            d=1,                       # number of diff
                            ic = "bic",                # information criterion
                            lambda = 0,                # log transformation
                            seasonal = FALSE,          # without seasonality
                            stepwise = FALSE,
                            approximation = FALSE)

arima.bic.MEX
```

```
## Series: ts.MEX
## ARIMA(1,1,0)
## Box Cox transformation: lambda= 0
##
## Coefficients:
##          ar1
##       0.2727
## s.e.  0.1280
##
## sigma^2 estimated as 0.001038:  log likelihood=123.46
## AIC=-242.92   AICc=-242.72   BIC=-238.7
```

```
coeftest(arima.bic.MEX)        # testing the coefficients
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.27275    0.12797  2.1313  0.03307 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
checkresiduals(arima.bic.MEX)  # checking the residuals
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,0)
## Q* = 12.755, df = 9, p-value = 0.174
##
## Model df: 1.   Total lags used: 10
```

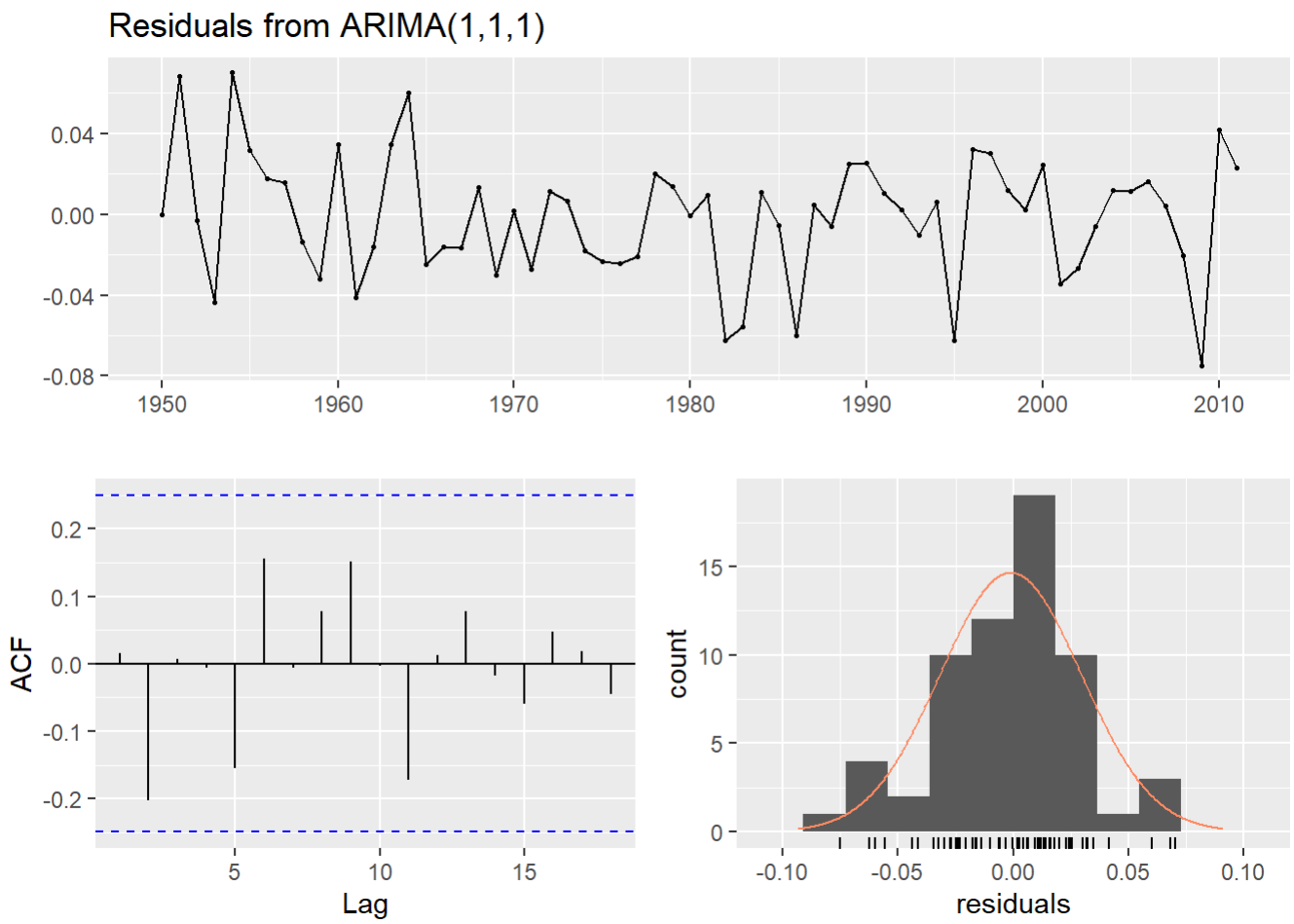
```
arima.aic.MEX <- auto.arima(ts.MEX,                    # data
                            d=1,                       # number of diff
                            ic = "aic",                # information criterion
                            lambda = 0,                # log transformation
                            seasonal = FALSE,          # without seasonality
                            stepwise = FALSE,
                            approximation = FALSE)

arima.aic.MEX
```

```
## Series: ts.MEX
## ARIMA(1,1,1)
## Box Cox transformation: lambda= 0
##
## Coefficients:
##          ar1      ma1
##       0.9622  -0.8308
## s.e.  0.0508   0.0836
##
## sigma^2 estimated as 0.000979:  log likelihood=125.52
## AIC=-245.03   AICc=-244.61   BIC=-238.7
```

```
coeftest(arima.aic.MEX)        # testing the coefficients
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ar1  0.962245   0.050806 18.9395 < 2.2e-16 ***
## ma1 -0.830823   0.083644 -9.9328 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
checkresiduals(arima.aic.MEX)  # checking the residuals
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)
## Q* = 8.2868, df = 8, p-value = 0.406
##
## Model df: 2.   Total lags used: 10
```

In order to compare these two models, we will create out-of-sample forecasts with 22 periods. We will estimate the models recursively to get the forecast one-step-ahead 22 times.

```
H <- 22 # number of one-step-ahead forecast

test.MEX <- window(ts.MEX,
                   start = max(TFP$year)-H+1,
                   end = max(TFP$year))  #test data / out-of-sample

#recursive estimation of the bic model

fosa.bic.MEX <- rep(0, NROW=(H))

for(i in 0:(H-1)) {fosa.bic.MEX[i] <- forecast(Arima(
                                                    window(ts.MEX, start = min(TFP$year), end =
max(TFP$year)-(H-i)), # data
                                                    order = arimaorder(arima.bic.MEX),
# order selected in auto.arima
                                                    lambda = 0),
# log transformation
                                                h = 1)$mean}
# mean = returning only the forecast
#one-step-ahead forecast for bic model

fosa.bic.MEX <- ts(fosa.bic.MEX,
                   start = max(TFP$year)-H+1,
                   end = max(TFP$year))
```

```
#number of one-step-ahead forecast - H = 22

#recursive estimation of the aic model

fosa.aic.MEX <- rep(0, NROW=(H))

for(i in 0:(H-1)) {fosa.aic.MEX[i] <- forecast(Arima(
                                                    window(ts.MEX, start = min(TFP$year), end =
max(TFP$year)-(H-i)), # data
                                                    order = arimaorder(arima.aic.MEX),
# order selected in auto.arima
                                                    lambda = 0),
# log transformation
                                                h = 1)$mean}
# mean = returning only the forecast
#one-step-ahead forecast for aic model

fosa.aic.MEX <- ts(fosa.aic.MEX,
                   start = max(TFP$year)-H+1,
                   end = max(TFP$year))
```

Now we can test which model has the least prediction error. We will use Diebold-Mariano test for predictive accuracy. The result of the test indicates that the models have the same levels of accuracy in the forecasts.

```
eosa.bic.MEX <- test.MEX-fosa.bic.MEX # prediction error - bic model
eosa.aic.MEX <- test.MEX-fosa.aic.MEX # prediction error - aic model

dm.test(eosa.aic.MEX, eosa.bic.MEX)  # Diebold-Mariano test
```

```
##
##  Diebold-Mariano Test
##
## data:  eosa.aic.MEXeosa.bic.MEX
## DM = -1.1407, Forecast horizon = 1, Loss function power = 2, p-value =
## 0.2668
## alternative hypothesis: two.sided
```

However, we will choose the model with the lowest Mean square prediction error (MSPE). In other words, ARIMA(1,1,1) selected by *aic*. It should be noted that we could also combine the forecasts of both models to carry out the forecast 10 years ahead.

```
MSPE.bic.MEX <- sum(eosa.bic.MEX^2)/H  # Mean square prediction error (MSPE) - bic model
MSPE.aic.MEX <- sum(eosa.aic.MEX^2)/H  # Mean square prediction error (MSPE) - aic model

MSPE.aic.MEX < MSPE.bic.MEX   # aic model is better than bic model
```

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

1.4.1.3 CAN

The case of CAN is very similar to the case of MEX. In the case of CAN, the *bic* information criterion selects an ARIMA(0,1,1), while the *aic* selects an ARIMA(2,1,2).In both cases, there is no evidence of autocorrelation in the residuals and all parameters are significant.

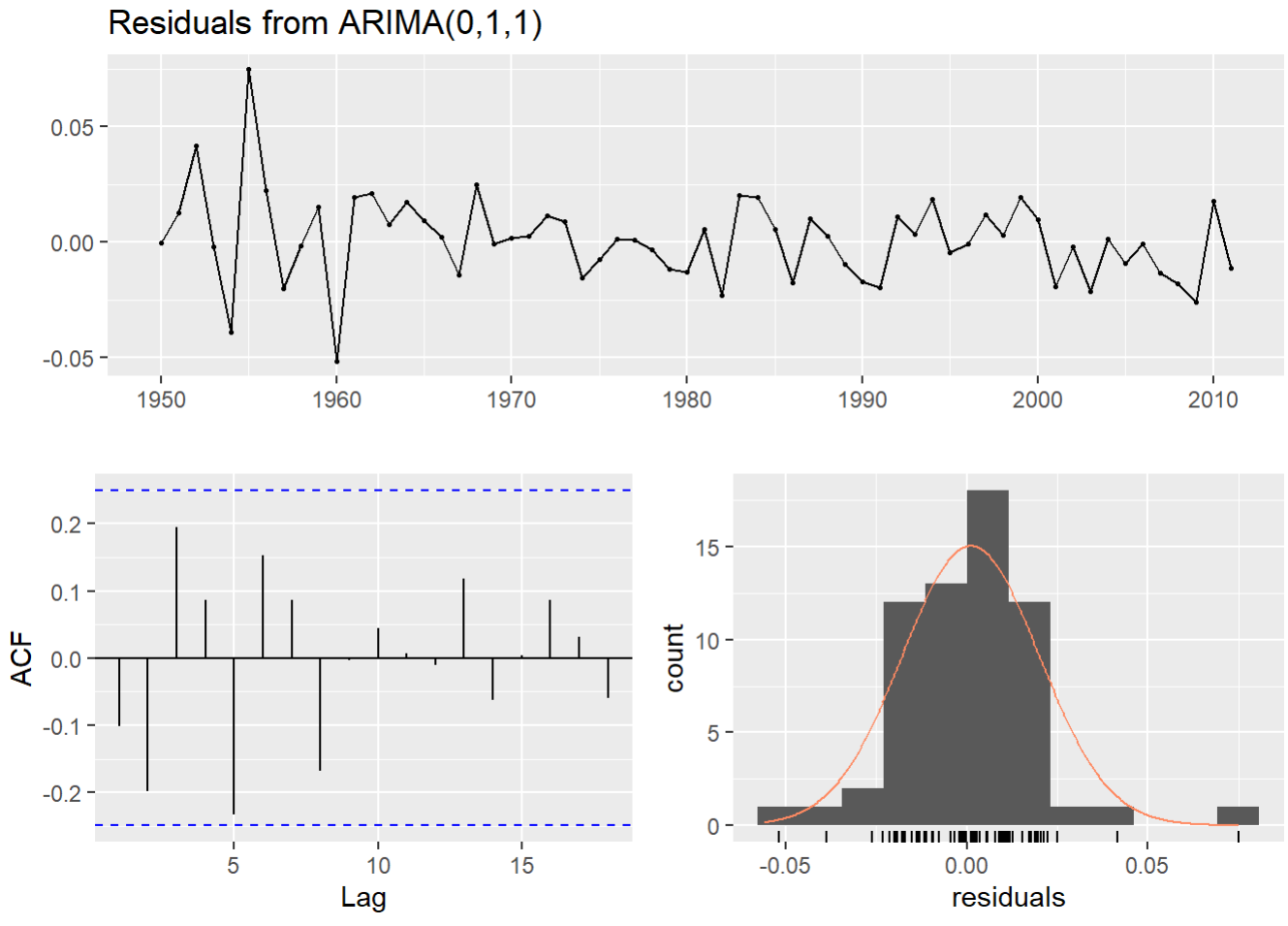
```
arima.bic.CAN <- auto.arima(ts.CAN,                    # data
                            d=1,                       # number of diff
                            ic = "bic",                # information criterion
                            lambda = 0,                # log transformation
                            seasonal = FALSE,          # without seasonality
                            stepwise = FALSE,
                            approximation = FALSE)

arima.bic.CAN
```

```
## Series: ts.CAN
## ARIMA(0,1,1)
## Box Cox transformation: lambda= 0
##
## Coefficients:
##          ma1
##       0.4330
## s.e.  0.1458
##
## sigma^2 estimated as 0.0003669:  log likelihood=155.12
## AIC=-306.23   AICc=-306.02   BIC=-302.01
```

