Tobacco consumption prediction for 2021

Elias Garza, Diego Rodriguez, Federico Medina

28/2/2022

```
library(dplyr)
library(fpp2)
library(readr)
library(ggplot2)
library(forecast)
library(forecastHybrid)
library(gbm)
library(nnfor)
```

Reading Data

First of all, we need to read the data and analyze it to understand the structure of the dataset and decide what to do. Here, we realized that the dataset is conformed by the data of 13 tobacco products in 20 years. Also, we decided to create a new column named item which concatenate other 2 variables (Submeasure and Data Value Unit), in this way is easier to work with each product given that some of them have the same submeasure.

```
Tmatrix <- read_csv("Tobacco_Consumption.csv")
Tdata<-as.data.frame(Tmatrix)
Tdata$item<-paste(Tdata$Submeasure," in ",Tdata$`Data Value Unit`)
```

Dividing by products

Next step, we divided the dataset in 13 different dataframes for each product using the new variable item so we can work with them separately.

```
Products<-list()
for (i in Tdata$item[1:13]){
  Products<-c(Products,list(filter(Tdata, item==i)))
}
names(Products)<-Tdata$item[1:13]
```

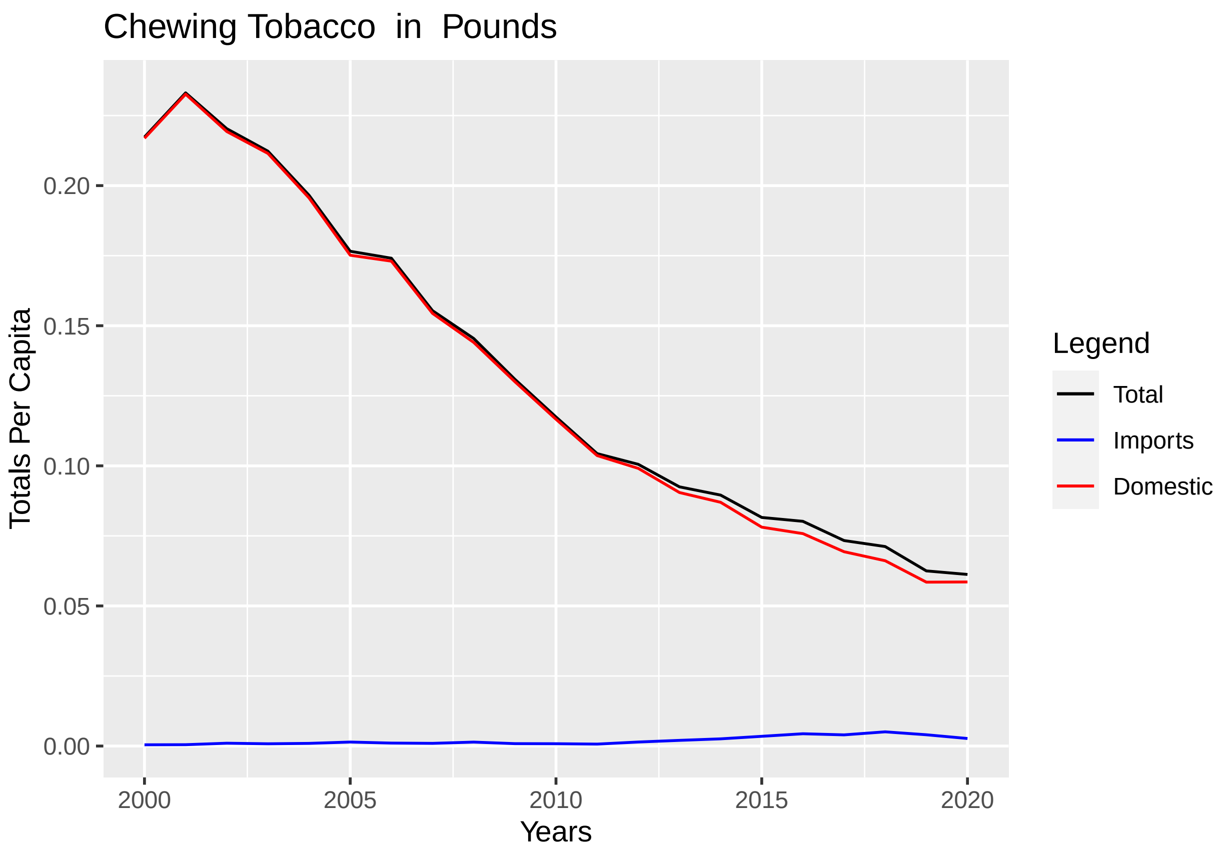
and created df of Totals, Imports and Domestic per Capita per Product since these are de variables with which we will work. For this, we created new vectors for the per capita values, since the ones on the original dataframe appear to be rounded and some of this values cause problems, more than anything the 0s.

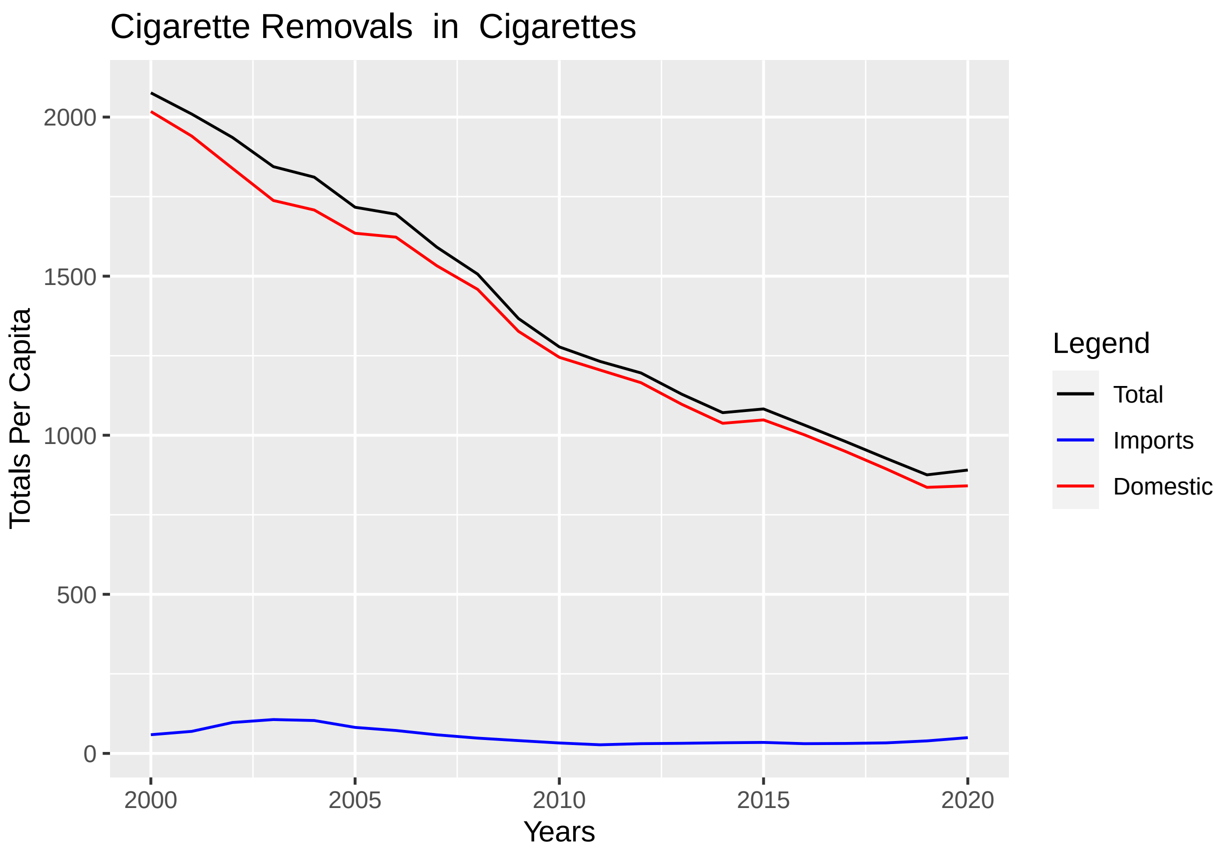
```
totalsPerCapita<-Products[[1]]%>%select(11)/Products[[1]]$Population
importsPerCapita<-Products[[1]]%>%select(10)/Products[[1]]$Population
domesticPerCapita<-Products[[1]]%>%select(9)/Products[[1]]$Population
for(j in c(2:13)){
  totalsPerCapita<-cbind(totalsPerCapita,Products[[j]]%>%select(11)/Products[[j]]$Population)
  importsPerCapita<-cbind(importsPerCapita,Products[[j]]%>%select(10)/Products[[j]]$Population)
  domesticPerCapita<-cbind(domesticPerCapita,Products[[j]]%>%select(9)/Products[[j]]$Population)
}
```

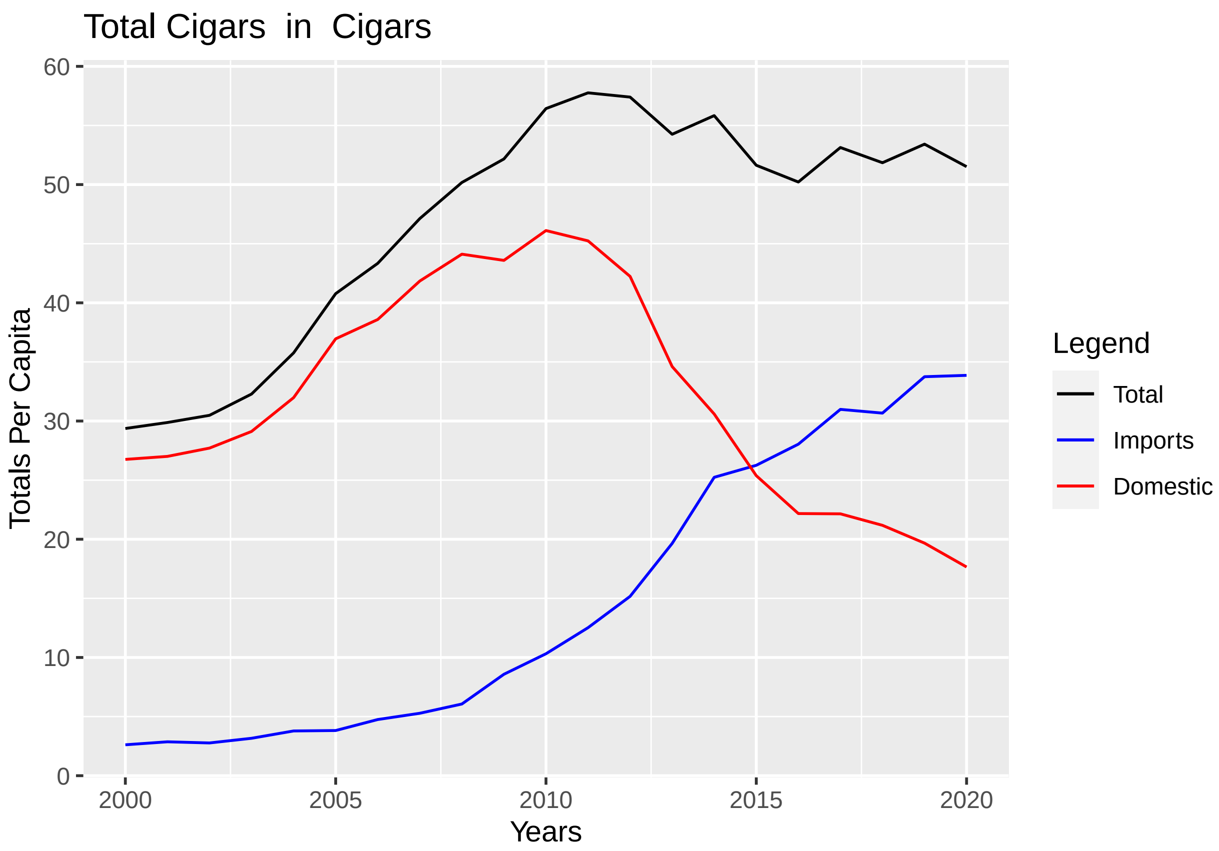
From the next plots we can extract some information:

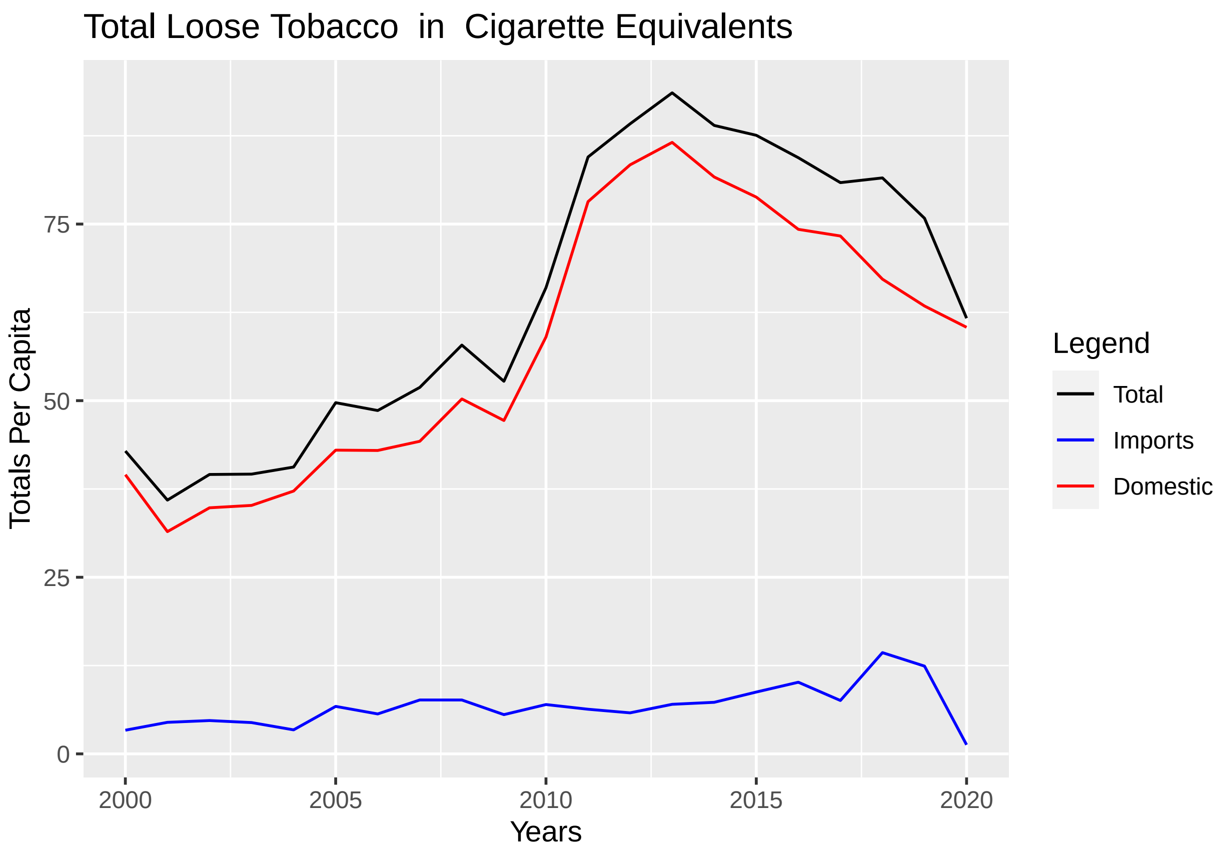
- Something happened in 2008 that affected tobacco consumption in most of its forms.
- From snuff and chewing tobacco (the only 2 forms of noncombustible tobacco), it seems like chewing tobacco is loosing popularity while snuff gaining.
- Both cigars and large cigars are preferable from imports to domestic ones. This is important to point out since for every other product there is a big preference for domestic production.

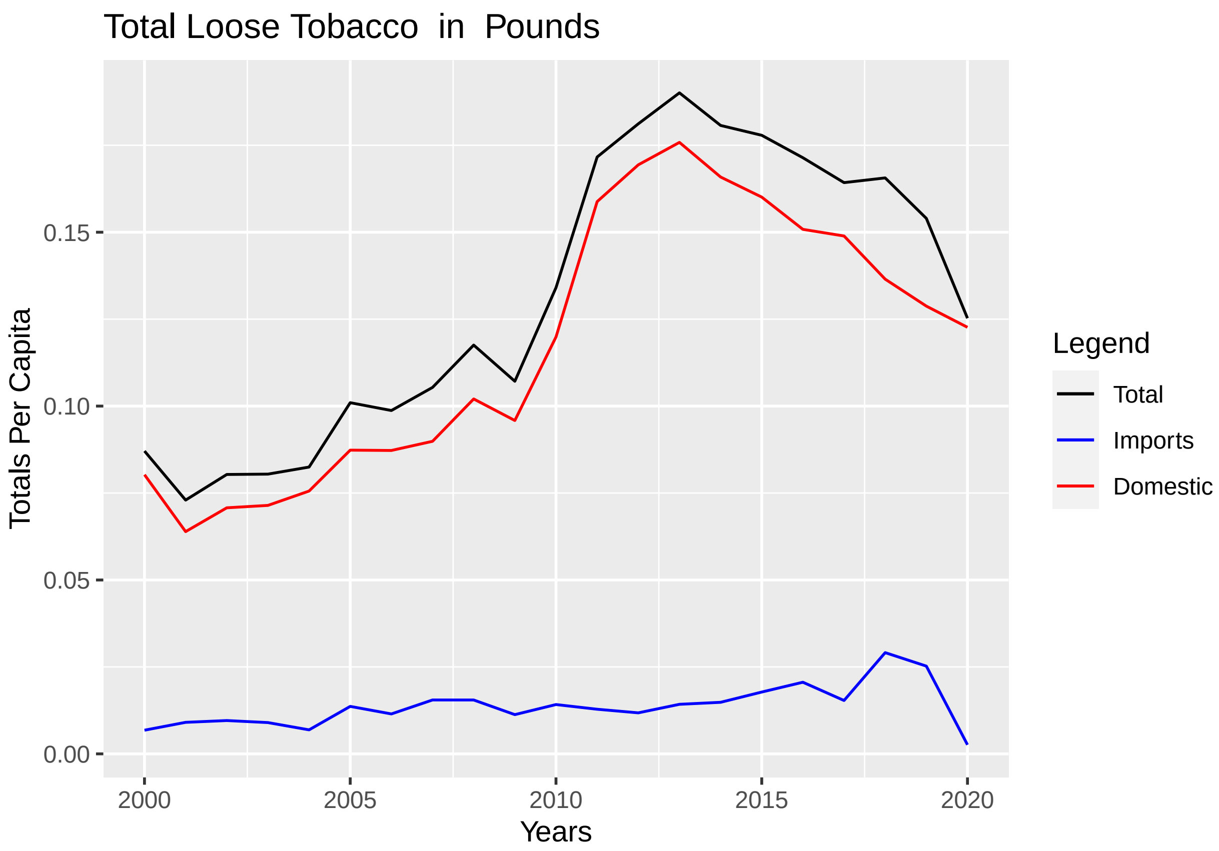
```
for(i in c(1:13)){
  print(
      ggplot(data=Products[[i]], aes(x=c(2000:2020))) +
      geom_line(aes(y =totalsPerCapita[[i]],color='Total'))+
      geom_line(aes(y=importsPerCapita[[i]],color='Imports'))+
      geom_line(aes(y=domesticPerCapita[[i]],color='Domestic'))+
      xlab('Years')+ylab('Totals Per Capita')+
      labs(title=names(Products)[i])+
      scale_color_manual(name='Legend',values = c('Total' = "black", "Imports" = "blue",'Domestic'='red
}
```

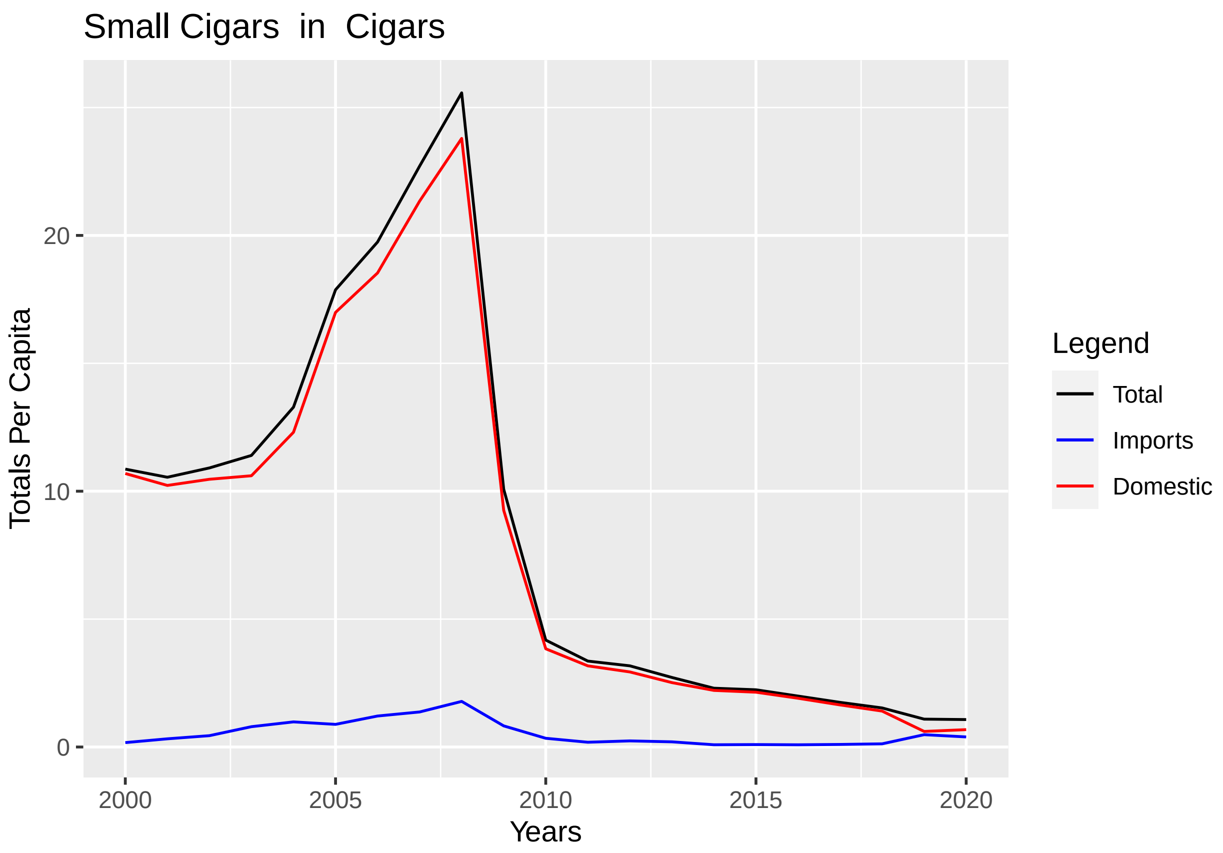

Chewing Tobacco in Pounds
Totals Per Capita
0.20
0.15
0.10
0.05
0.00
2000
2005
2010
2015
2020
Years
Legend
Total
Imports
Domestic

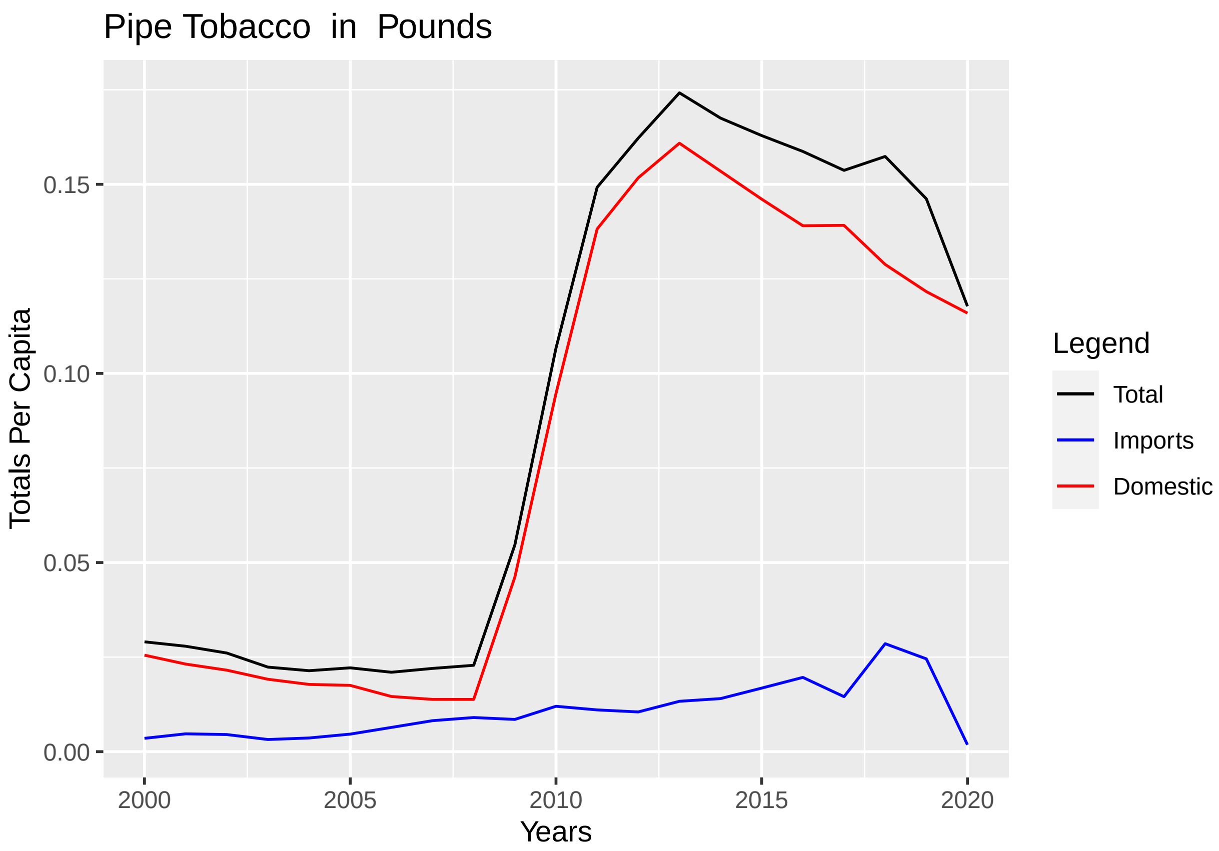

Cigarette Removals in Cigarettes
Totals Per Capita
2000
1500
1000
500
0
2000
2005
2010
2015
2020
Years
Legend
Total
Imports
Domestic

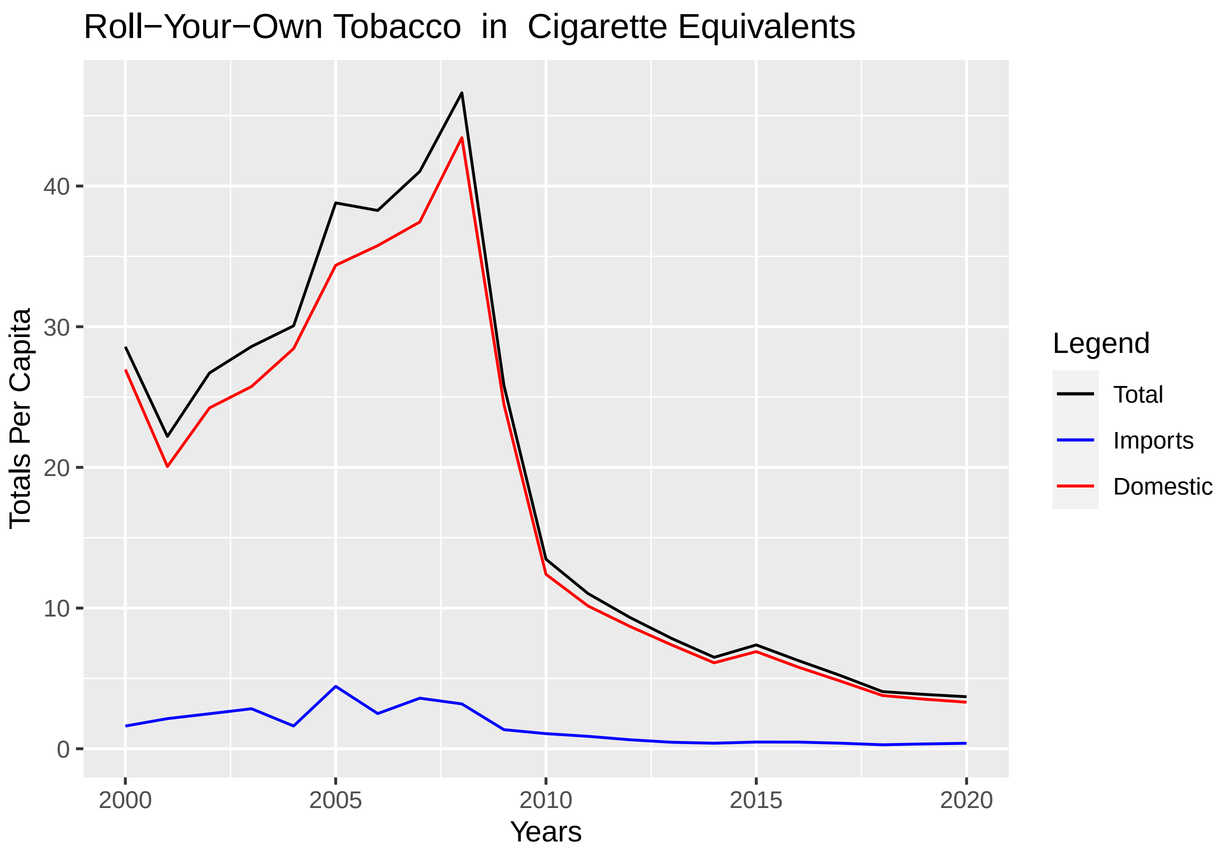

Total Cigars in Cigars
Totals Per Capita
60
50
40
30
20
10
0
2000
2005
2010
2015
2020
Years
Legend
Total
Imports
Domestic

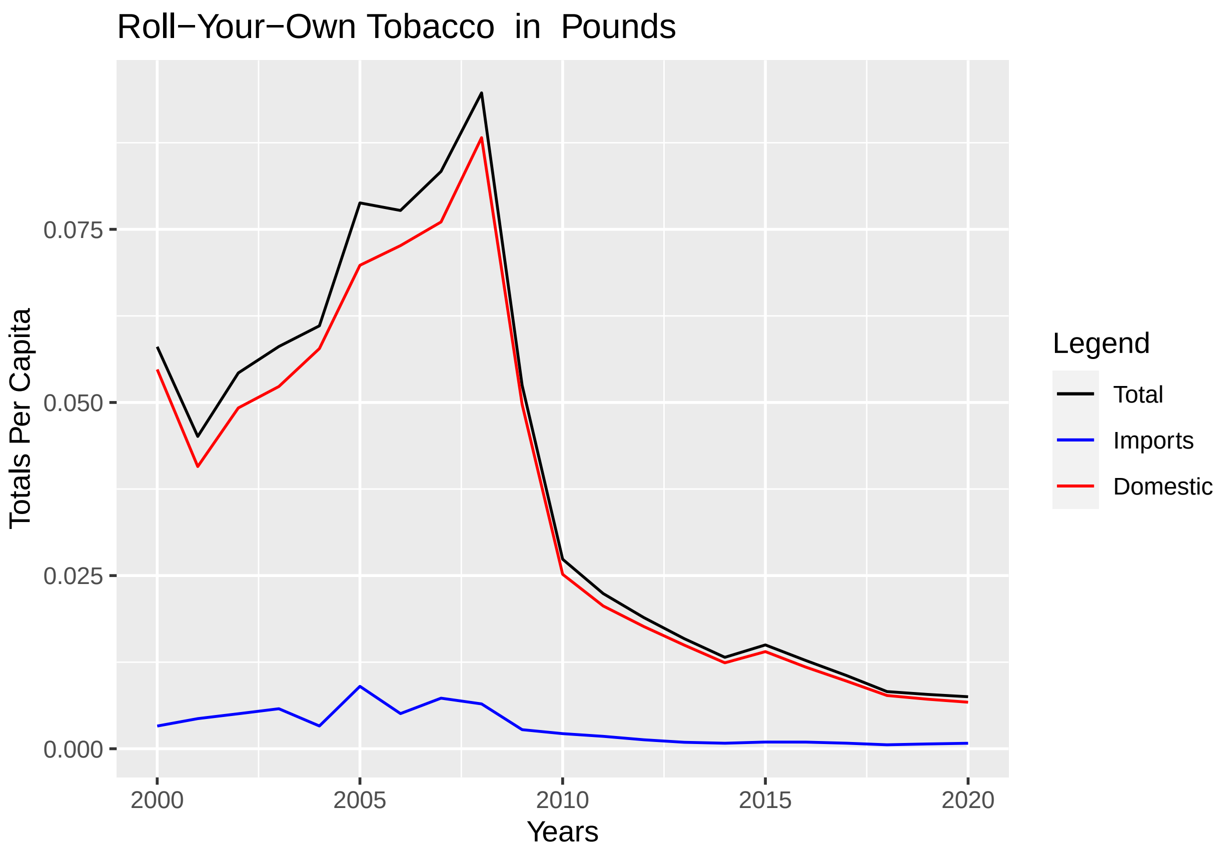

Total Loose Tobacco in Cigarette Equivalents
Totals Per Capita
75
50
25
0
2000
2005
2010
2015
2020
Years
Legend
Total
Imports
Domestic

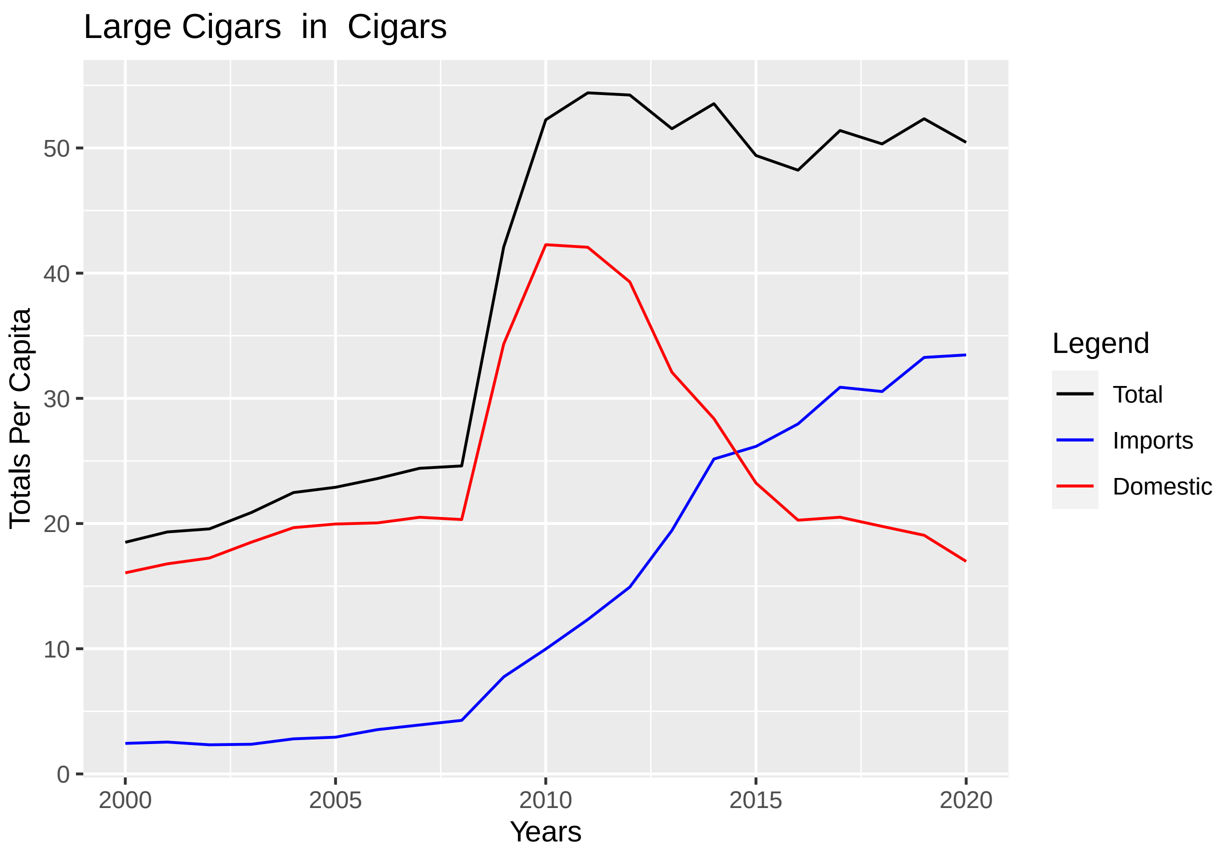

Total Loose Tobacco in Pounds
Totals Per Capita
0.00
0.05
0.10
0.15
2000
2005
2010
2015
2020
Years
Legend
Total
Imports
Domestic