```
coeftest(arima.bic.CAN)        # testing the coefficients
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ma1  0.43305    0.14581  2.9699 0.002979 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
checkresiduals(arima.bic.CAN)  # checking the residuals
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)
## Q* = 14.482, df = 9, p-value = 0.1062
##
## Model df: 1.   Total lags used: 10
```

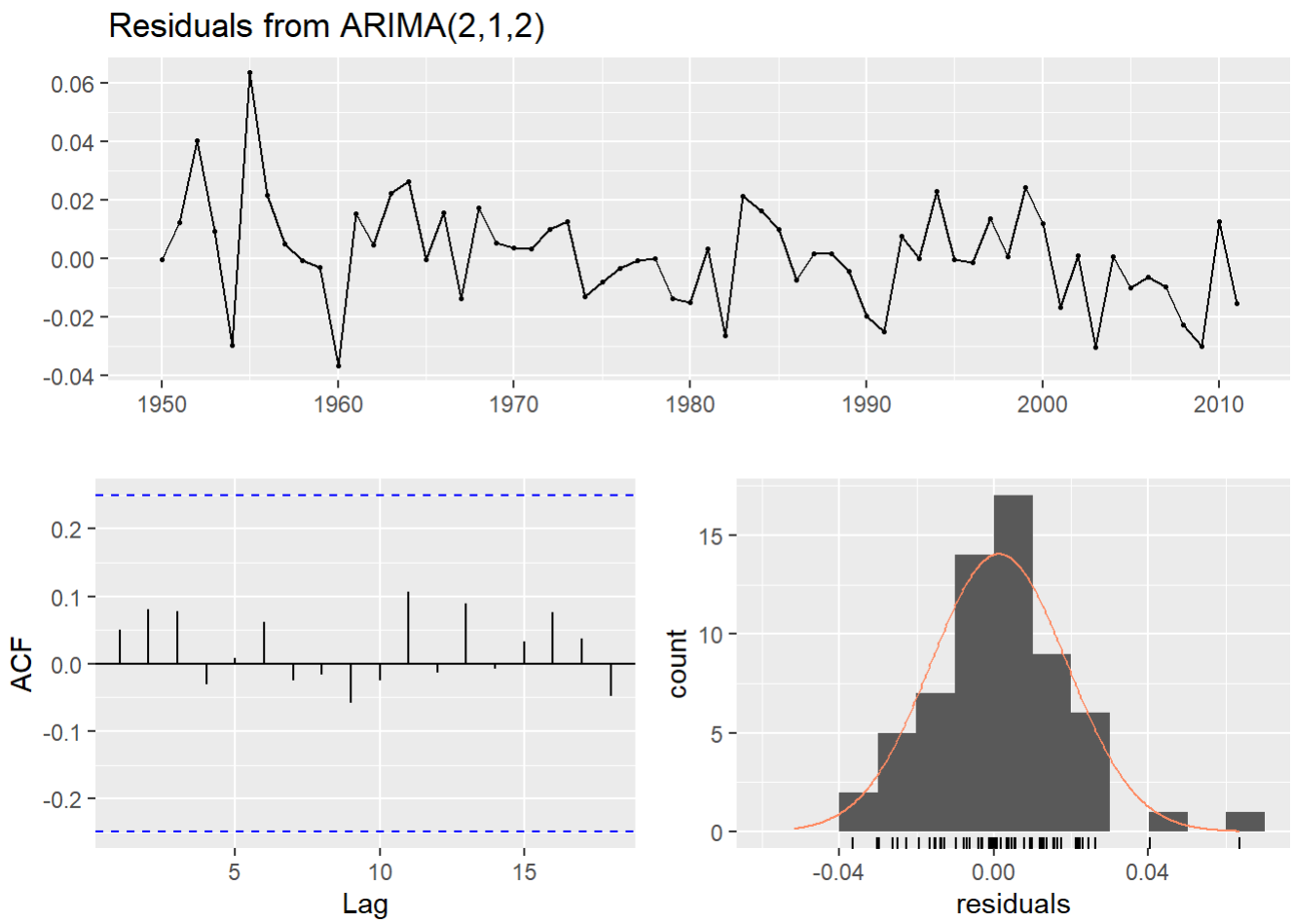
```
arima.aic.CAN <- auto.arima(ts.CAN,                    # data
                            d=1,                       # number of diff
                            ic = "aic",                # information criterion
                            lambda = 0,                # log transformation
                            seasonal = FALSE,          # without seasonality
                            stepwise = FALSE,
                            approximation = FALSE)

arima.aic.CAN
```

```
## Series: ts.CAN
## ARIMA(2,1,2)
## Box Cox transformation: lambda= 0
##
## Coefficients:
##           ar1      ar2     ma1     ma2
##       -0.7739  -0.6935  1.1092  0.6922
## s.e.   0.1992   0.2069  0.2294  0.1855
##
## sigma^2 estimated as 0.0003358:  log likelihood=159.1
## AIC=-308.19   AICc=-307.1   BIC=-297.64
```

```
coeftest(arima.aic.CAN)        # testing the coefficients
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ar1 -0.77390    0.19919 -3.8852 0.0001023 ***
## ar2 -0.69349    0.20693 -3.3514 0.0008041 ***
## ma1  1.10915    0.22940  4.8351 1.331e-06 ***
## ma2  0.69217    0.18555  3.7304 0.0001912 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
checkresiduals(arima.aic.CAN)  # checking the residuals
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,2)
## Q* = 1.7052, df = 6, p-value = 0.9447
##
## Model df: 4.   Total lags used: 10
```

As in MEX case, in order to compare these two models, we will create out-of-sample forecasts with 22 periods. We will estimate the models recursively to get the forecast one-step-ahead 22 times.

```
#number of one-step-ahead forecast - H = 22

#test data / out-of-sample

test.CAN <- window(ts.CAN,
                   start = max(TFP$year)-H+1,
                   end = max(TFP$year))

#recursive estimation of the bic model

fosa.bic.CAN <- rep(0, NROW=(H))

for(i in 0:(H-1)) {fosa.bic.CAN[i] <- forecast(Arima(
                                                    window(ts.CAN, start = min(TFP$year), end =
max(TFP$year)-(H-i)), # data
                                                    order = arimaorder(arima.bic.CAN),
# order selected in auto.arima
                                                    lambda = 0),
# log transformation
                                                h = 1)$mean}
# mean = returning only the forecast
#one-step-ahead forecast for bic model

fosa.bic.CAN <- ts(fosa.bic.CAN,
                   start = max(TFP$year)-H+1,
                   end = max(TFP$year))
```

```
#number of one-step-ahead forecast - H = 22

#recursive estimation of the aic model

fosa.aic.CAN <- rep(0, NROW=(H))

for(i in 0:(H-1)) {fosa.aic.CAN[i] <- forecast(Arima(
                                                    window(ts.CAN, start = min(TFP$year), end =
max(TFP$year)-(H-i)), # data
                                                    order = arimaorder(arima.aic.CAN),
# order selected in auto.arima
                                                    lambda = 0),
# log transformation
                                                h = 1)$mean}
# mean = returning only the forecast
#one-step-ahead forecast for aic model

fosa.aic.CAN <- ts(fosa.aic.CAN,
                   start = max(TFP$year)-H+1,
                   end = max(TFP$year))
```

Now we can test which model has the least prediction error. We will use Diebold-Mariano test for predictive accuracy. The result of the test indicates that the bic model is better than aic model. Therefore, we will choose the ARIMA(0,1,1) selected by *bic*.

```
eosa.bic.CAN <- test.CAN-fosa.bic.CAN # prediction error - bic model
eosa.aic.CAN <- test.CAN-fosa.aic.CAN # prediction error - aic model

MSPE.bic.CAN <- sum(eosa.bic.CAN^2)/H  # Mean square prediction error (MSPE) - bic model
MSPE.aic.CAN <- sum(eosa.aic.CAN^2)/H  # Mean square prediction error (MSPE) - aic model

MSPE.aic.CAN > MSPE.bic.CAN   # bic model is better than aic model
```

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

```
dm.test(eosa.aic.CAN, eosa.bic.CAN, alternative = "greater")  # Diebold-Mariano test confirms th
at bic model is better than aic model
```

```
##
##  Diebold-Mariano Test
##
## data:  eosa.aic.CANeosa.bic.CAN
## DM = 1.6942, Forecast horizon = 1, Loss function power = 2, p-value =
## 0.0525
## alternative hypothesis: greater
```

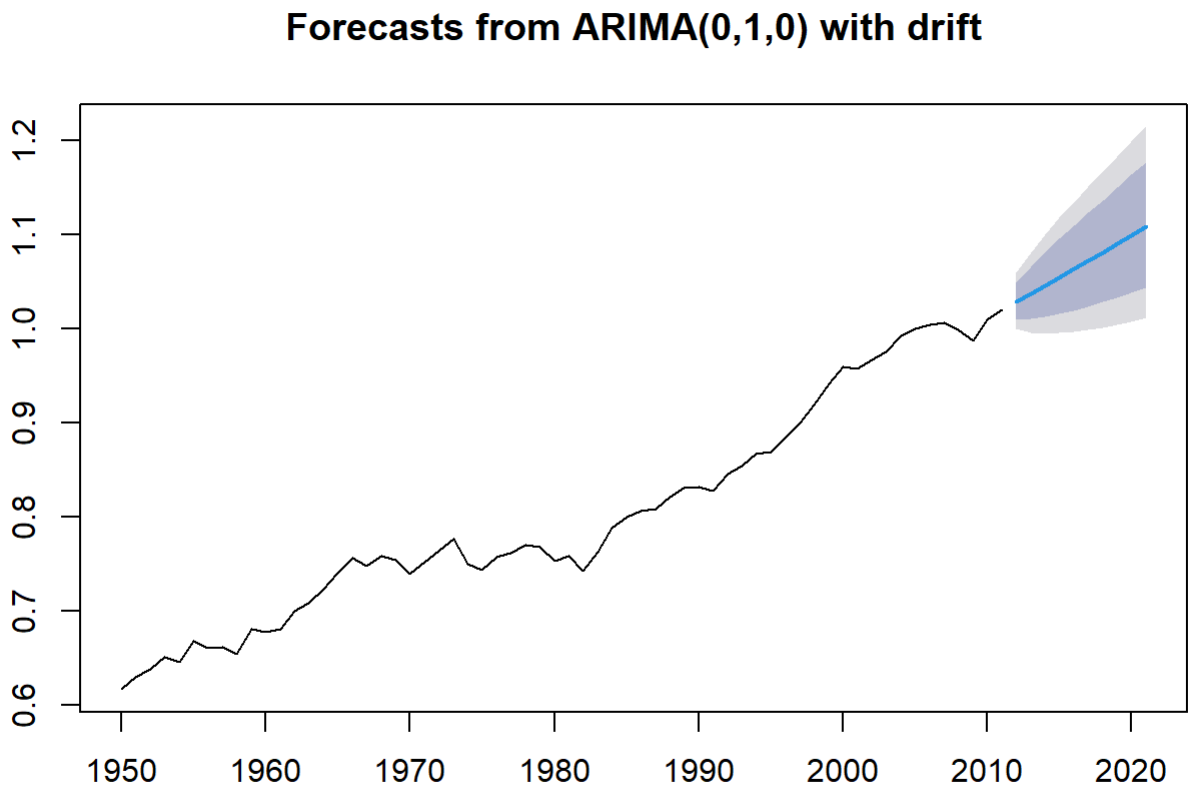
1.4.2 Forecast 10 years of the series

Selecting the number of years for the forecast

```
h <- 10 # numbers of years ahead for the forecast
```

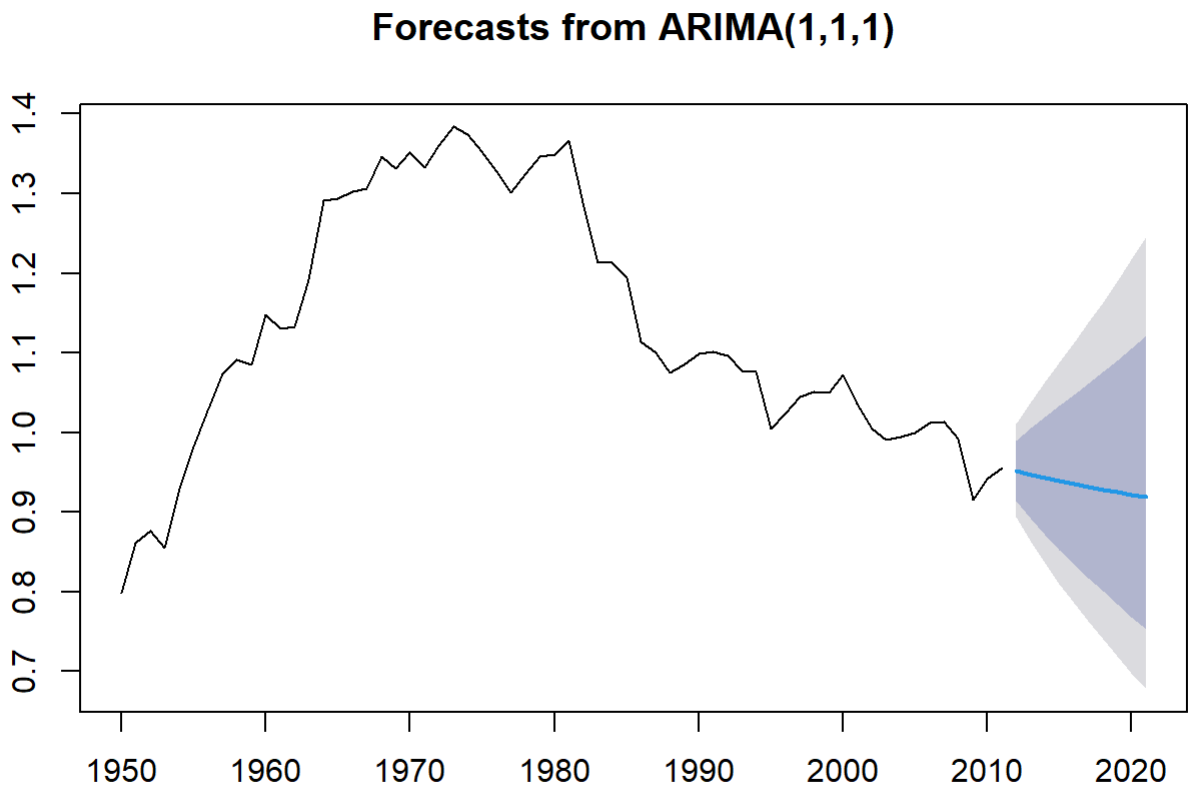
1.4.2.1 USA

```
forecast.USA <- forecast(arima.bic.USA, h=h) # forecast h years ahead
plot(forecast.USA)
```



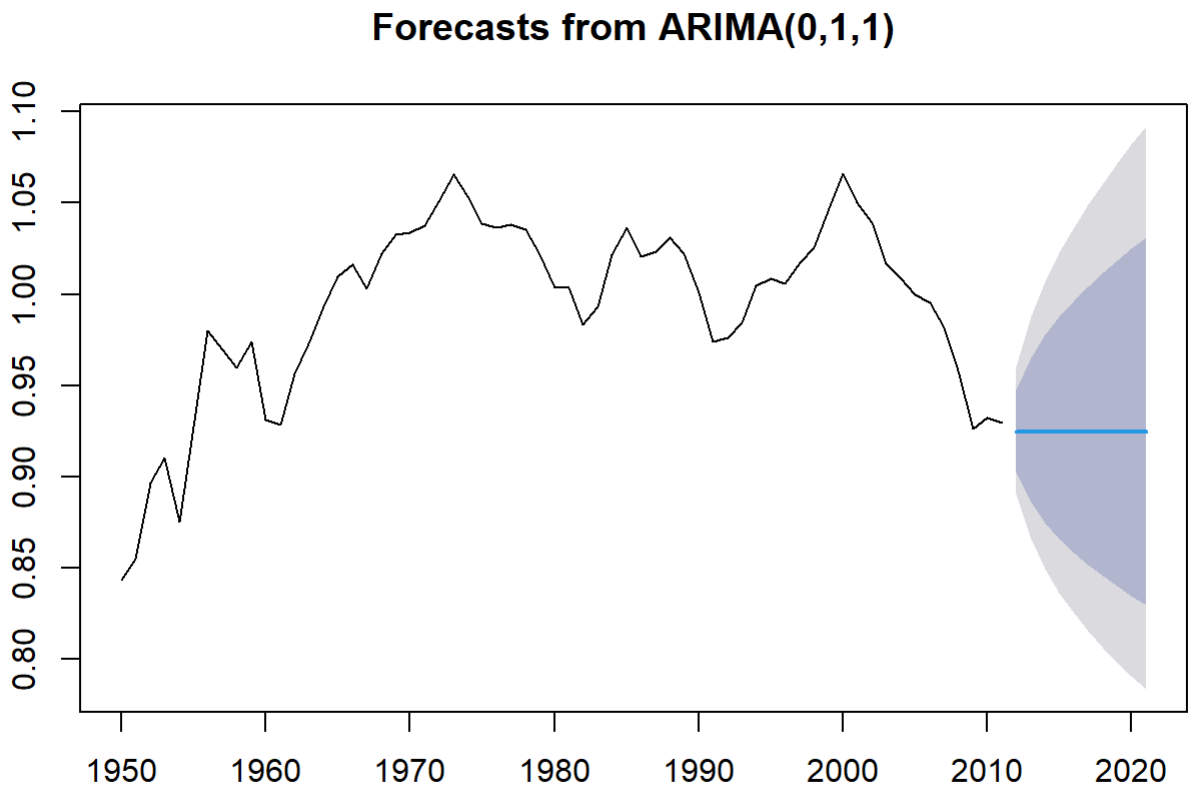
1.4.2.2 MEX

```
forecast.MEX <- forecast(arima.aic.MEX, h=h)  # forecast h years ahead
plot(forecast.MEX)
```



1.4.2.3 CAN

```
forecast.CAN <- forecast(arima.bic.CAN, h=h)  # forecast h years ahead
plot(forecast.CAN)
```



1.5 Can you think about another feature that could be helpful in explaining TFP series? Explain.

According to the neoclassical growth theory, the total factor productivity (TFP) is related to GDP, physical capital, human capital and the labor. Therefore, when analyzing the database **pwt8.0** (we can see information with the function `help(pwt8.0)`), we can infer that it is possible to use the variables **emp**, **hc**, **rgdpna**, **rkna**, together with the variable **rtfpna**, to better explain the TFP series. The econometric techniques are based on cointegration and the vector error correction model (VECM). When variables are cointegrated, there are error correction mechanisms that combine long-term equilibrium relationships with short-term adjustment dynamics.

The first step is to make a detailed analysis of the univariate properties of all time series. Cointegration requires two conditions: 1) there must be a set of variables of the same order of integration; and 2) their linear combination should result in a stationary series. When the time series are of different order of integration, there is certainly no relationship between them. On the other hand, if the time series present the same order of integration, then the cointegration test can be continued.

The second step is to perform any of the various cointegration tests, such as the Johansen test, the Engle-Granger test and the ECM test. If the variables are co-integrated, we proceed with the third step, which is to estimate a VECM. The fourth step is to perform different analyzes of the model and the TFP series, such as testing Granger's causality and obtaining the impulse response functions.

2 Case 2

- Attached to this test is a .csv file which contains data from Comexstat, which is basically the official data source for brazilian exports e imports, maintened by the government;
- The dataset contains all trackings of monthly imports and exports of a range of products (soybeans, soybean meal, soybean oil, corn, wheat and sugar), by brazilian states, by routes (air, sea, ground, etc) e from/to which country;
- We ask you to address a couple quentions below. Remember that data viz is one important skill to show besides any analytical skill. So we encourage you to use and explore a bunch of graphs and tables to show your point.

1. Show the evolution of total monthly and total annual exports from Brazil (all states and to everywhere) of ‘soybeans’, ‘soybean oil’ and ‘soybean meal’;
2. What are the 3 most important products exported by Brazil in the last 5 years?
3. What are the main routes through which Brazil have been exporting ‘corn’ in the last few years? Are there differences in the relative importance of routes depending on the product?
4. Which countries have been the most important trade partners for Brazil in terms of ‘corn’ and ‘sugar’ in the last 3 years?
5. For each of the products in the dataset, show the 5 most important states in terms of exports?
6. Now, we ask you to show your modelling skills. Feel free to use any type of modelling approach, but bear in mind that the modelling approach depends on the nature of your data, and so different models yield different estimates and forecasts. To help you out in this task we also provide you with a dataset of possible covariates (.xlsx). They all come from public sources (IMF, World Bank) and are presented in index number format. Question: What should be the total brazilian soybeans, soybean_meal, and corn export forecasts, in tons, for the next 11 years (2020-2030)? We’re mostly interested in the annual forecast.

2.1 Open the data_comexstat.csv file attached

We will import the data *data_comexstat.csv* and we will organize the date, separating the year and month.

```
data_cmst <- read_csv("data_comexstat.csv")       #import data

data_cmst <- data_cmst %>%                                              # data
                    mutate(year = year(date), month = month(date))   # separating the year and m
onth
```

2.2 Show the evolution of total monthly and total annual exports from Brazil of ‘soybeans’, ‘soybean oil’ and ‘soybean meal’.

2.2.1 Annual exports

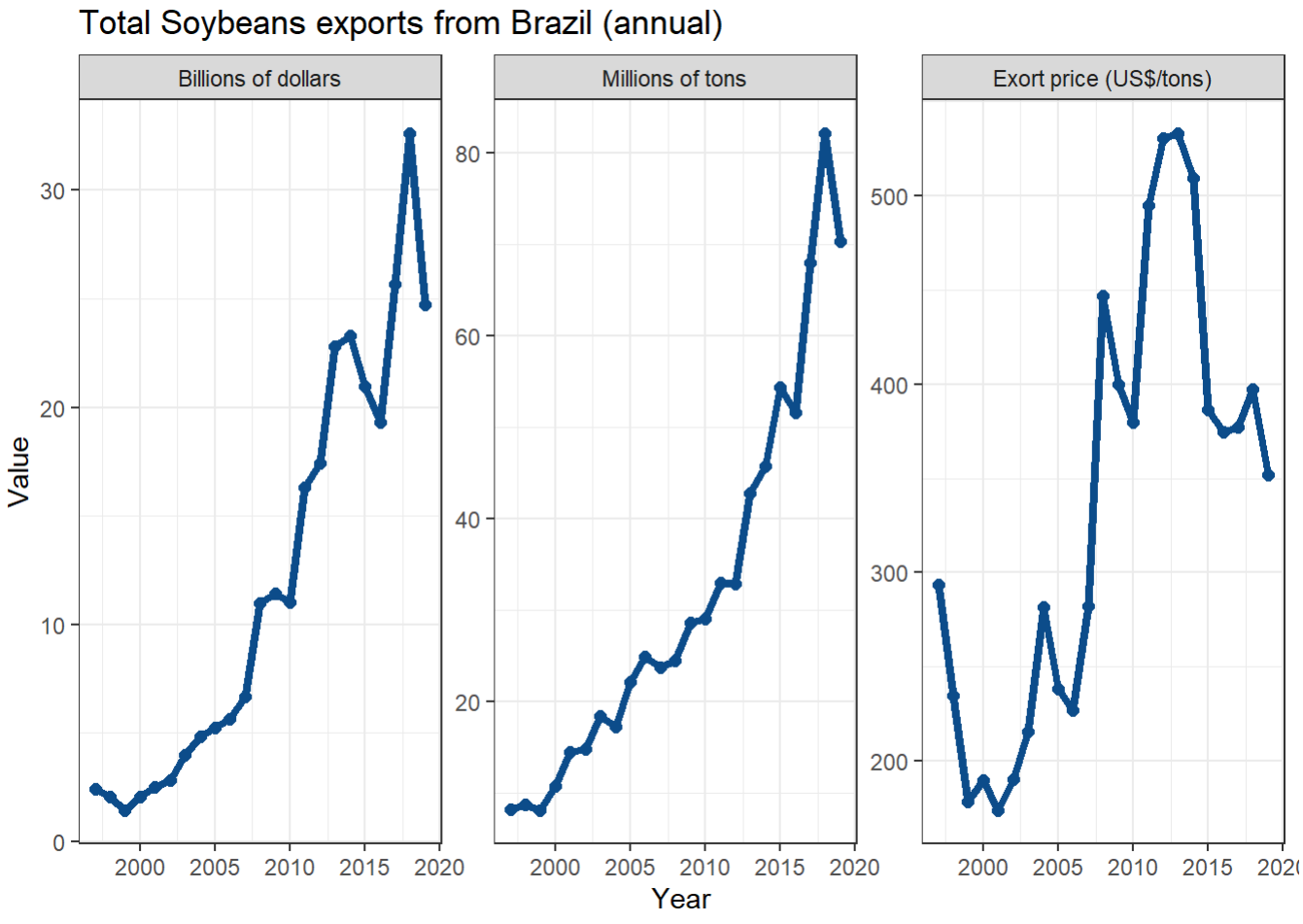
Summarizing the necessary information:

```
data_cmst.year <- data_cmst %>%   # data
  filter(type == "Export", product %in% c("soybeans", "soybean_oil", "soybean_meal")) %>%  # fil
ter export and products
  group_by(product,year) %>%
  summarise(total_export.usd = sum(usd)/10^9, total_export.tons = sum(tons)/10^6) %>%  # export
 b.usd and m.tons by product.year
  mutate(export.price = total_export.usd/total_export.tons*10^3) %>%    # export price
  rename("Billions of dollars" = total_export.usd, "Millions of tons" = total_export.tons, "Exor
t price (US$/tons)" = export.price) %>%
  melt(id.vars= c("product", "year")) %>%    # turning variables into observations
  as_tibble()
```

2.2.1.1 Soybeans

Plotting the data:

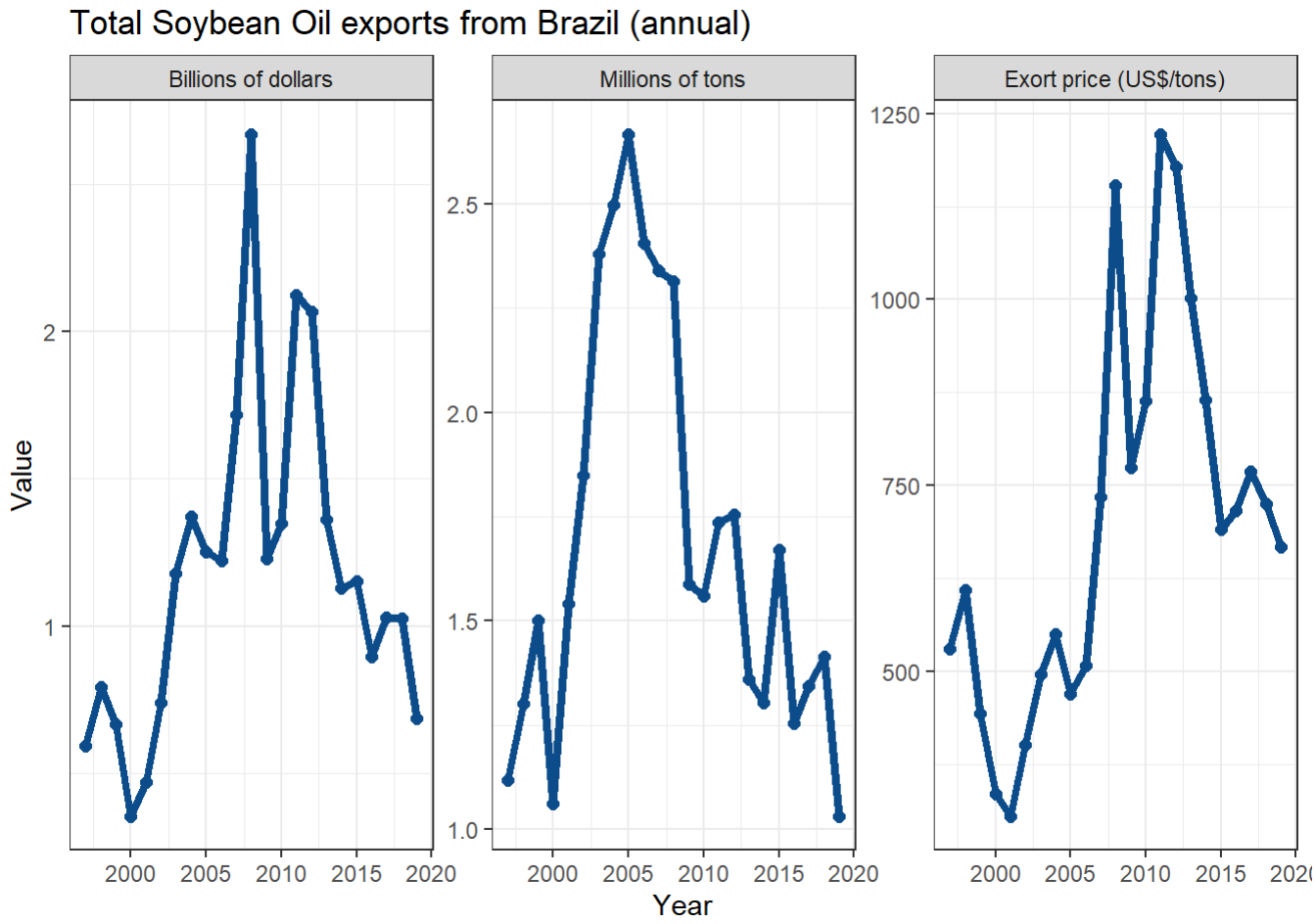
```
data_cmst.year %>%
  filter(product == "soybeans") %>%
  ggplot() +
    aes(x = year, y = value) +
    geom_line(size = 1.5, colour = "#0c4c8a") +
    geom_point(size = 2, colour = "#0c4c8a") +
    labs(x = "Year", y = "Value", title = "Total Soybeans exports from Brazil (annual)") +
    theme_bw() +
    facet_wrap(vars(variable), scales = "free_y")
```



2.2.1.2 Soybean oil

Plotting the data:

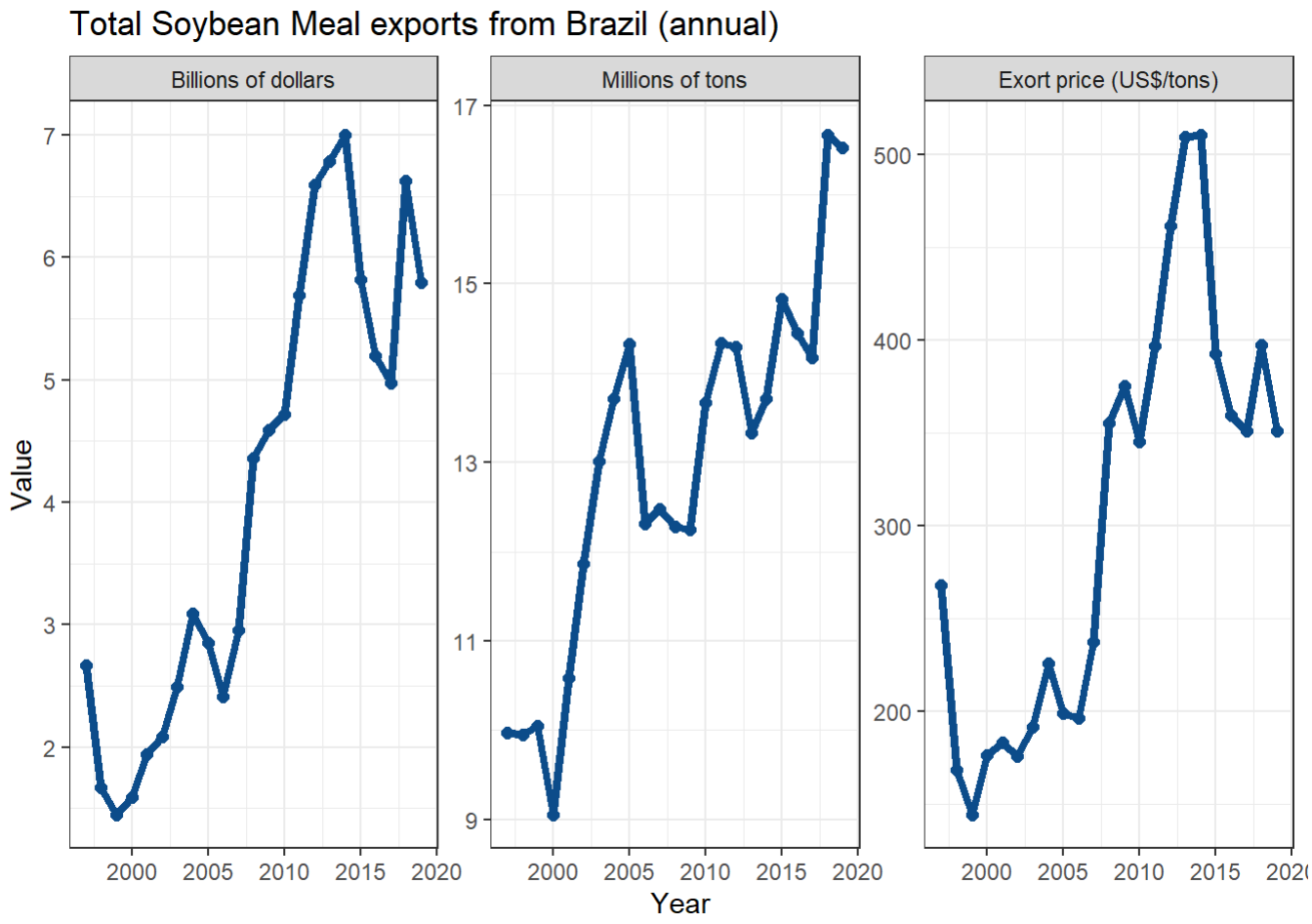
```
data_cmst.year %>%
  filter(product == "soybean_oil") %>%
  ggplot() +
    aes(x = year, y = value) +
    geom_line(size = 1.5, colour = "#0c4c8a") +
    geom_point(size = 2, colour = "#0c4c8a") +
    labs(x = "Year", y = "Value", title = "Total Soybean Oil exports from Brazil (annual)") +
    theme_bw() +
    facet_wrap(vars(variable), scales = "free_y")
```



2.2.1.3 Soybean meal

Plotting the data:

```
data_cmst.year %>%
  filter(product == "soybean_meal") %>%
  ggplot() +
    aes(x = year, y = value) +
    geom_line(size = 1.5, colour = "#0c4c8a") +
    geom_point(size = 2, colour = "#0c4c8a") +
    labs(x = "Year", y = "Value", title = "Total Soybean Meal exports from Brazil (annual)") +
    theme_bw() +
    facet_wrap(vars(variable), scales = "free_y")
```



2.2.2 Monthly exports

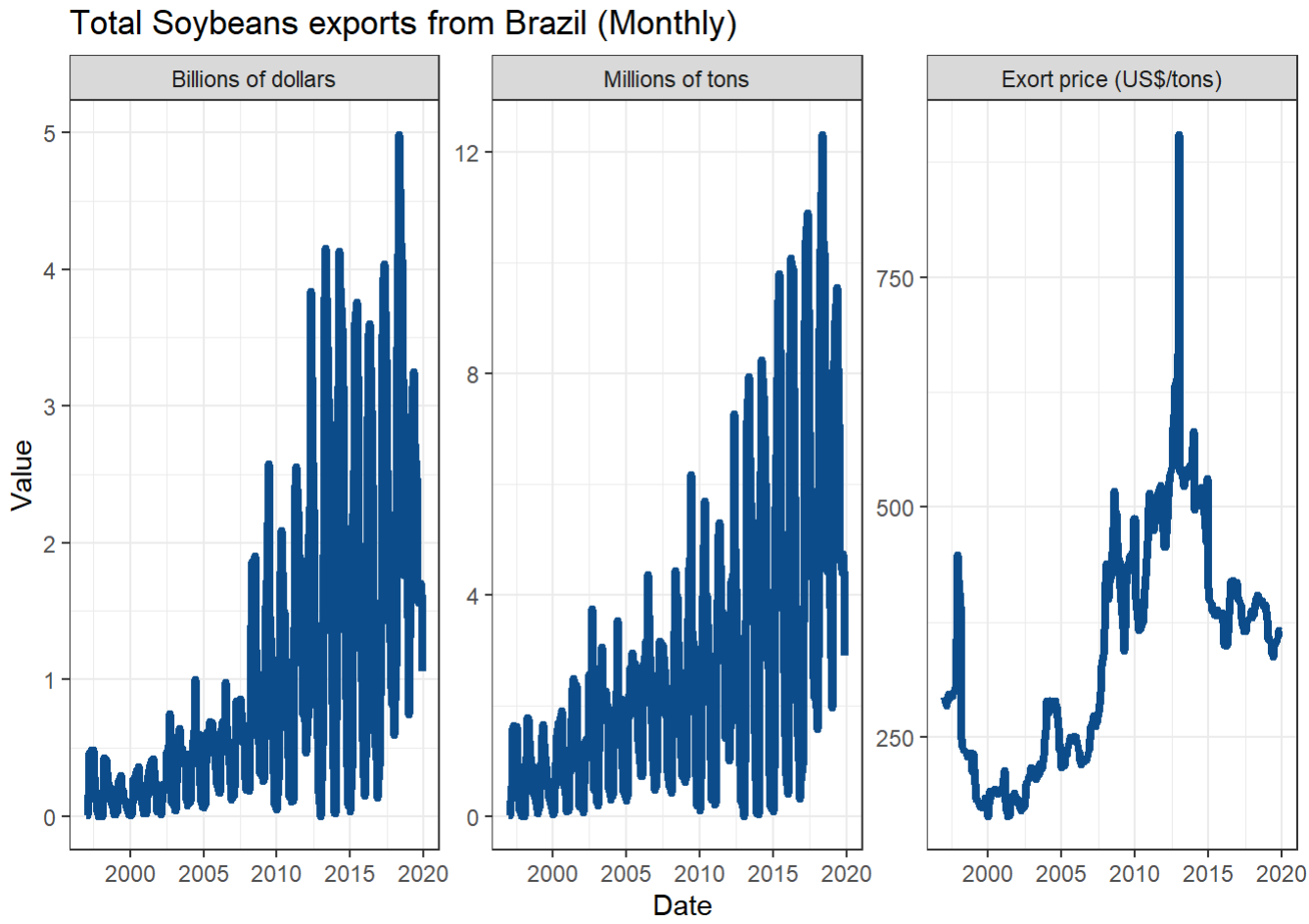
Summarizing the necessary information:

```
data_cmst.month <- data_cmst %>%      # data
  filter(type == "Export", product %in% c("soybeans", "soybean_oil", "soybean_meal")) %>%   # fi
lter export and products
  group_by(product,date) %>%
  summarise(total_export.usd = sum(usd)/10^9, total_export.tons = sum(tons)/10^6) %>%  # export
 b.usd and m.tons by product.month
  mutate(export.price = total_export.usd/total_export.tons*10^3) %>%    # export price
  rename("Billions of dollars" = total_export.usd, "Millions of tons" = total_export.tons, "Exor
t price (US$/tons)" = export.price) %>%
  melt(id.vars= c("product", "date")) %>%         # turning variables into observations
  as_tibble()
```

2.2.2.1 Soybeans

Plotting the data:

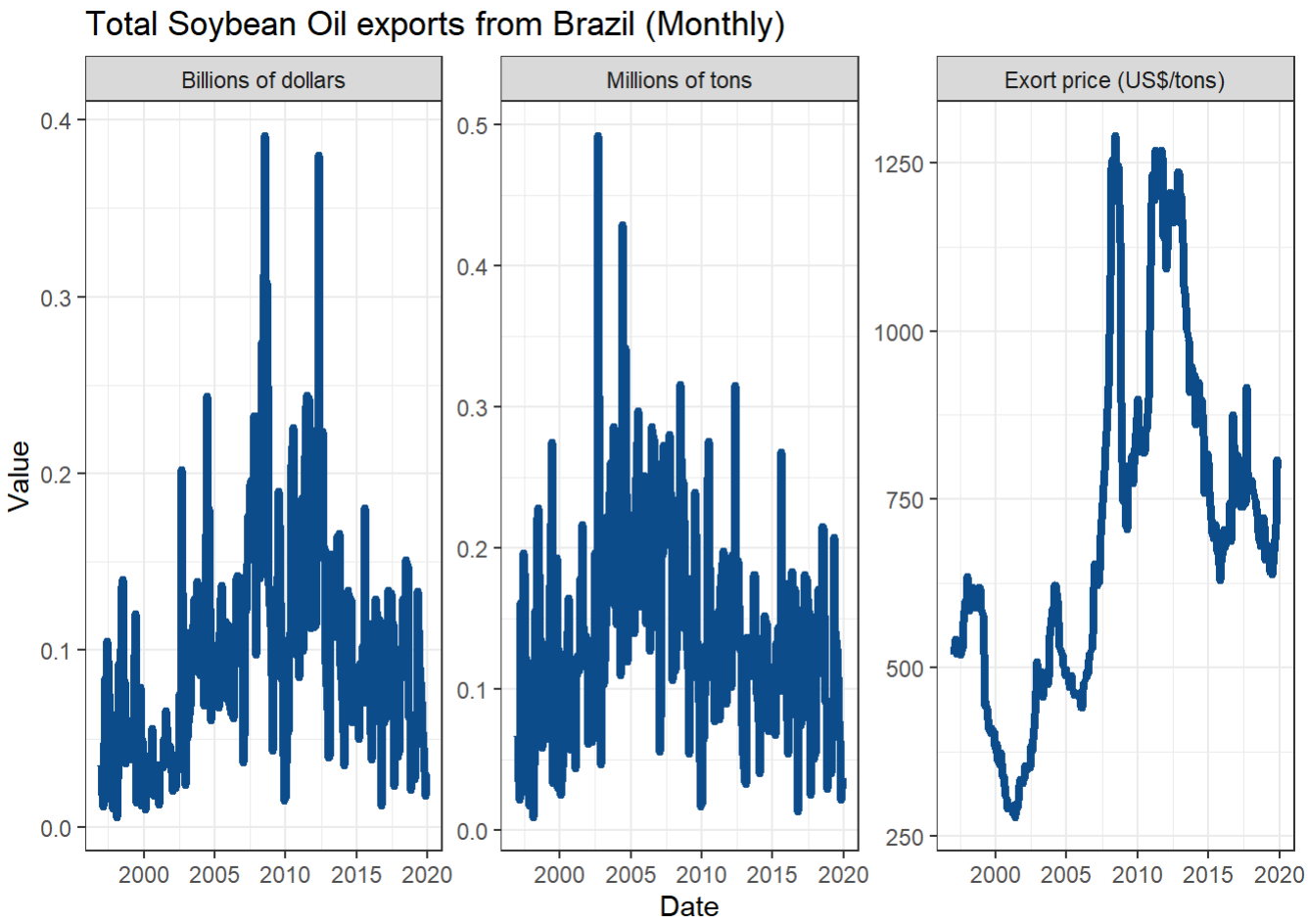
```
data_cmst.month %>%
  filter(product == "soybeans") %>%
  ggplot() +
    aes(x = date, y = value) +
    geom_line(size = 1.5, colour = "#0c4c8a") +
    labs(x = "Date", y = "Value", title = "Total Soybeans exports from Brazil (Monthly)") +
    theme_bw() +
    facet_wrap(vars(variable), scales = "free_y")
```



2.2.2.2 Soybean oil

Plotting the data:

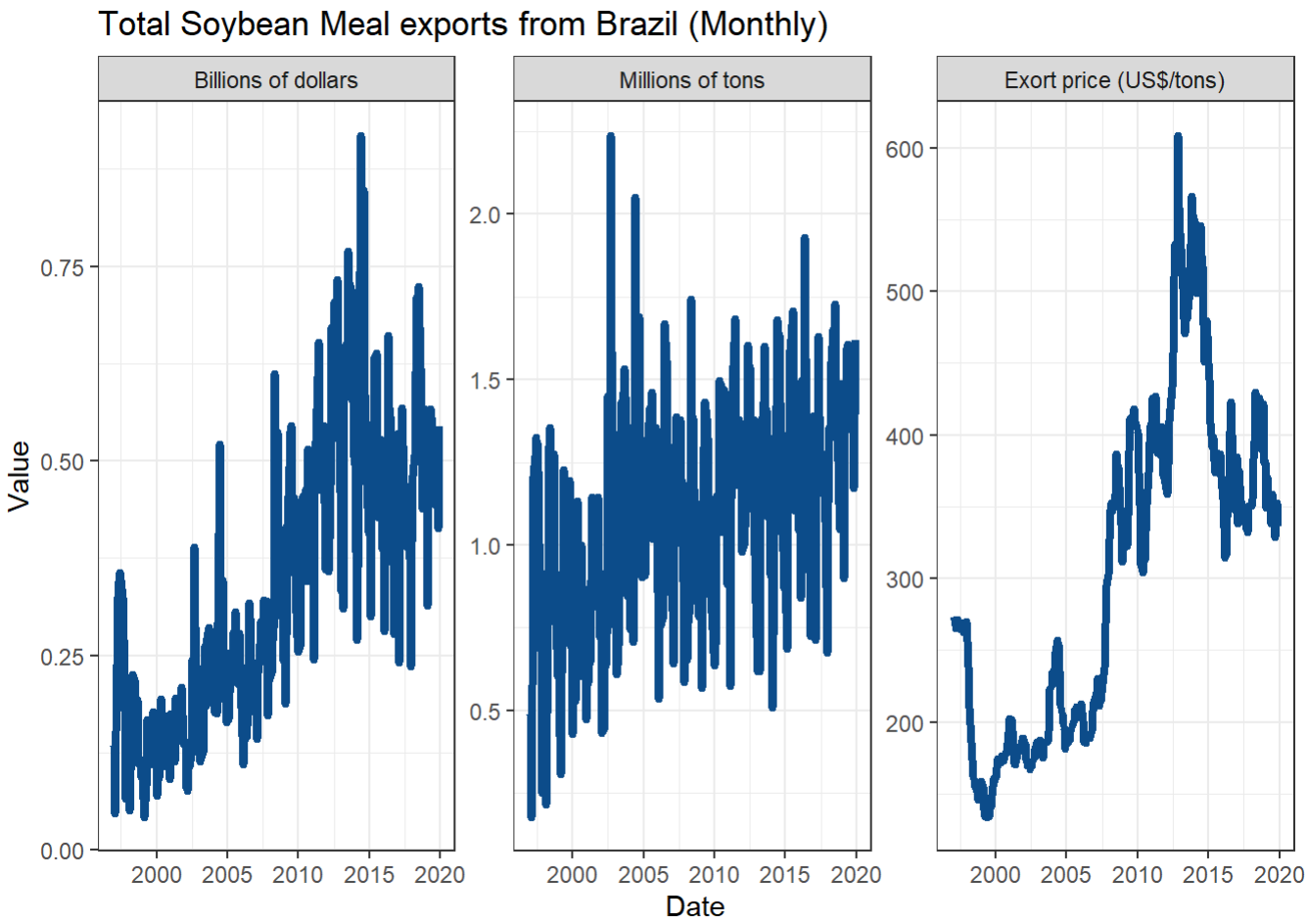
```
data_cmst.month %>%
  filter(product == "soybean_oil") %>%
  ggplot() +
    aes(x = date, y = value) +
    geom_line(size = 1.5, colour = "#0c4c8a") +
    labs(x = "Date", y = "Value", title = "Total Soybean Oil exports from Brazil (Monthly)") +
    theme_bw() +
    facet_wrap(vars(variable), scales = "free_y")
```



2.2.2.3 Soybean meal

Plotting the data:

```
data_cmst.month %>%
  filter(product == "soybean_meal") %>%
  ggplot() +
    aes(x = date, y = value) +
    geom_line(size = 1.5, colour = "#0c4c8a") +
    labs(x = "Date", y = "Value", title = "Total Soybean Meal exports from Brazil (Monthly)") +
    theme_bw() +
    facet_wrap(vars(variable), scales = "free_y")
```



2.3 What are the 3 most important products exported by Brazil in the last 5 years?

Selecting key arguments:

```
Y <- 5 # Number of the last years

n.pe <- 3 # Number of the most important products exported
```

Tidying up the data:

```
# getting the most exported products from Brazil in the last Y years

data_cmst.ly <- data_cmst %>%                          # data
  filter(type == "Export", year > max(year)-Y) %>%  # export in the last 5 years
  group_by(product) %>%
  summarise(usd.ly = sum(usd)/10^9) %>%             # export in billions of dollars by product
  mutate(percent = usd.ly/sum(usd.ly)) %>%          # Share of the products
  arrange(desc(usd.ly))                             # ordering
```

Plotting the data:

```
#Plotting the data:

data_cmst.ly %>%  # data
  ggplot() +      # plot
    aes(x = reorder(product,-usd.ly), weight = usd.ly) +
    geom_bar(fill = "#0c4c8a") +
    labs(x = "Product", y = "Billions of dollars", title = "Export from Brazil in the last 5 yea
rs by product") +
    theme_bw()
```

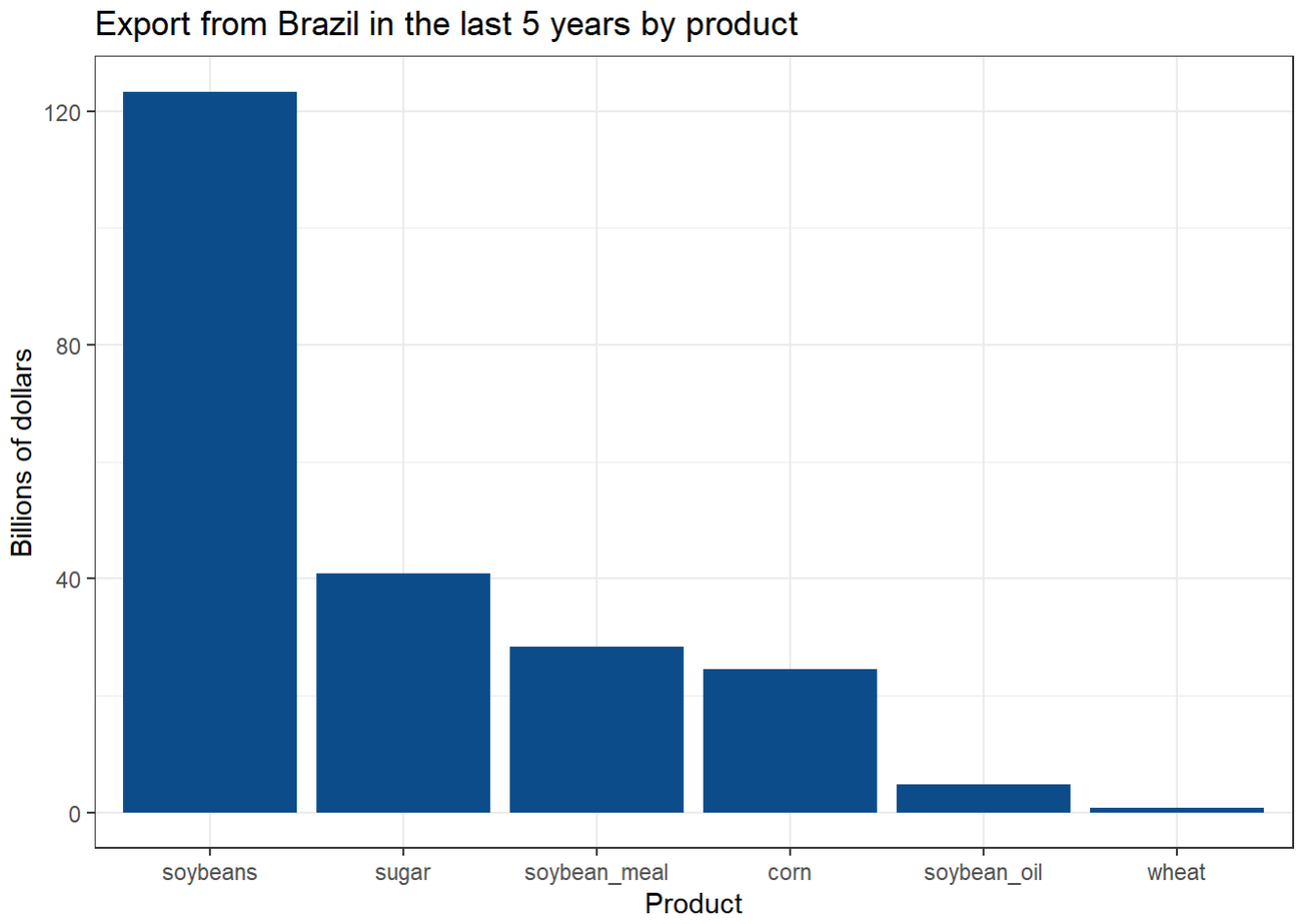


Table the 3 most important products exported:

```
# generating the table

data_cmst.ly %>%      # data
  top_n(n.pe) %>%     # choosing the most exported products

kable(digits = 2,
      align = "lcc",
      col.names = c("Product", "Export in billions of dollars", "Percent"),
      row.names = TRUE,
      caption = "The most important products exported by Brazil in the last 5 years (2015-2019)"
) %>%
  kable_styling(full_width = FALSE)
```

The most important products exported by Brazil in the last 5 years (2015-2019)

	Product	Export in billions of dollars	Percent
1	soybeans	123.33	0.55
2	sugar	40.94	0.18
3	soybean_meal	28.41	0.13

The table and the figure above show that the 3 products (among the 6 in the `data_comexstat.csv` file attached) most exported by Brazil in the last 5 years were: soybeans, sugar and soybean_meal.

2.4 What are the main routes through which Brazil have been exporting ‘corn’ in the last few years? Are there differences in the relative importance of routes depending on the product?

Selecting key arguments:

```
fY <- 5 # Number of last few years
```

Tidying up the data:

```
# getting the main routes through which Brazil have been export in the last fY years for all pro
ducts

data_cmst.rt <- data_cmst %>%     # data
  filter(type == "Export", year > max(year)-fY) %>%   # export in the last few (fY) years
  group_by(product, route) %>%
  summarise(usd.lfy = sum(usd), tons.lfy = sum(tons)) %>%    # export in b.dollars and m.tons by
product and route
  rename("Share in value (usd)" = usd.lfy, "share in volume (tons)" = tons.lfy) %>%
  melt(c("product", "route"))        # turning variables into observations
```

Table the main routes which Brazil have been exporting corn in the last 5 years:

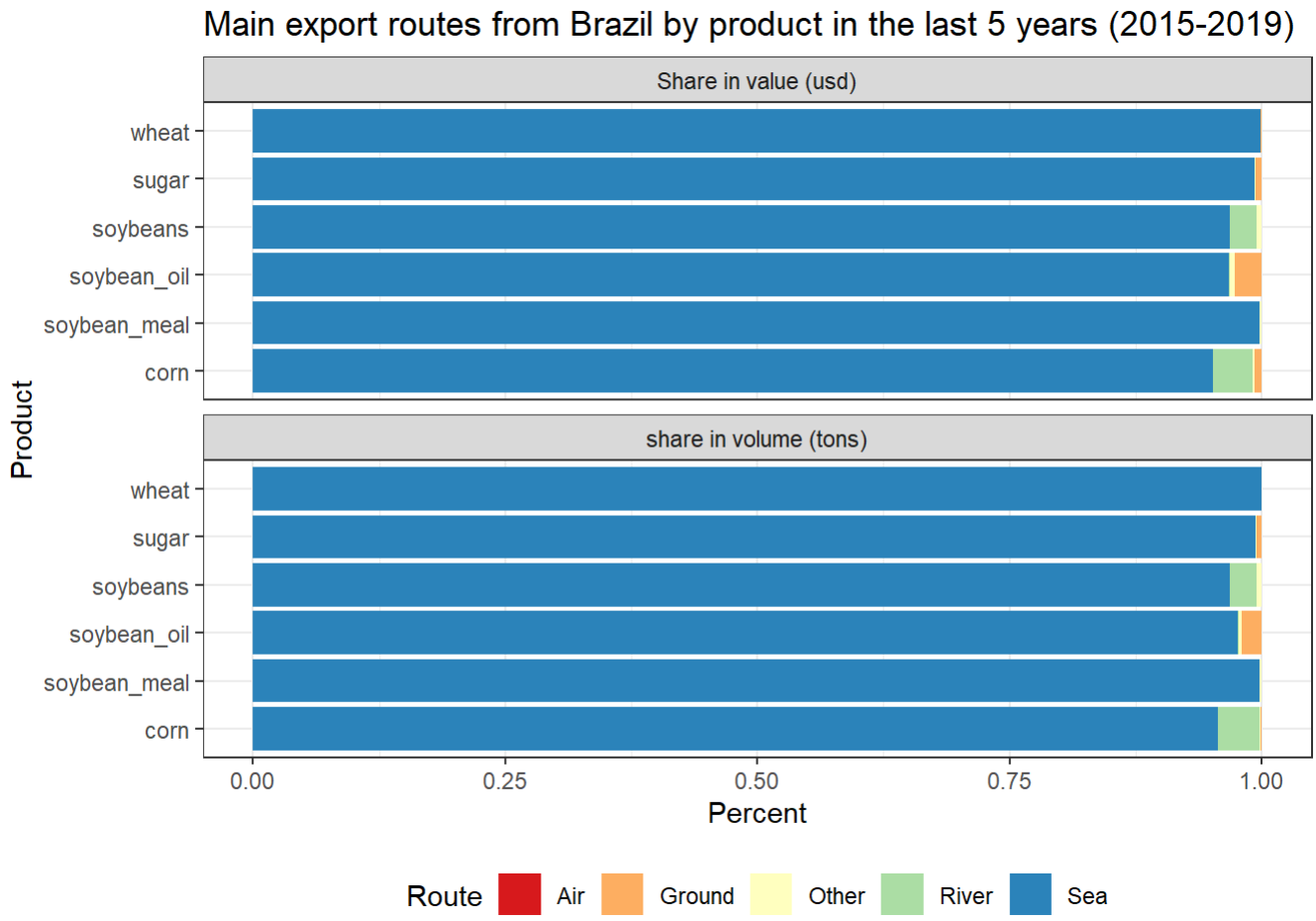
```
#tidying up the data

data_cmst.rt %>%      # data
  filter(product == "corn" , variable == "Share in value (usd)") %>%   # filter corn and usd
  dplyr::select(-variable) %>%             # removing a variable
  mutate( value = value/10^9,
          percent = value/sum(value)) %>%   # renaming and exchanging units
  arrange(desc(value))  %>%      # ordering

    kable(digits = 2,      #generating the table
          align = "llcc",
          col.names = c("Product", "Route", "Export in billions of dollars", "Percent"),
          row.names = TRUE,
          caption = "Main corn export routes from Brazil in the last 5 years (2015-2019)") %>%
    kable_styling(full_width = FALSE)
```

Main corn export routes from Brazil in the last 5 years (2015-2019)

	Product	Route	Export in billions of dollars	Percent
1	corn	Sea	23.26	0.95
2	corn	River	0.97	0.04
3	corn	Ground	0.17	0.01
4	corn	Other	0.04	0.00
5	corn	Air	0.00	0.00

Plotting all products:

```
# Plot the data

data_cmst.rt %>%   # data
  ggplot() +       # plot
    aes(x = product, fill = route, weight = value) +
    geom_bar(position = "fill") +
    scale_fill_brewer(palette = "Spectral") +
    labs(x = "Product", y = "Percent", title = "Main export routes from Brazil by product in the
last 5 years (2015-2019)", fill = "Route") +
    coord_flip() +
    theme_bw() +
    facet_wrap(vars(variable), nrow = 2L) +
    theme(legend.position = "bottom")
```



The table and the figure above show that the main corn export route from Brazil in the last 5 years was the *sea route*. Roughly speaking, all 6 products have a very similar structure to the export routes, that is, the *sea route* is by far the main one for all 6 products. However, small differences exist, such as the flow of a small portion of corn production by *river route*.

2.5 Which countries have been the most important trade partners for Brazil in terms of ‘corn’ and ‘sugar’ in the last 3 years?

Selecting key arguments:

```
y.tp <- 3 # Number of the last years

n.tp <- 5 # Number of the most important trade partners

prdct.tp <- c("corn", "sugar")  # in terms of ‘corn’ and ‘sugar’
```

Tidying up the data:

```
# getting the most important trade partners (n.tp) for Brazil in terms of 'corn' and 'sugar' in
 the last 3 (y.tp) years

data_cmst.tp <- data_cmst %>%
  filter(year > max(year)-y.tp, product %in% prdct.tp) %>% #corn and sugar in the last.3.y
  group_by(product, country) %>%
  summarise(trade = sum(usd)/10^9) %>%      # trade = Export + Import by country and product
  mutate(percent = trade/sum(trade)) %>%    # share of each country in trade by product
  top_n(n.tp) %>%                           # choosing the most important countries
  arrange(product, desc(trade)) %>%         # ordering
  mutate(n = 1:n.tp) %>%                    # numbering
  relocate(n, .before = product)            # relocating the columns
```

Plot the data:

```
# Plotting the data "corn":
data_cmst.tp %>%           # data
  ggplot() +               # plot
    aes(x = reorder(country, -trade), weight = trade) +
    geom_bar(fill = "#0c4c8a") +
    labs(x = "Country", y = "Billions of dollars",
         title = "The most important trade partners in Brazil in terms of corn and sugar in the
 last 3 years (2017-2019)") +
    theme_minimal() +
    facet_wrap(vars(product), scales = "free")
```

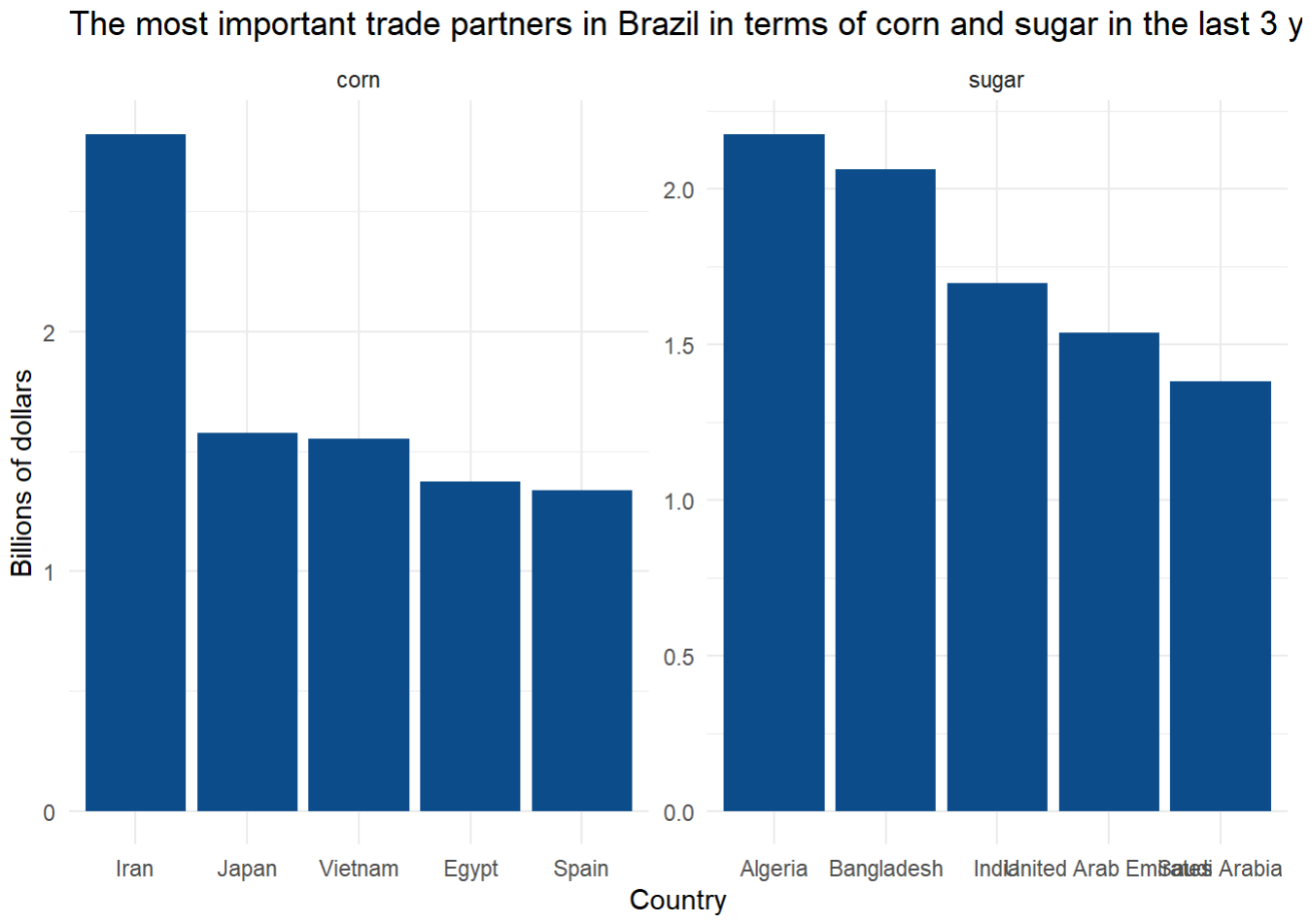


Table the most important trade partners for Brazil:

```
# generating the table
data_cmst.tp %>%              #data
  kable(digits =2,            #table
        align = "lllcc",
        col.names = c("", "Product", "Country", "Trade in billions of dollars", "Percent"),
        caption = "The most important trade partners for Brazil by product in the last 3 years
 (2017-2019)") %>%
  kable_styling(full_width = FALSE) %>%
  pack_rows(index = table(rep(prdct.tp, n.tp)))
```

The most important trade partners for Brazil by product in the last 3 years (2017-2019)

	Product	Country	Trade in billions of dollars	Percent
corn				
1	corn	Iran	2.82	0.17
2	corn	Japan	1.57	0.10
3	corn	Vietnam	1.55	0.10
4	corn	Egypt	1.38	0.08
5	corn	Spain	1.34	0.08
sugar				
1	sugar	Algeria	2.18	0.09
2	sugar	Bangladesh	2.06	0.09
3	sugar	India	1.70	0.07
4	sugar	United Arab Emirates	1.54	0.07
5	sugar	Saudi Arabia	1.38	0.06

The table and figure above show that, in the case of corn, the 5 most important trade partners for Brazil in the last 3 years were: Iran,Japan, Vietnam, Egypt and Spain. In the case of sugar, the most important were: Algeria,Bangladesh, India, United Arab Emirates and Saudi Arabia. It should be noted that the trade volume of these two products is relatively dispersed, when compared to soybeans, which are mainly destined for China.

2.6 For each of the products in the dataset, show the 5 most important states in terms of exports?

Selecting key arguments:

```
n.st <- 5 # Number of the most important states

prdct.st <- c("corn", "soybean_meal", "soybean_oil", "soybeans", "sugar", "wheat")
```

Tidying up the data:

```
# getting the 5 most important states in terms of exports

data_cmst.st <- data_cmst %>%
  filter(type == "Export", product %in% prdct.st) %>%               # in terms of exports
  group_by(product, state) %>%
  summarise(export = sum(usd)/10^9) %>%      # Export by product and state
  mutate(percent = export/sum(export)) %>%   # share of each state in exports by product
  top_n(n.st) %>%                            # choosing the most important states
  arrange(product, desc(export)) %>%         # ordering
  mutate(n = 1:n.st) %>%                     # numbering
  relocate(n, .before = product)             # changing the order of the columns
```

Plotting the data:

```
# plotting the data
data_cmst.st %>%   #data
  ggplot() +       # plot
    aes(x = state, weight = export) +
    geom_bar(fill = "#0c4c8a") +
    labs(x = "State", y = "Billions of dollars", title = "The 5 most important states in terms o
f exports by product (1997-2019)") +
    theme_bw() +
    facet_wrap(vars(product), scales = "free")
```

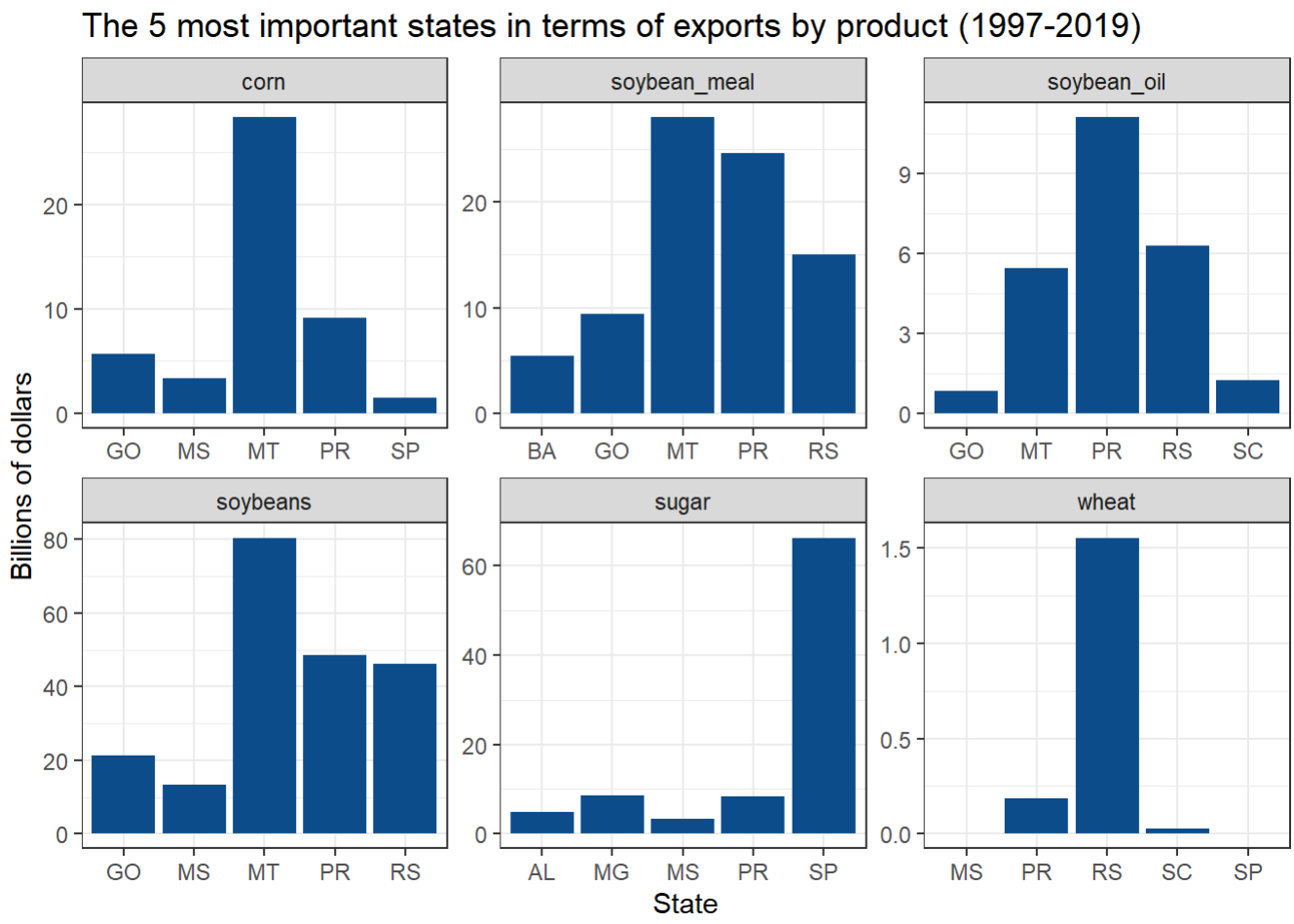


Table the main exporting states of Brazil:

```
# generating the table
data_cmst.st %>%     # data
  kable(digits = 2,  # table
        align = "lllcc",
        col.names = c("", "Product", "State", "Export in billions of dollars", "Percent"),
        caption = "The 5 most important states in terms of exports by product (1997-2019)") %>%
  kable_styling(full_width = FALSE) %>%
  pack_rows(index = table(rep(prdct.st, n.st)))
```

The 5 most important states in terms of exports by product (1997-2019)

	Product	State	Export in billions of dollars	Percent
corn				
1	corn	MT	28.38	0.55
2	corn	PR	9.16	0.18
3	corn	GO	5.71	0.11
4	corn	MS	3.41	0.07

	Product	State	Export in billions of dollars	Percent
5	corn	SP	1.51	0.03
soybean_meal				
1	soybean_meal	MT	27.99	0.30
2	soybean_meal	PR	24.65	0.26
3	soybean_meal	RS	15.01	0.16
4	soybean_meal	GO	9.42	0.10
5	soybean_meal	BA	5.44	0.06
soybean_oil				
1	soybean_oil	PR	11.11	0.41
2	soybean_oil	RS	6.28	0.23
3	soybean_oil	MT	5.43	0.20
4	soybean_oil	SC	1.24	0.05
5	soybean_oil	GO	0.86	0.03
soybeans				
1	soybeans	MT	80.51	0.29
2	soybeans	PR	48.48	0.18
3	soybeans	RS	46.14	0.17
4	soybeans	GO	21.27	0.08
5	soybeans	MS	13.28	0.05
sugar				
1	sugar	SP	66.21	0.68
2	sugar	MG	8.56	0.09
3	sugar	PR	8.27	0.08
4	sugar	AL	4.83	0.05
5	sugar	MS	3.22	0.03

	Product	State	Export in billions of dollars	Percent
wheat				
1	wheat	RS	1.56	0.88
2	wheat	PR	0.18	0.10
3	wheat	SC	0.03	0.01
4	wheat	SP	0.00	0.00
5	wheat	MS	0.00	0.00

2.7 What should be the total brazilian soybeans, soybean_meal, and corn export forecasts, in tons, for the next 11 years (2020-2030)? We're mostly interested in the annual forecast.

It should be noted that we will estimate a model for each product. For each one, we will do the modeling using three variables: exports in tons, price and world GDP (the last two variables available in the covariate database). In addition, in all cases we will use a time series econometrics approach. That is, we will do the cointegration test procedures. If the series are cointegrated, we will estimate a VECM. Then, we will transform it into a VAR and carry out the forecasts. On the other hand, if the series do not cointegrate, we will estimate a VAR in the first difference (i.e. with stationary series) and carry out the forecasts. Finally, it is worth noting that we will make a logarithmic transformation in all variables.

Import covariates.xlsx data

```
covariates <- read_excel("covariates.xlsx")
```

2.7.1 Soybeans

Tidying up the data for modeling

```
# generating the exports data for modeling

product_s <- "soybeans"  #choosing the product

covarietes_slct <- covariates %>%
  dplyr::select(contains(c("year", product_s,"world"))) # removing other prices and gdp


data_s <- data_cmst %>%   # data
  filter(product == product_s) %>%  #filter by product
  group_by(year) %>%
  summarise(export_tons = sum(tons)) %>%  # export in tons by years
  inner_join(covarietes_slct,by = "year") %>%  # joining cavariates and exports data - years con
tained in exports only
  rename(price = paste0("price_",product_s)) %>%  # change the name of price variable
  mutate_all(log) %>%   # log transformation
  dplyr::select(-year) %>%   # removing the year variable
  ts(start = min(data_cmst$year), end = max(data_cmst$year))  # time series
```

First, we will analyze the stationarity of the three series in log: export (tons), price and world gdp. The result of adt test indicates that all three are integrated in order 1, I (1).

```
# Making the adf test for stationarity
  #export_tons
data_s[,1] %>%
  ndiffs(test = "adf")
```

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

```
  #price
data_s[,2] %>%
  ndiffs(test = "adf")
```

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

```
  #gdp_world
data_s[,3] %>%
  ndiffs(test = "adf")
```

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

Thus, we can perform the cointegration test. We chose the Johasen cointegration test.

```
# Johansen Cointegration test

ca_jo.s <- data_s %>%  # data
  ca.jo(type = "trace",
        K = (VARselect(data_s)$selection %>%
                mean() %>%
                round()) -1) # lag selection

summary(ca_jo.s)  # Johansen test results
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.626844849 0.277299217 0.007847558
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 2 |  0.16  6.50  8.18 11.65
## r <= 1 |  6.65 15.66 17.95 23.52
## r = 0  | 26.37 28.71 31.52 37.22
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                export_tons.l3    price.l3 gdp_world.l3
## export_tons.l3      1.0000000  1.00000000     1.000000
## price.l3            0.3434157 -0.08180894     2.714105
## gdp_world.l3       -2.9315044 -3.87907321    -6.506515
##
## Weights W:
## (This is the loading matrix)
##
##               export_tons.l3   price.l3  gdp_world.l3
## export_tons.d     -1.1788765 0.04149115  2.041534e-02
## price.d            0.4060469 0.13738932 -2.996721e-02
## gdp_world.d        0.1177783 0.01204723  5.286993e-05
```

The result of the Johansen test indicates that there is 1 cointegration vector. Thus, we will define the following parameter:

```
n_coint.s <- 1 # number of cointegration vector
```

Now, we will transform the estimated VECM in the Johansen test into a VAR and we will do diagnostic tests on the residuals. All tests indicate that the model fits well.

```
#transforming VECM into VAR
 VAR.s <- vec2var(ca_jo.s,       # Johansen Cointegration test model
                r = n_coint.s) # number of cointegration vector

 # diagnostic tests
 serial.test(VAR.s)
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object VAR.s
## Chi-squared = 71.075, df = 120, p-value = 0.9999
```

```
 arch.test(VAR.s)
```

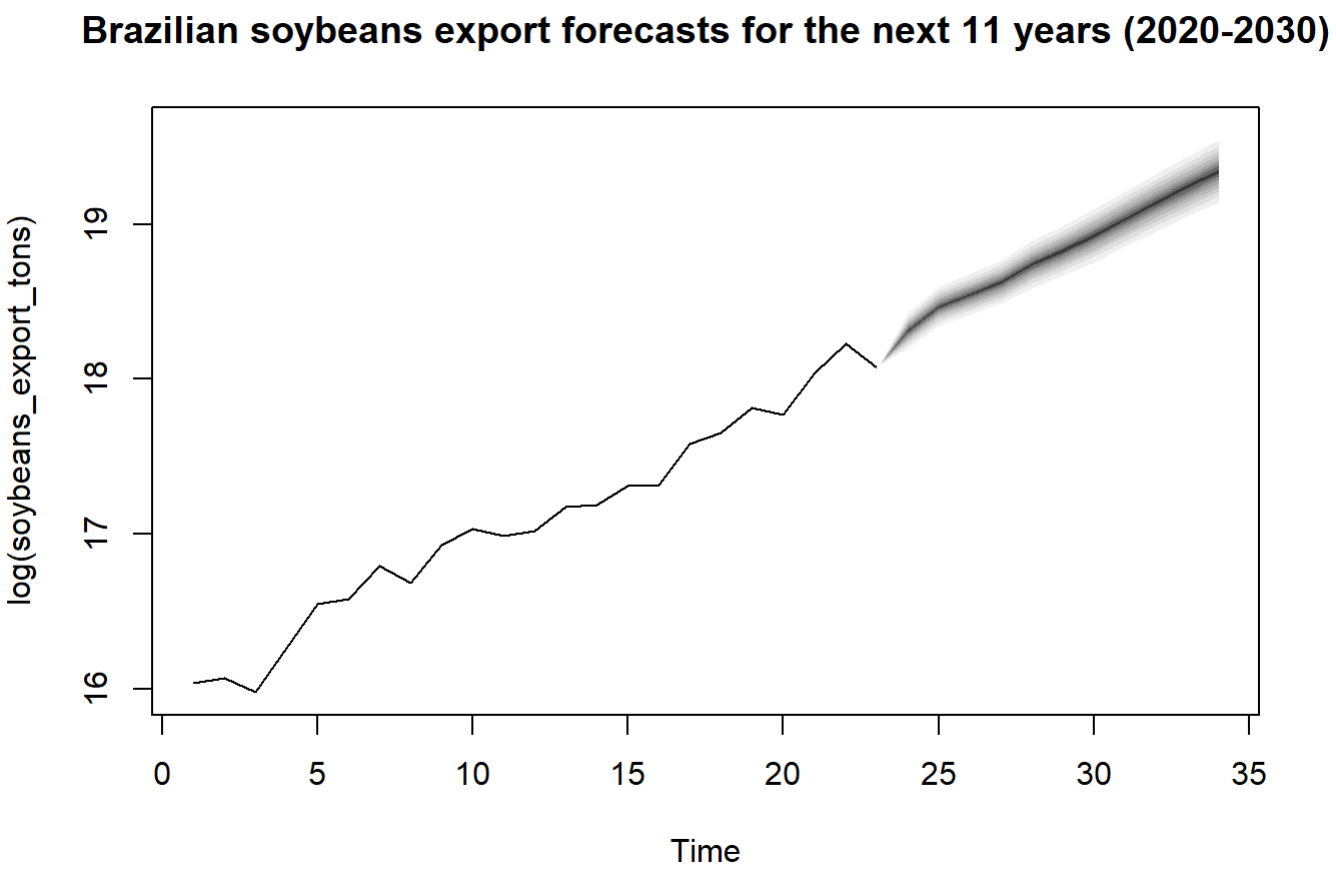
```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object VAR.s
## Chi-squared = 90, df = 180, p-value = 1
```