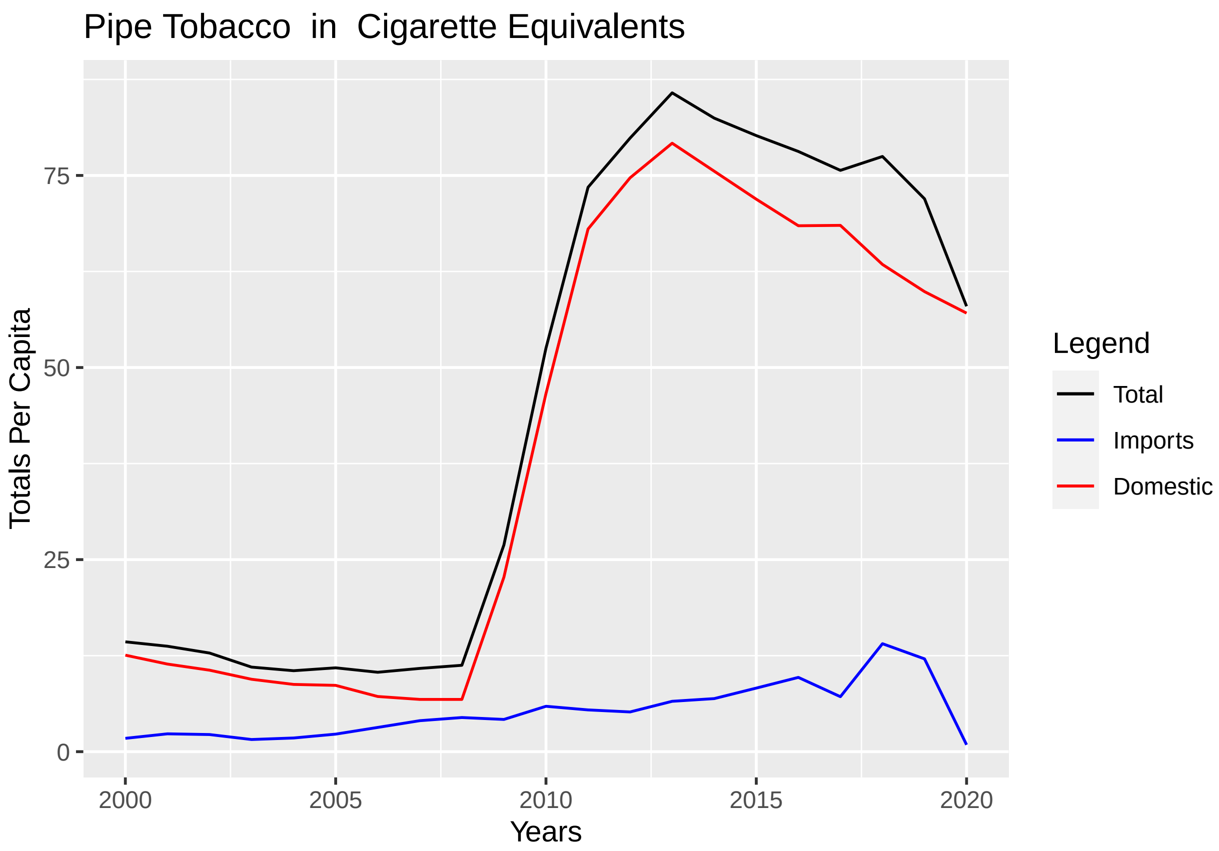

Small Cigars in Cigars
Totals Per Capita
0
10
20
2000
2005
2010
2015
2020
Years
Legend
Total
Imports
Domestic

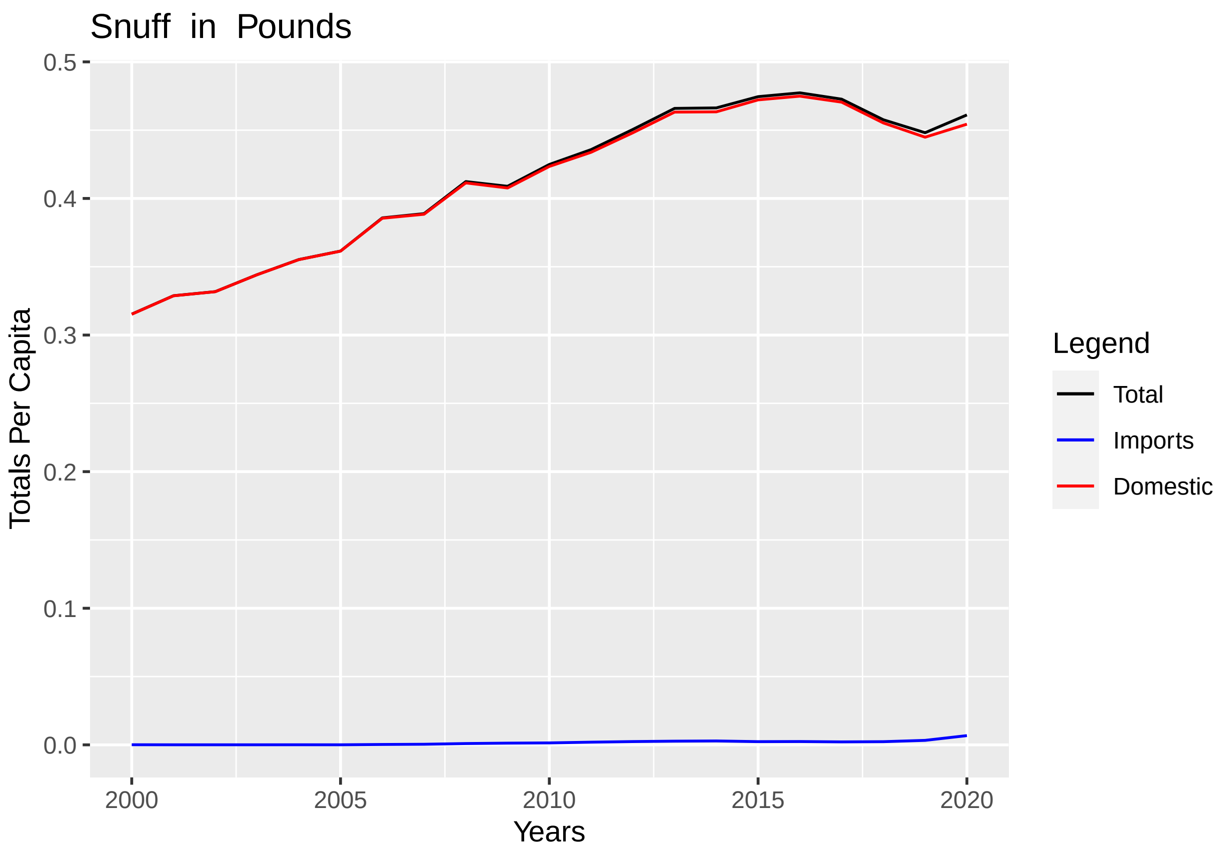

Pipe Tobacco in Pounds
Totals Per Capita
0.00
0.05
0.10
0.15
2000
2005
2010
2015
2020
Years
Legend
Total
Imports
Domestic

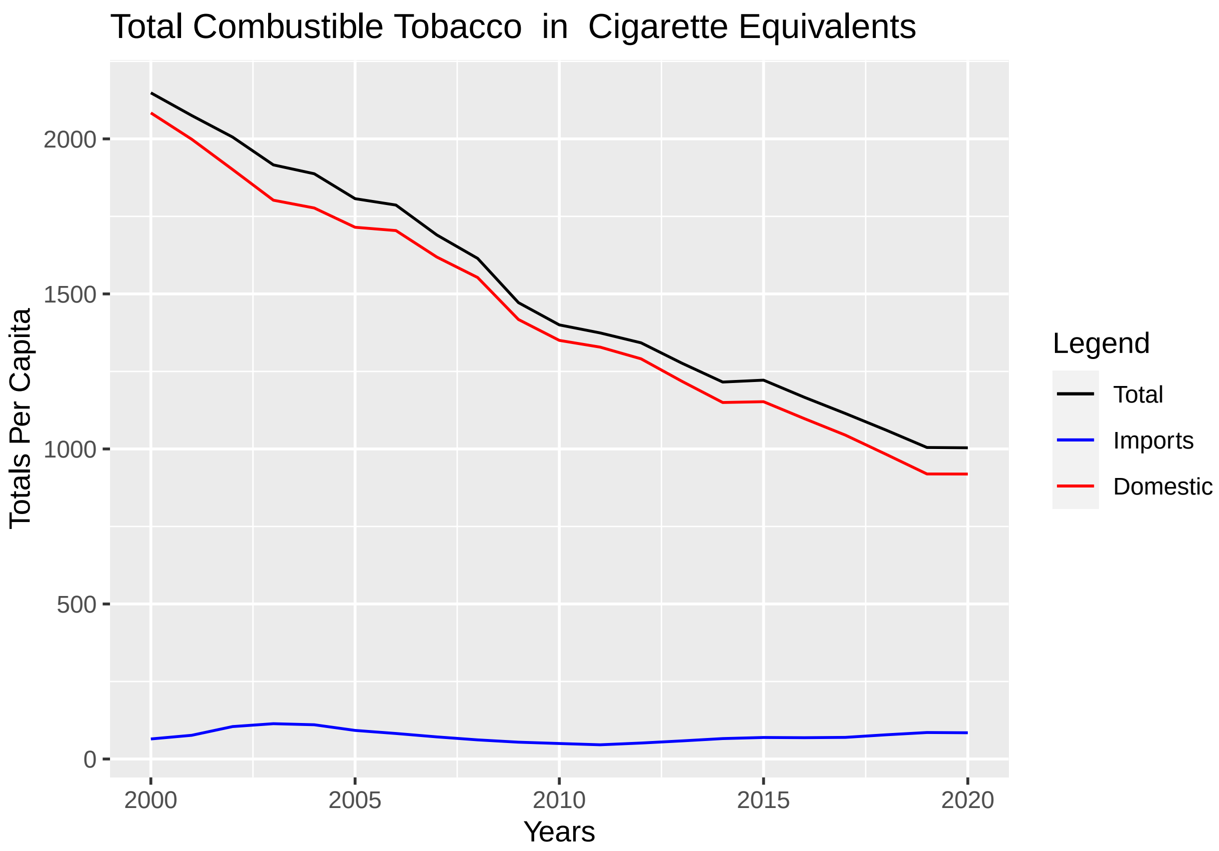

Roll−Your−Own Tobacco in Cigarette Equivalents
Totals Per Capita
40
30
20
10
0
2000
2005
2010
2015
2020
Years
Legend
Total
Imports
Domestic


Roll−Your−Own Tobacco in Pounds
Totals Per Capita
0.000
0.025
0.050
0.075
2000
2005
2010
2015
2020
Years
Legend
Total
Imports
Domestic


Large Cigars in Cigars
Totals Per Capita
0
10
20
30
40
50
2000
2005
2010
2015
2020
Years
Legend
Total
Imports
Domestic


Pipe Tobacco in Cigarette Equivalents
Totals Per Capita
75
50
25
0
2000
2005
2010
2015
2020
Years
Legend
Total
Imports
Domestic


Snuff in Pounds
Totals Per Capita
0.5
0.4
0.3
0.2
0.1
0.0
2000
2005
2010
2015
2020
Years
Legend
Total
Imports
Domestic



Generating Training and Testing data

To start with the prediction section, first its needed to create a training and testing data. We ll use data from 2000 to 2016 as training and from 2017 to 2020 for testing.

```
trainTotals<-list()
testTotals<-list()
for(i in c(1:13)){
  trainTotals<-c(trainTotals, list(ts(head(totalsPerCapita[[i]],17),start=c(2000),end=c(2016),frequency
  testTotals<-c(testTotals, list(ts(tail(totalsPerCapita[[i]],4),start=c(2017),end=c(2020),frequency =
}
```