```
 normality.test(VAR.s)
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object VAR.s
## Chi-squared = 3.7679, df = 6, p-value = 0.708
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object VAR.s
## Chi-squared = 1.7802, df = 3, p-value = 0.6193
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object VAR.s
## Chi-squared = 1.9877, df = 3, p-value = 0.575
```

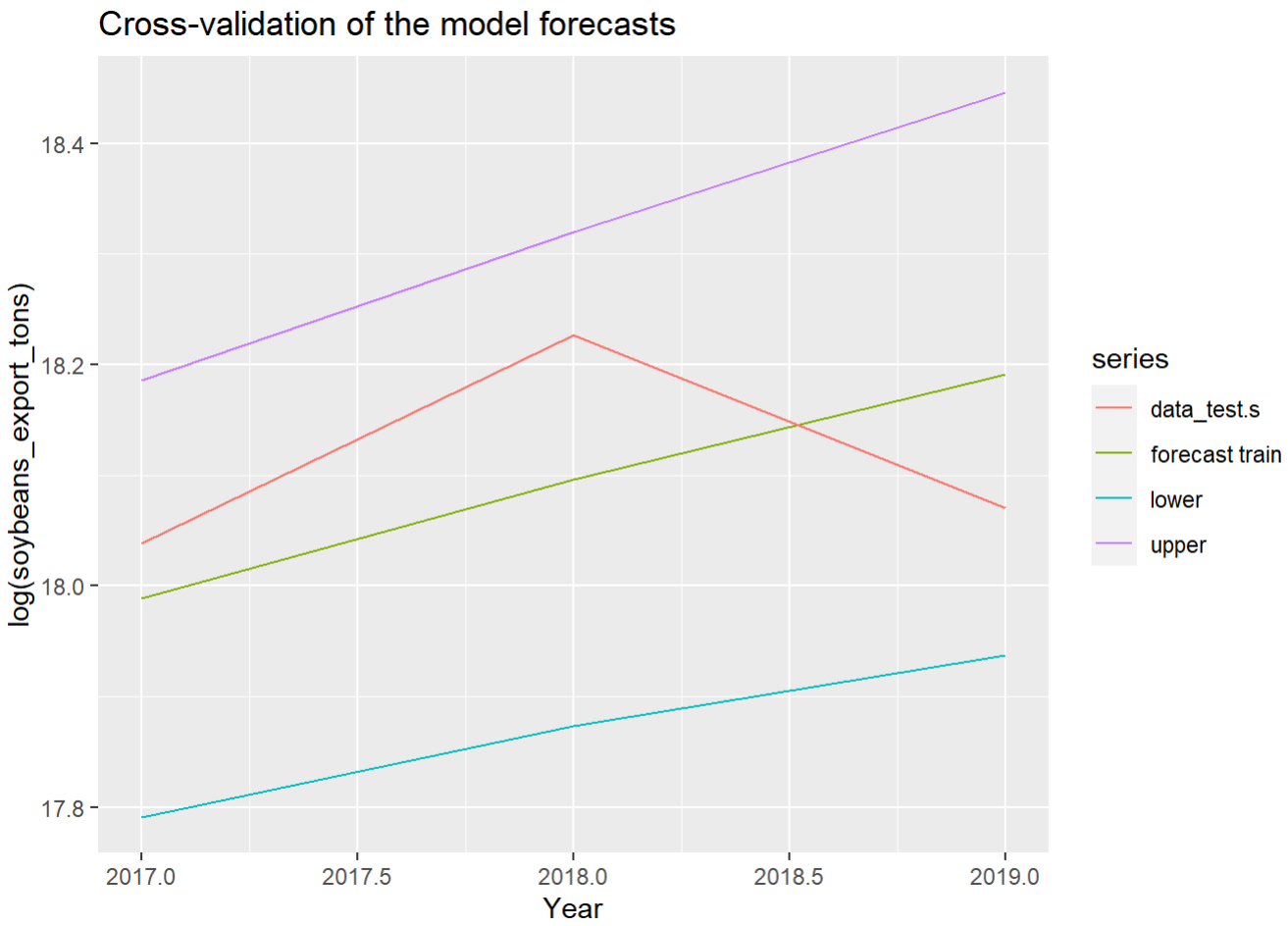
Therefore, we can make export forecasts for the next 11 years (2020-2030).

```
# Forecasting the exports tons

forecast.s <- predict(VAR.s, n.ahead = 11)

forecast.s %>%
  fanchart(names = "export_tons", ylab = paste0("log(",product_s,"_export_tons)"), xlab = "Time"
,
         main = paste0("Brazilian ", product_s, " export forecasts for the next 11 years (2020-2
030)"))
```



When performing a cross-validation we see that the model has a good forecasting capacity.

```
l.s <- 3 # number of periods out-of-sample

# test data - out of sample

data_test.s <- data_s[,1] %>% window(start = max(data_cmst$year)+1-l.s)

# training data - inside of sample

data_train_s <- data_s %>%
                window(end = max(data_cmst$year)-l.s)

# training model
ca_jo.train_s <- data_train_s %>%
  ca.jo(type = "trace",
        K = (VARselect(data_s)$selection %>%
               mean() %>%
               round()) -1) # lag selection

VAR.train_s <- vec2var(ca_jo.train_s,  #transforming VECM (Johansen Cointegration test model) in
to VAR
                 r = n_coint.s)   # number of cointegration vector

# training forecast

forecast.train_s <- predict(VAR.train_s, n.ahead = l.s)
forecast.train_s <- forecast.train_s[[1]]
forecast.train_s <- data.frame(forecast.train_s$export_tons[,1:3]) %>%
                        rename( "forecast train" = fcst) %>%
                        ts(start = max(data_cmst$year)+1-l.s)

#plot test data and training forecast

autoplot(forecast.train_s) +
  autolayer(data_test.s) +
  labs(x= "Year",y= paste0("log(",product_s,"_export_tons)"),
       title="Cross-validation of the model forecasts")
```



2.7.2 Soybeans_meal

Tidying up the data for modeling

```
# generating the exports data for modeling

product_sm <- "soybean_meal"  #choosing the product

covarietes_slct <- covariates %>%
  dplyr::select(contains(c("year", product_sm,"world"))) # removing other prices and gdp


data_sm <- data_cmst %>%   # data
  filter(product == product_sm) %>%  #filter by product
  group_by(year) %>%
  summarise(export_tons = sum(tons)) %>%  # export in tons by years
  inner_join(covarietes_slct,by = "year") %>%  # joining cavariates and exports data - years con
tained in exports only
  rename(price = paste0("price_",product_sm)) %>%  # change the name of price variable
  mutate_all(log) %>%   # log transformation
  dplyr::select(-year) %>%   # removing the year variable
  ts(start = min(data_cmst$year), end = max(data_cmst$year))  # time series
```

We will analyze the stationarity of the three series in log: export (tons), price and world gdp. The result of adt test indicates that all three are integrated in order 1, I (1).

```
# Making the adf test for stationarity
  #export_tons
data_sm[,1] %>%
  ndiffs(test = "adf")
```

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

```
  #price
data_sm[,2] %>%
  ndiffs(test = "adf")
```

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

```
  #gdp_world
data_sm[,3] %>%
  ndiffs(test = "adf")
```

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

Thus, we can perform the Johasen cointegration test.

```
# Johansen Cointegration test

ca_jo.sm <- data_sm %>%  # data
  ca.jo(type = "trace",
        K = (VARselect(data_sm)$selection %>%
                mean() %>%
                round()) -1) # lag selection

summary(ca_jo.sm)  # Johansen test results
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.9207002880 0.2807163990 0.0005650992
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 2 |  0.01  6.50  8.18 11.65
## r <= 1 |  6.60 15.66 17.95 23.52
## r = 0  | 57.29 28.71 31.52 37.22
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                export_tons.l3  price.l3 gdp_world.l3
## export_tons.l3     1.00000000  1.000000    1.0000000
## price.l3           0.08806742 -4.484252   -0.8555189
## gdp_world.l3      -0.65689271  3.513093    1.2368039
##
## Weights W:
## (This is the loading matrix)
##
##               export_tons.l3    price.l3  gdp_world.l3
## export_tons.d     -0.9275859 0.033048290  0.0046337582
## price.d            0.2400955 0.059615210 -0.0154922981
## gdp_world.d        0.1749518 0.004093412  0.0003050201
```

The result of the Johansen test indicates that there is 1 cointegration vector. Thus, we will define the following parameter:

```
n_coint.sm <- 1 # number of cointegration vector
```

Now, we will transform the estimated VECM in the Johansen test into a VAR and we will do diagnostic tests on the residuals. All tests indicate that the model fits well.

```
#transforming VECM into VAR
 VAR.sm <- vec2var(ca_jo.sm,       # Johansen Cointegration test model
                r = n_coint.sm) # number of cointegration vector

 # diagnostic tests
 serial.test(VAR.sm)
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object VAR.sm
## Chi-squared = 68.708, df = 120, p-value = 1
```

```
arch.test(VAR.sm)
```

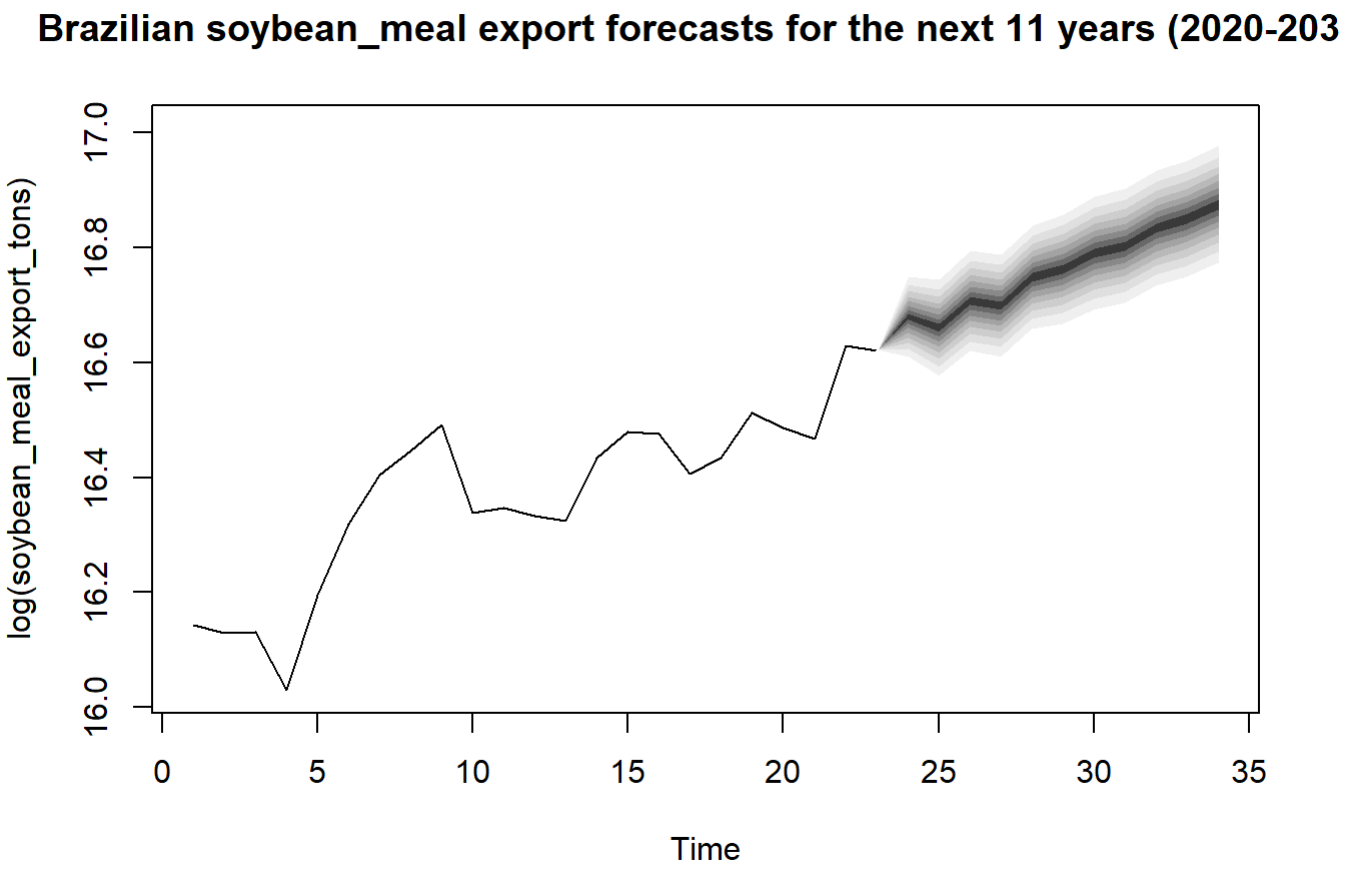
```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object VAR.sm
## Chi-squared = 90, df = 180, p-value = 1
```

```
normality.test(VAR.sm)
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object VAR.sm
## Chi-squared = 4.462, df = 6, p-value = 0.6144
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object VAR.sm
## Chi-squared = 2.8143, df = 3, p-value = 0.4212
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object VAR.sm
## Chi-squared = 1.6478, df = 3, p-value = 0.6486
```

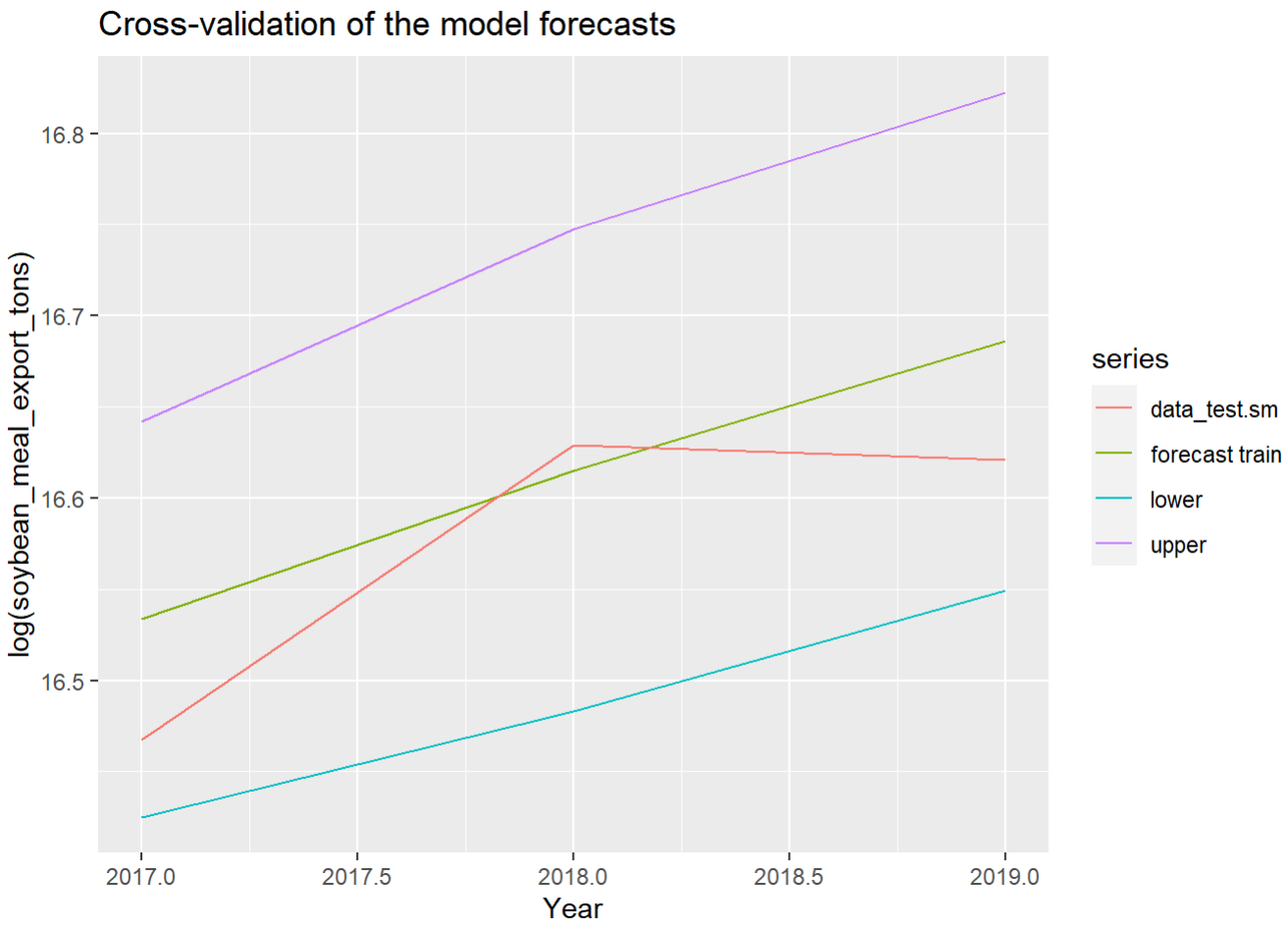
Therefore, we can make export forecasts for the next 11 years (2020-2030).

```
# Forecasting the exports tons

forecast.sm <- predict(VAR.sm, n.ahead = 11)

# plot
forecast.sm %>%
  fanchart(names = "export_tons", ylab = paste0("log(",product_sm,"_export_tons)"), xlab = "Tim
e",
         main = paste0("Brazilian ", product_sm, " export forecasts for the next 11 years (2020-
2030)"))
```



When doing a cross-validation we see that the corn model does not have as good a forecasting capacity as the soybean and soybean meal models.

```
l.sm <- 3 # number of periods out-of-sample

# test data - out of sample

data_test.sm <- data_sm[,1] %>% window(start = max(data_cmst$year)+1-l.sm)

# training data - inside of sample

data_train_sm <- data_sm %>%
                 window(end = max(data_cmst$year)-l.sm)

# training model
ca_jo.train_sm <- data_train_sm %>%
  ca.jo(type = "trace",
        K = (VARselect(data_sm)$selection %>%
               mean() %>%
               round()) -1) # lag selection

VAR.train_sm <- vec2var(ca_jo.train_sm,  #transforming VECM (Johansen Cointegration test model)
 into VAR
                  r = n_coint.sm)   # number of cointegration vector

# training forecast

forecast.train_sm <- predict(VAR.train_sm, n.ahead = l.sm)
forecast.train_sm <- forecast.train_sm[[1]]
forecast.train_sm <- data.frame(forecast.train_sm$export_tons[,1:3]) %>%
                         rename( "forecast train" = fcst) %>%
                         ts(start = max(data_cmst$year)+1-l.sm)

#plot test data and training forecast

autoplot(forecast.train_sm) +
  autolayer(data_test.sm) +
  labs(x= "Year",y= paste0("log(",product_sm,"_export_tons)"),
       title="Cross-validation of the model forecasts")
```



2.7.3 Corn

Tidying up the data for modeling

```
# generating the exports data for modeling

product_c <- "corn"  #choosing the product

covarietes_slct <- covariates %>%
  dplyr::select(contains(c("year", product_c,"world"))) # removing price of other products


data_c <- data_cmst %>%   # data
  filter(product == product_c) %>%  #filter by product
  group_by(year) %>%
  summarise(export_tons = sum(tons)) %>%  # export in tons by years
  inner_join(covarietes_slct,by = "year") %>%  # joining cavariates and exports data - years con
tained in exports only
  rename(price = paste0("price_",product_c)) %>%
  mutate_all(log) %>%   # log transformation
  dplyr::select(-year) %>%
  ts(start = min(data_cmst$year), end = max(data_cmst$year))  # time series
```

We will analyze the stationarity of the three series in log: export (tons), price and world gdp. The result of adt test indicates that all three are integrated in order 1, I (1).

```
# Making the adf test for stationarity
#export_tons
data_c[,1] %>%
  ndiffs(test = "adf")
```

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

```
#price
data_c[,2] %>%
  ndiffs(test = "adf")
```

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

```
#gdp_world
data_c[,3] %>%
  ndiffs(test = "adf")
```

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

Thus, we can perform the Johasen cointegration test. Unlike the case of soybeans and soybeam_meal, in the case of corn the result of the Johansen test indicates that the series do not cointegrate.

```
# Johansen Cointegration test

ca_jo.c <- data_c %>%  # data
  ca.jo(type = "trace",
        K = (VARselect(data_c)$selection %>%
               mean() %>%
               round()) -1) # lag selection
summary(ca_jo.c)  # Johansen test results
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.5106274 0.2878953 0.1541953
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 2 |  3.35  6.50  8.18 11.65
## r <= 1 | 10.14 15.66 17.95 23.52
## r = 0  | 24.43 28.71 31.52 37.22
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                export_tons.l3   price.l3 gdp_world.l3
## export_tons.l3      1.0000000   1.000000     1.000000
## price.l3           -0.1077534   3.780679    -1.516801
## gdp_world.l3       -3.8925723 -11.226030    -4.718273
##
## Weights W:
## (This is the loading matrix)
##
##               export_tons.l3      price.l3 gdp_world.l3
## export_tons.d  -1.2193766933  0.0825624189  -0.04644516
## price.d        -0.2151438807 -0.0706089224   0.05554351
## gdp_world.d    -0.0001451105  0.0002940069   0.00659239
```

```
                # number of cointegration vector = 0
```

In this way, we will estimate a VAR with the series in first difference (that is, with stationary series) and we will carry out diagnostic tests of the residuals. All results indicate that the model has a good fit.

```
# estimating the VAR

VAR.c <- diff(data_c) %>%
             VAR(p = VARselect(data_c)$selection %>%
                     mean() %>%
                     round()) # lag selection

# diagnostic tests

serial.test(VAR.c)
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object VAR.c
## Chi-squared = 75.016, df = 108, p-value = 0.9934
```

```
arch.test(VAR.c)
```

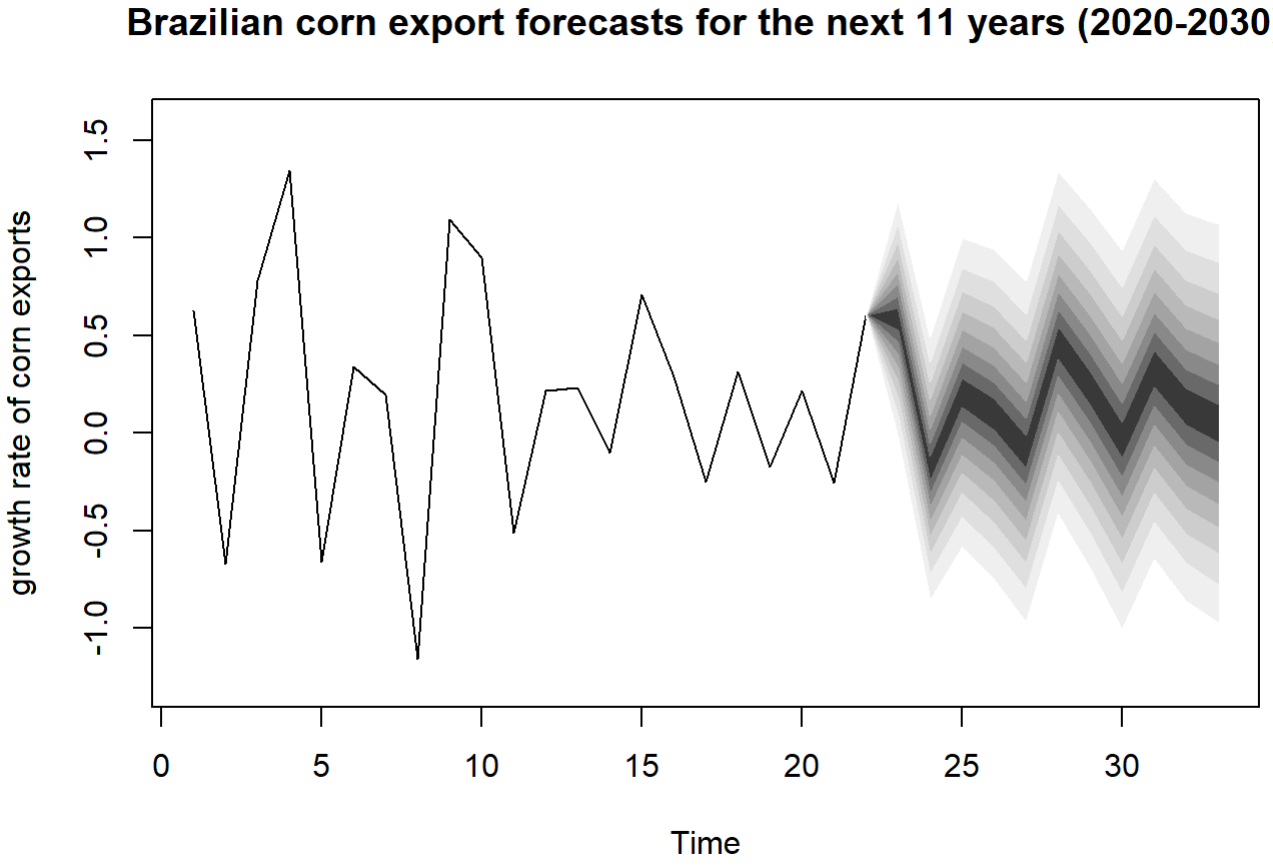
```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object VAR.c
## Chi-squared = 78, df = 180, p-value = 1
```

```
normality.test(VAR.c)
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object VAR.c
## Chi-squared = 1.8467, df = 6, p-value = 0.9332
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object VAR.c
## Chi-squared = 0.96859, df = 3, p-value = 0.8089
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object VAR.c
## Chi-squared = 0.87816, df = 3, p-value = 0.8307
```

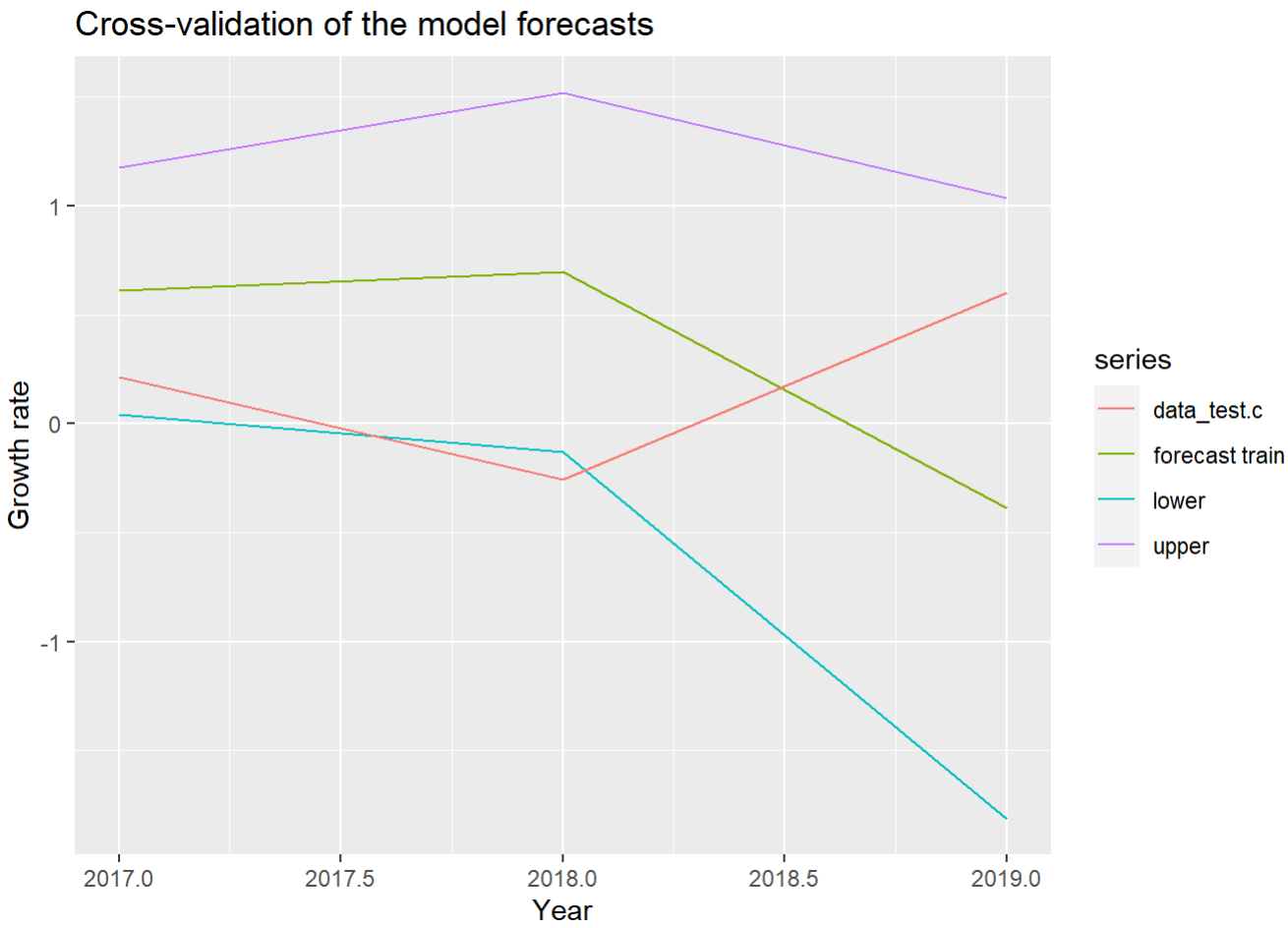
Therefore, we can make export forecasts for the next 11 years (2020-2030).

```
# Forecasting the exports tons

forecast.c <- predict(VAR.c, n.ahead = 11)

# Plot
forecast.c %>%
  fanchart(names = "export_tons", ylab = "growth rate of corn exports", xlab = "Time",
           main = paste0("Brazilian ", product_c, " export forecasts for the next 11 years (2020
-2030)"))
```



When doing a cross-validation we see that the corn model does not have as good a forecasting capacity as the soybean and soybean meal models.

```
l.c <- 3 # number of periods out-of-sample

# test data - out of sample

data_test.c <- diff(data_c)[,1] %>% window(start = max(data_cmst$year)+1-l.c)

# training data - inside of sample

data_train_c <- data_c %>%
                window(end = max(data_cmst$year)-l.c)

# training model
VAR.train_c <- diff(data_train_c) %>%
  VAR(p = VARselect(data_c)$selection %>%
        mean() %>%
        round()) # lag selection


# training forecast
forecast.train_c <- predict(VAR.train_c, n.ahead = l.c)
forecast.train_c <- forecast.train_c[[1]]
forecast.train_c <- data.frame(forecast.train_c$export_tons[,1:3]) %>%
                        rename( "forecast train" = fcst) %>%
                        ts(start = max(data_cmst$year)+1-l.c)
#plot test data and training forecast

autoplot(forecast.train_c) +
  autolayer(data_test.c) +
  labs(x= "Year",y= "Growth rate",
       title="Cross-validation of the model forecasts")
```


Cross-validation of the model forecasts
Growth rate
1
0
-1
2017.0
2017.5
2018.0
2018.5
2019.0
Year
series
data_test.c
forecast train
lower
upper