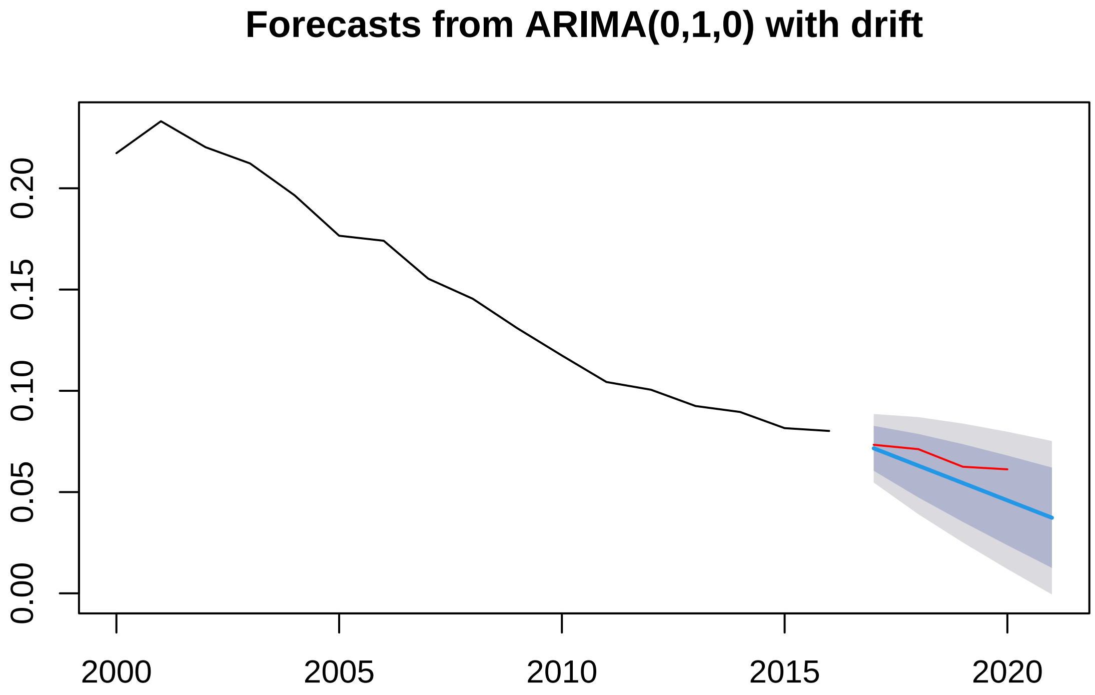
Function to get the Mean Squared Error from 2 vectors

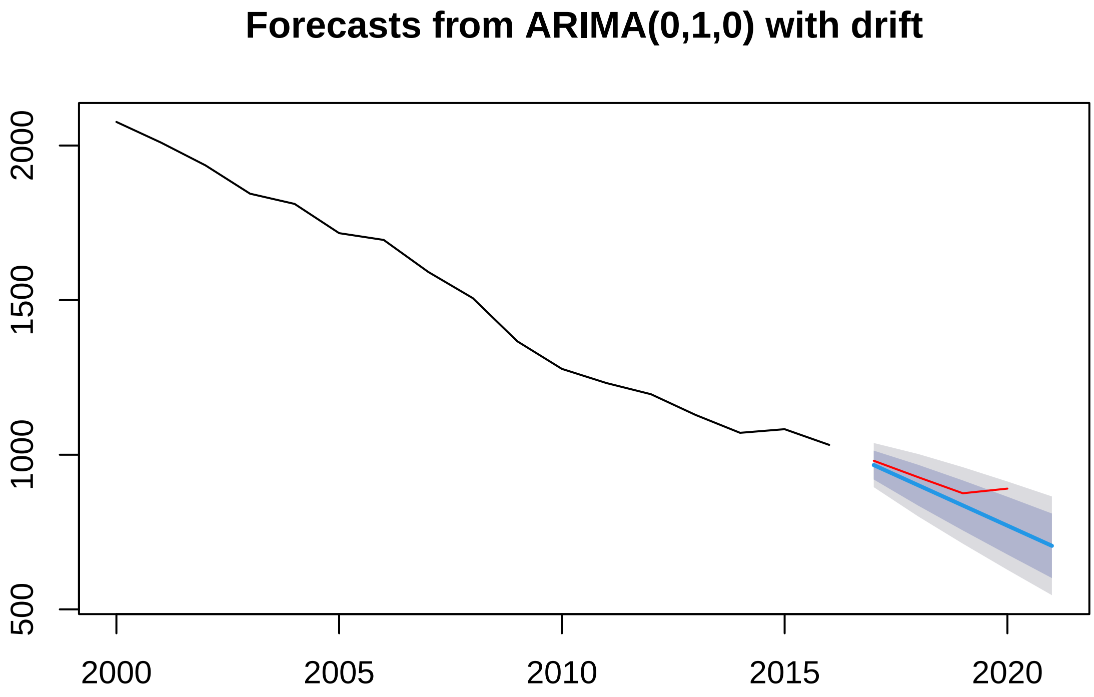
```
MSE<- function (v1,v2){
  return(sum((v1-v2)^2)/length(v1))
}
```

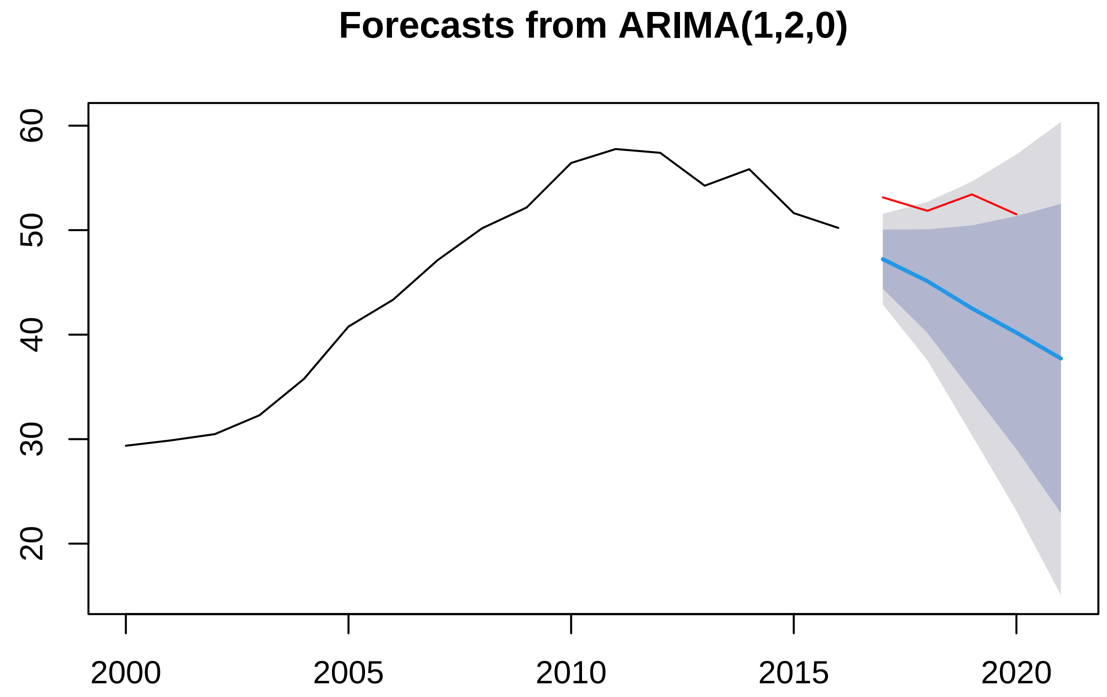
Now everything is ready to start with the models.

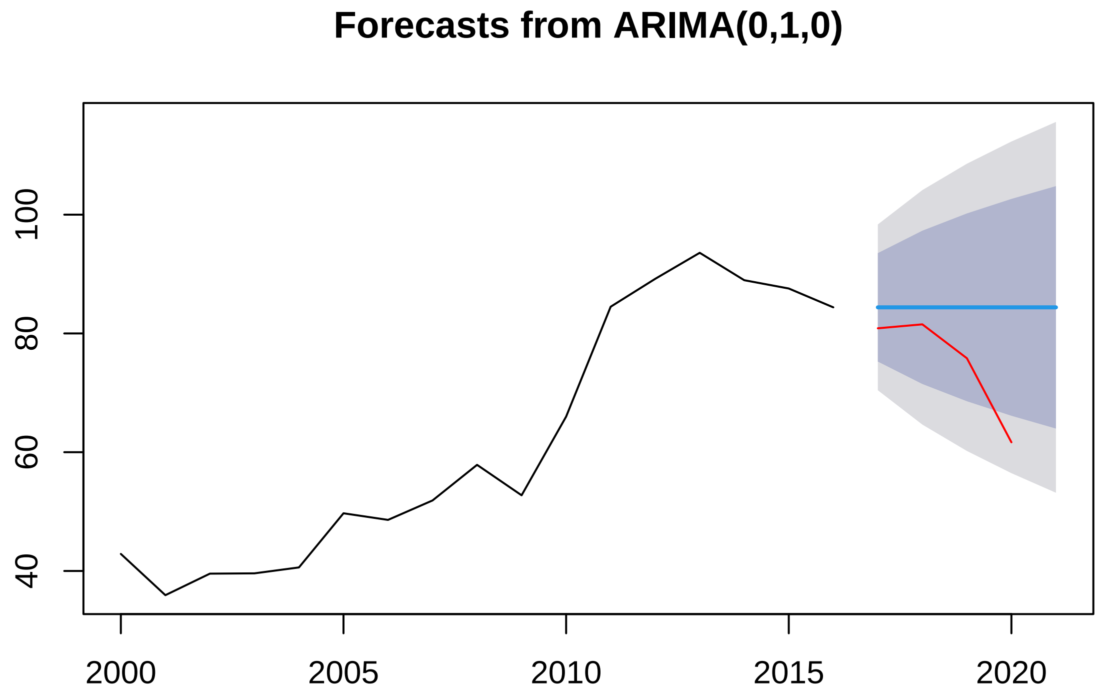
Auto-ARIMA

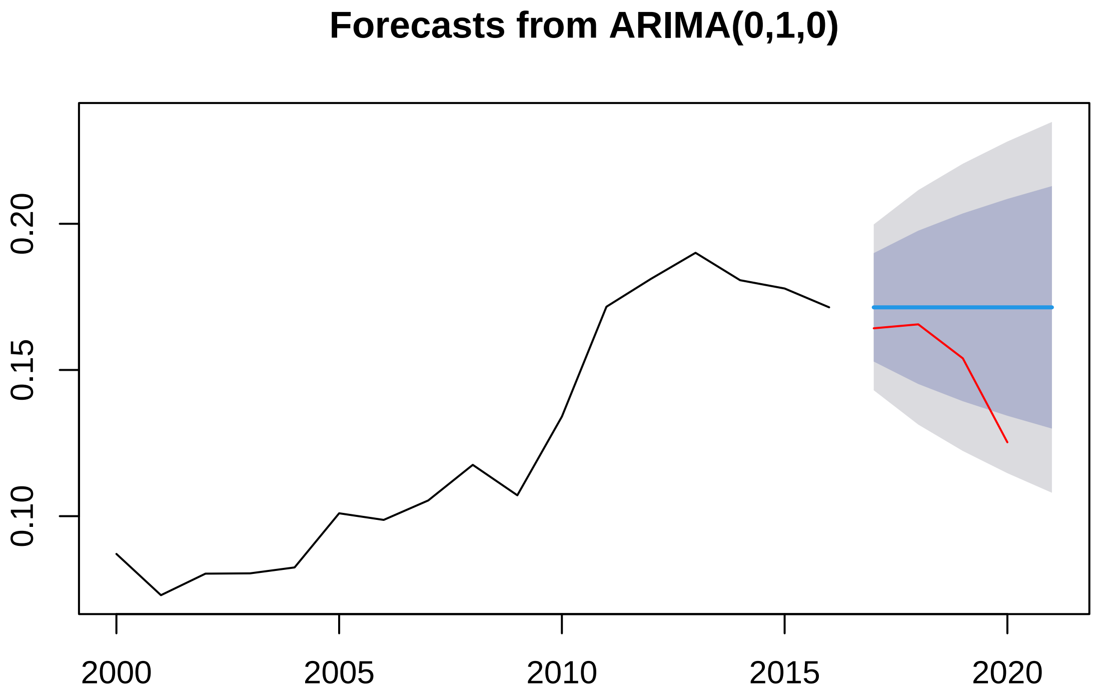
```
arimaErrors<-c()
for(i in c(1:13)){
  #Training and making forecast until 2021 using AUTO-Arima
  sarima_ts<-auto.arima(trainTotals[[i]])
  arima_model<-forecast::forecast(sarima_ts,h=5)

  #Plotting prediction and testing data (red for testing data)
  plot(arima_model)
  lines(testTotals[[i]],col='red')

  #Getting MSE (the head and tail are used to get from 2017-2020)
  prediction<-arima_model$fitted%>%as.numeric()%>%tail(5)%>%head(4)
  test<-testTotals[[i]]%>%as.numeric()

  #Saving MSE in arimaError vector
  arimaErrors<-c(arimaErrors,MSE(prediction,test))
}
```

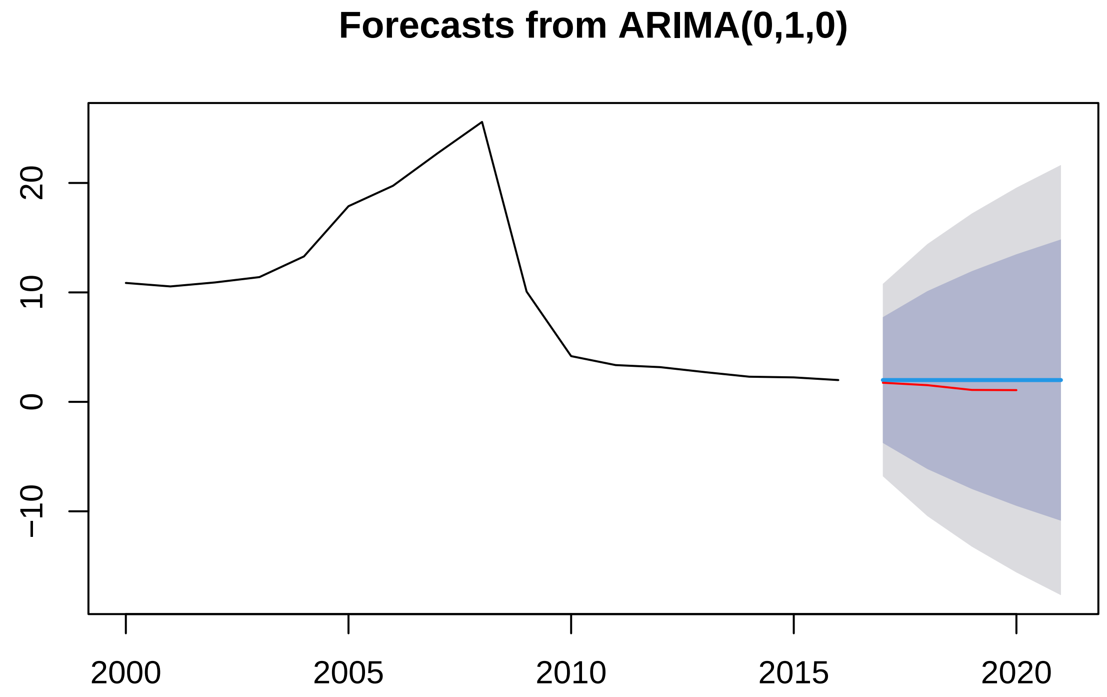


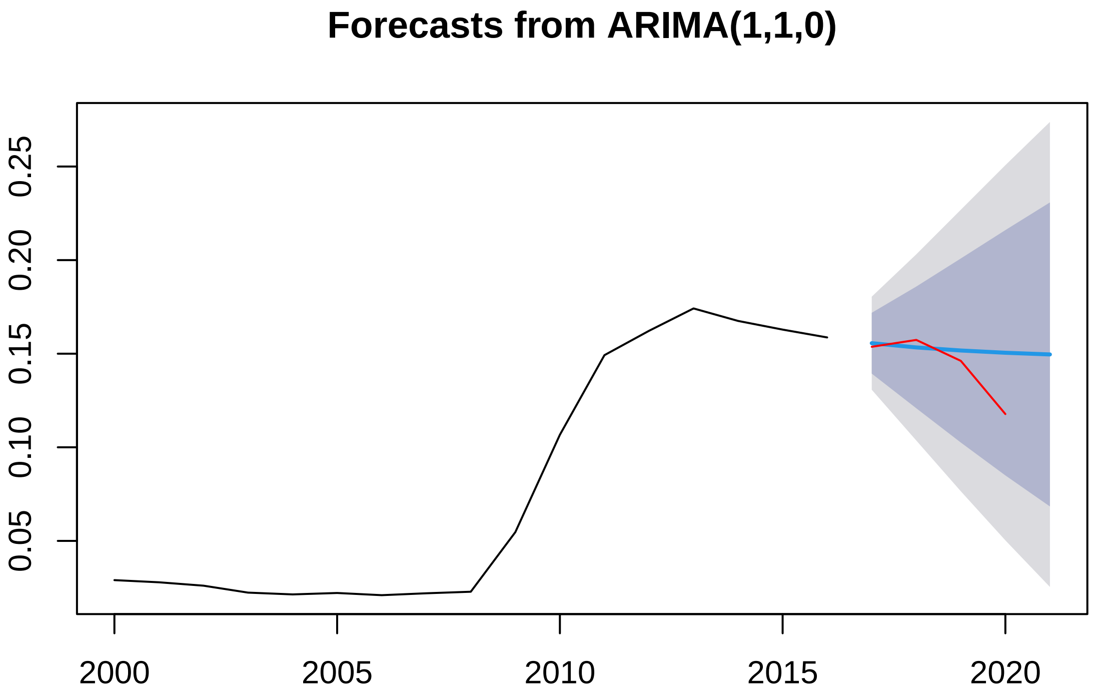

Forecasts from ARIMA(0,1,0) with drift
500
1000
1500
2000
2000
2005
2010
2015
2020

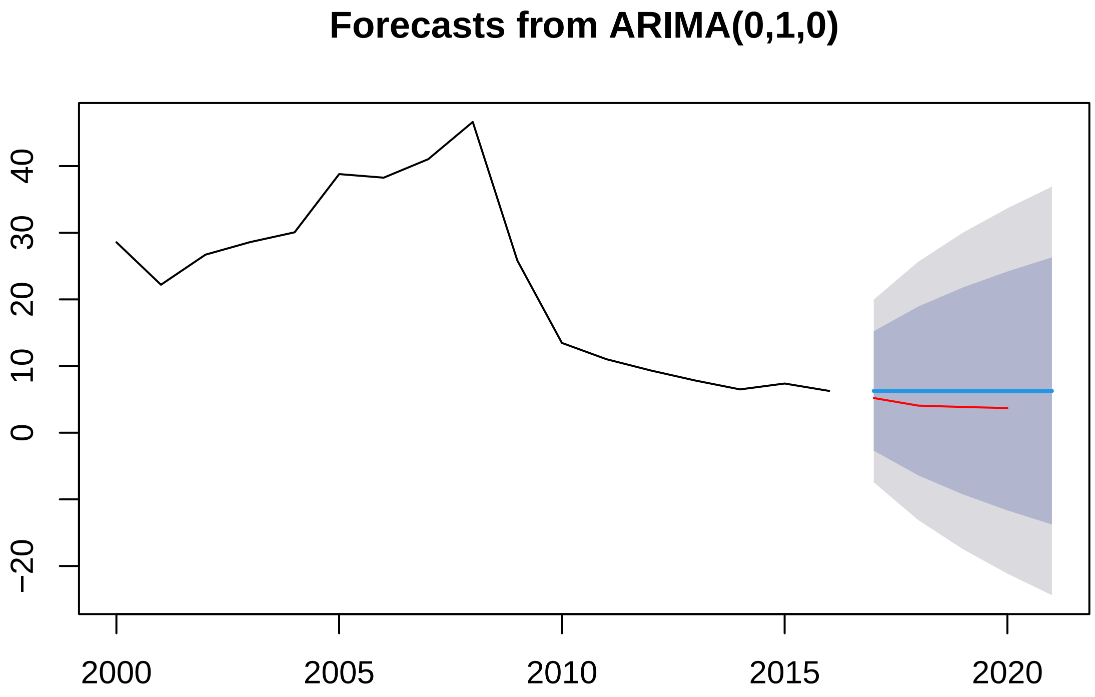

Forecasts from ARIMA(1,2,0)
60
50
40
30
20
2000
2005
2010
2015
2020

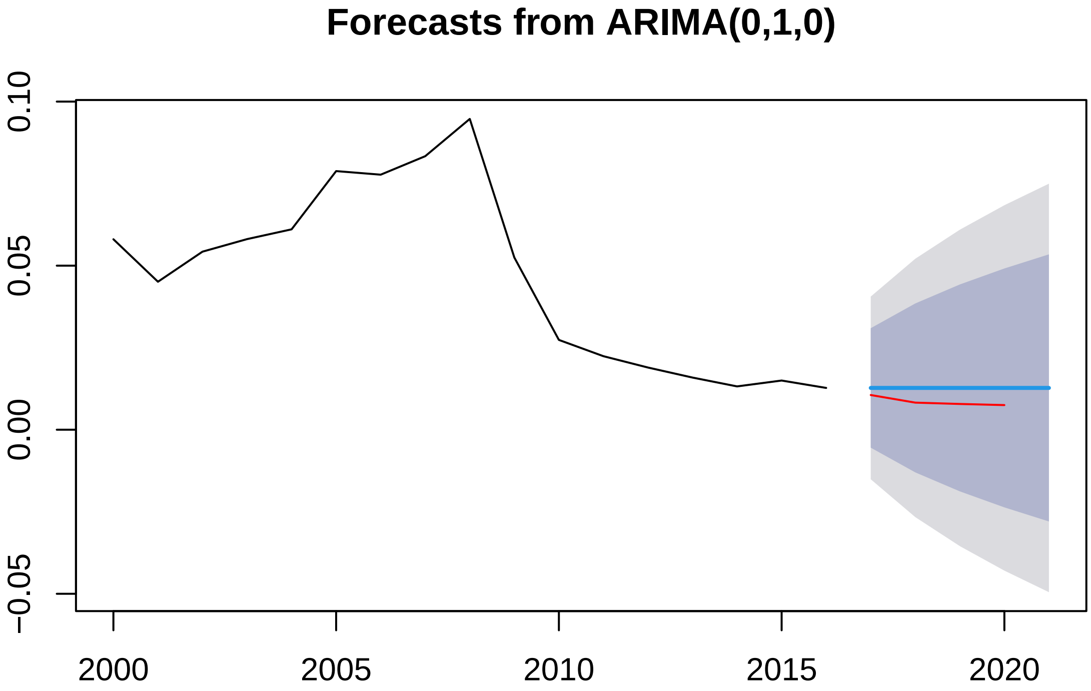

Forecasts from ARIMA(0,1,0)
100
80
60
40
2000
2005
2010
2015
2020

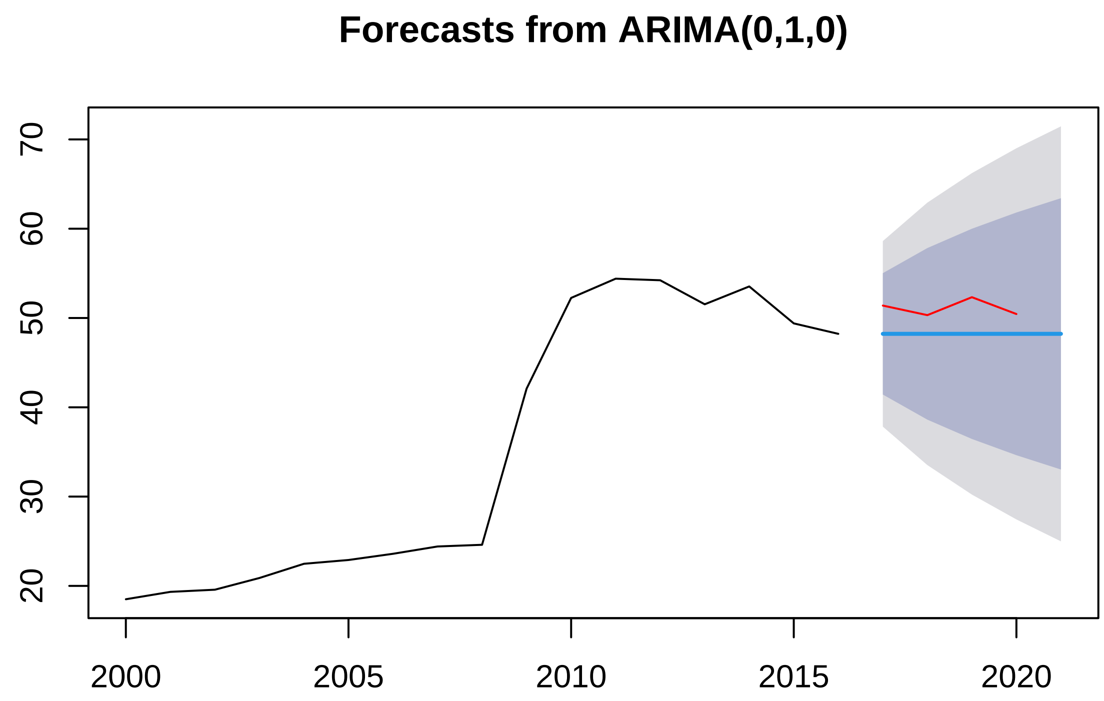

Forecasts from ARIMA(0,1,0)
0.10
0.15
0.20
2000
2005
2010
2015
2020

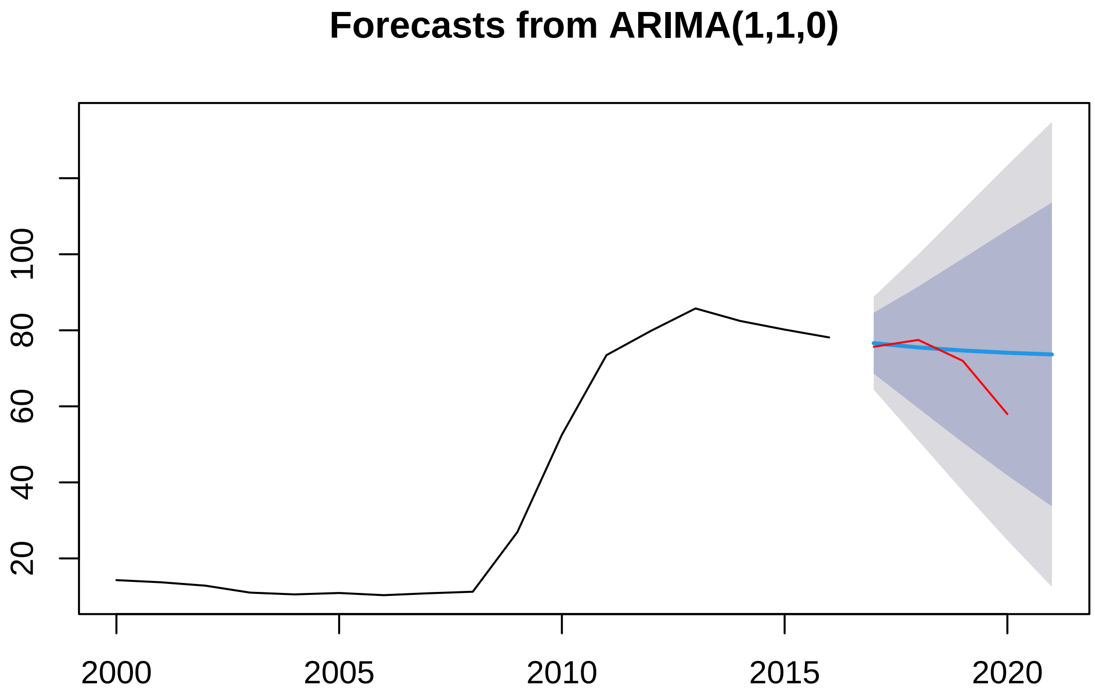

Forecasts from ARIMA(0,1,0)
20
10
0
−10
2000
2005
2010
2015
2020

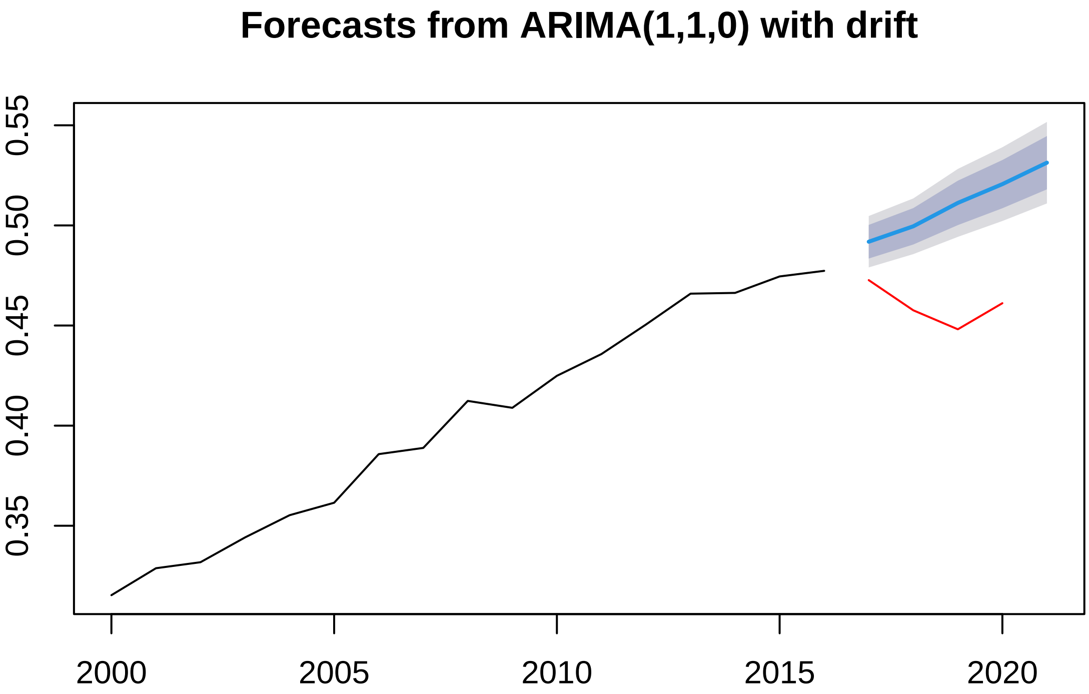

Forecasts from ARIMA(1,1,0)
0.05
0.10
0.15
0.20
0.25
2000
2005
2010
2015
2020


Forecasts from ARIMA(0,1,0)
40
30
20
10
0
−20
2000
2005
2010
2015
2020

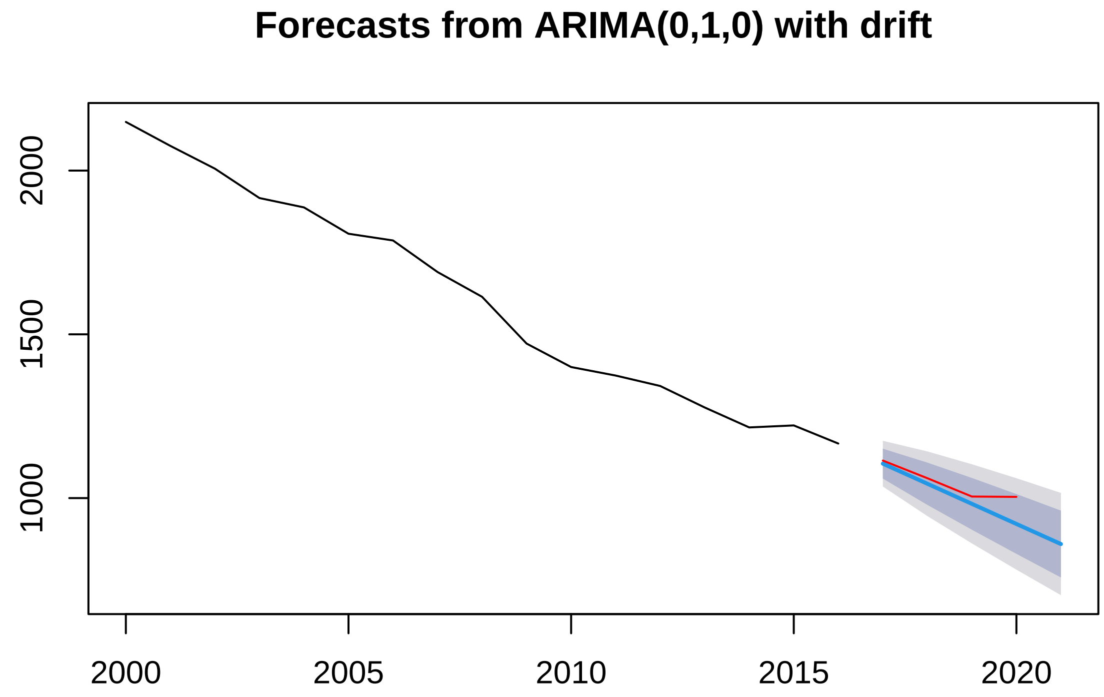

Forecasts from ARIMA(0,1,0)
0.10
0.05
0.00
−0.05
2000
2005
2010
2015
2020


Forecasts from ARIMA(0,1,0)
70
60
50
40
30
20
2000
2005
2010
2015
2020


Forecasts from ARIMA(1,1,0)
20
40
60
80
100
2000
2005
2010
2015
2020

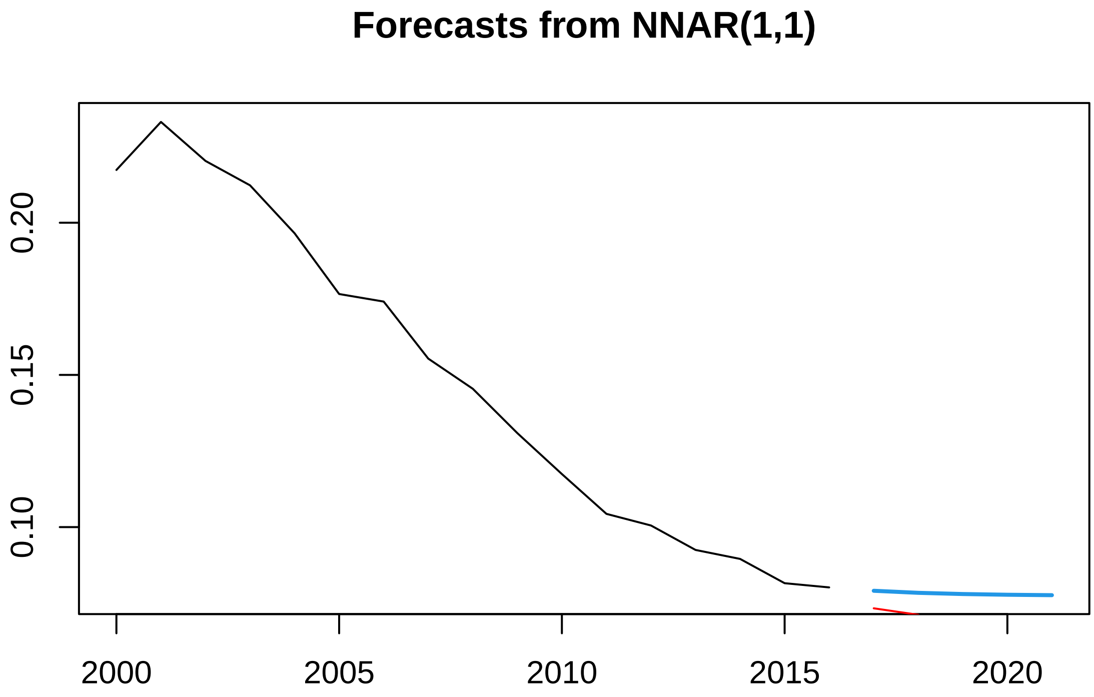


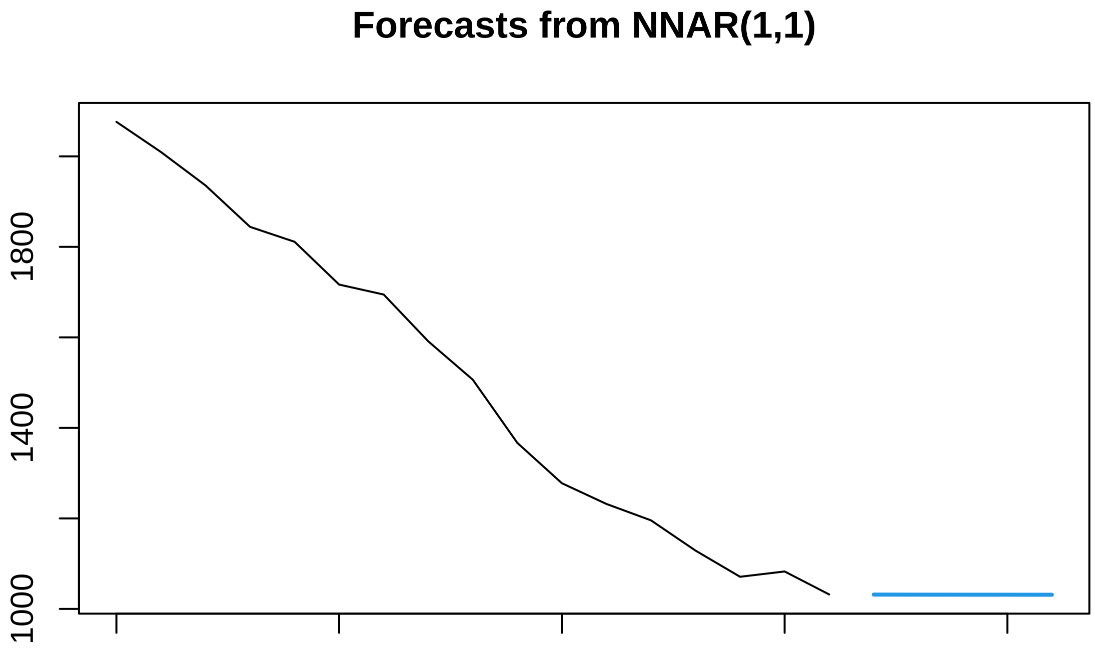
Forecasts from ARIMA(1,1,0) with drift

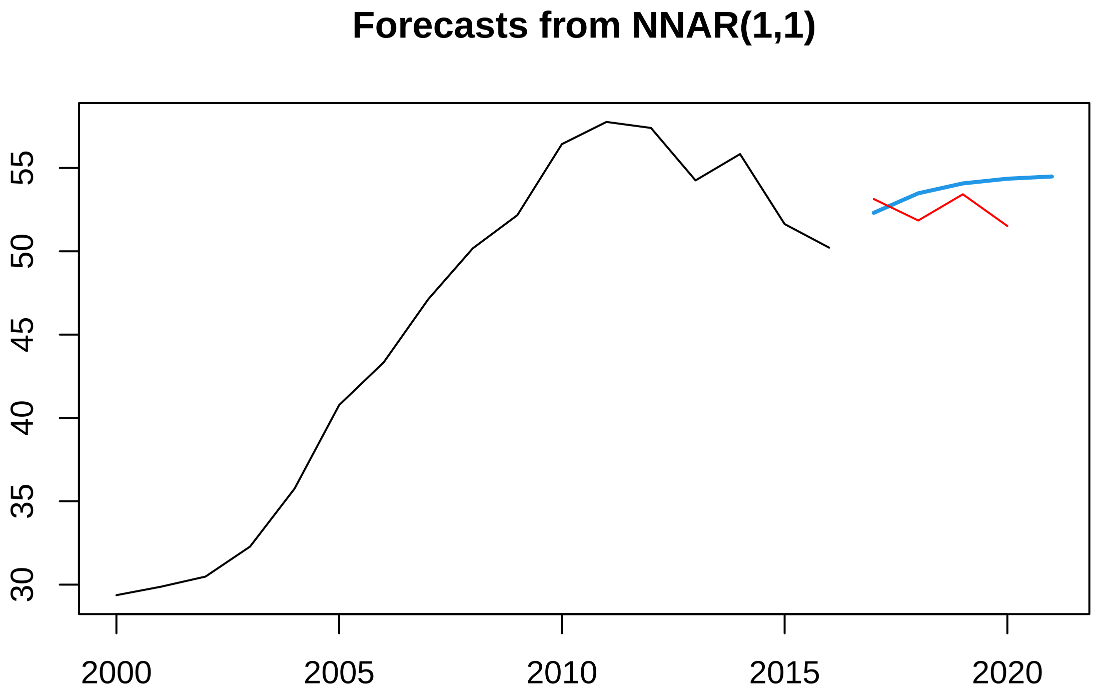


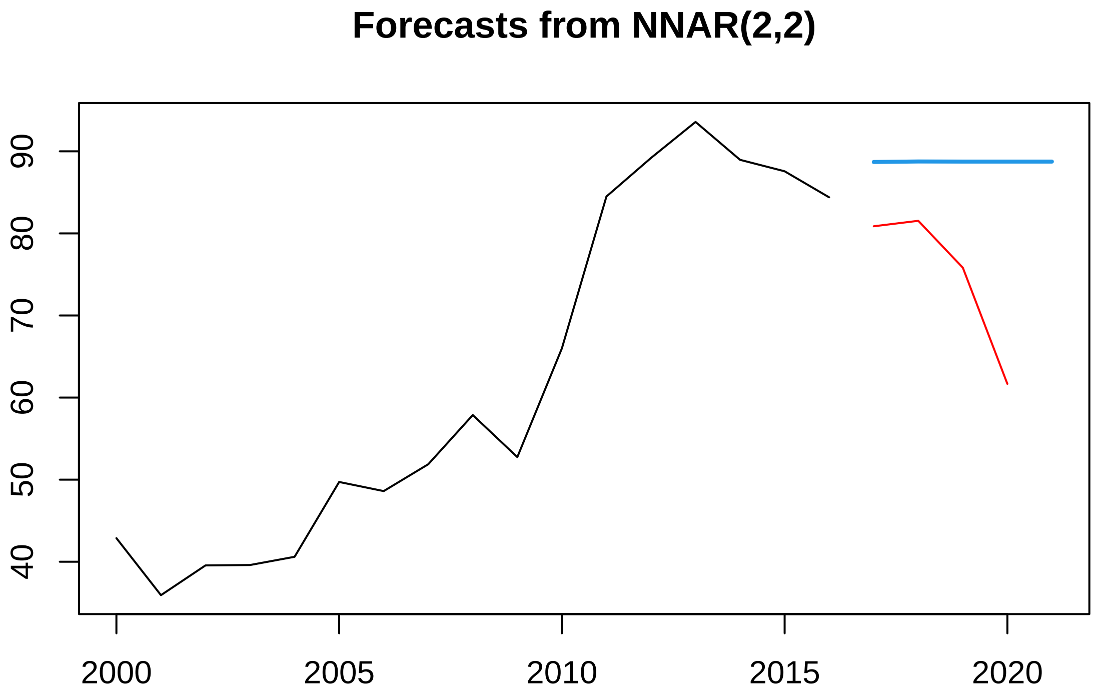
Neural Network Autoregression

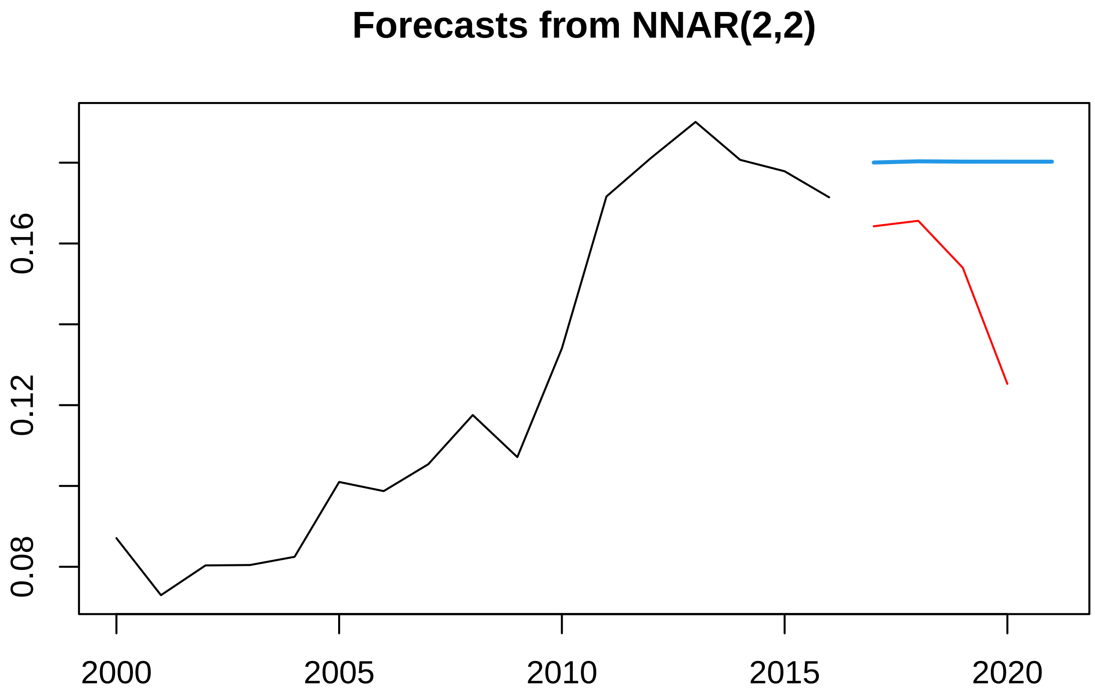
```
nnErrors<-c()
for(i in c(1:13)){
  #Training model
  fit<-nnetar(trainTotals[[i]],lambda='auto')
  nn_model<-forecast::forecast(fit,h=5)

  #Plotting prediction and testing data (red for testing data)
  plot(nn_model)
  lines(testTotals[[i]],col='red')

  #Getting MSE (the head and tail are used to get from 2017-2020)
  prediction<-nn_model$fitted%>%as.numeric()%>%tail(5)%>%head(4)
  test<-testTotals[[i]]%>%as.numeric()

  #Saving MSE in nnError vector
  nnErrors<-c(nnErrors,MSE(prediction,test))
}
```


Forecasts from NNAR(1,1)
0.20
0.15
0.10
2000
2005
2010
2015
2020

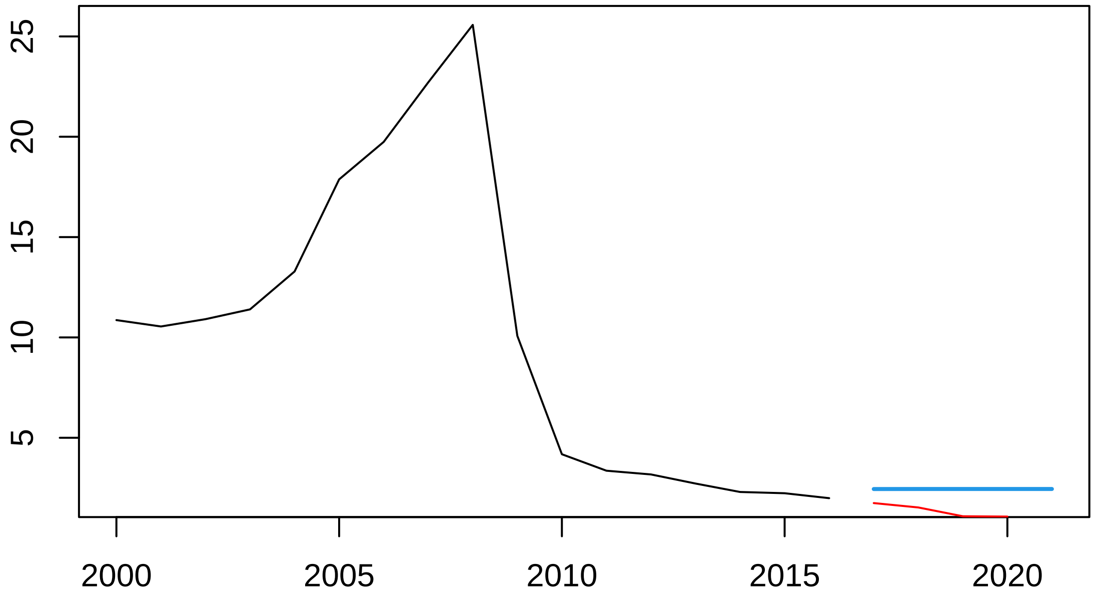

Forecasts from NNAR(1,1)
1000
1400
1800
2000
2005
2010
2015
2020

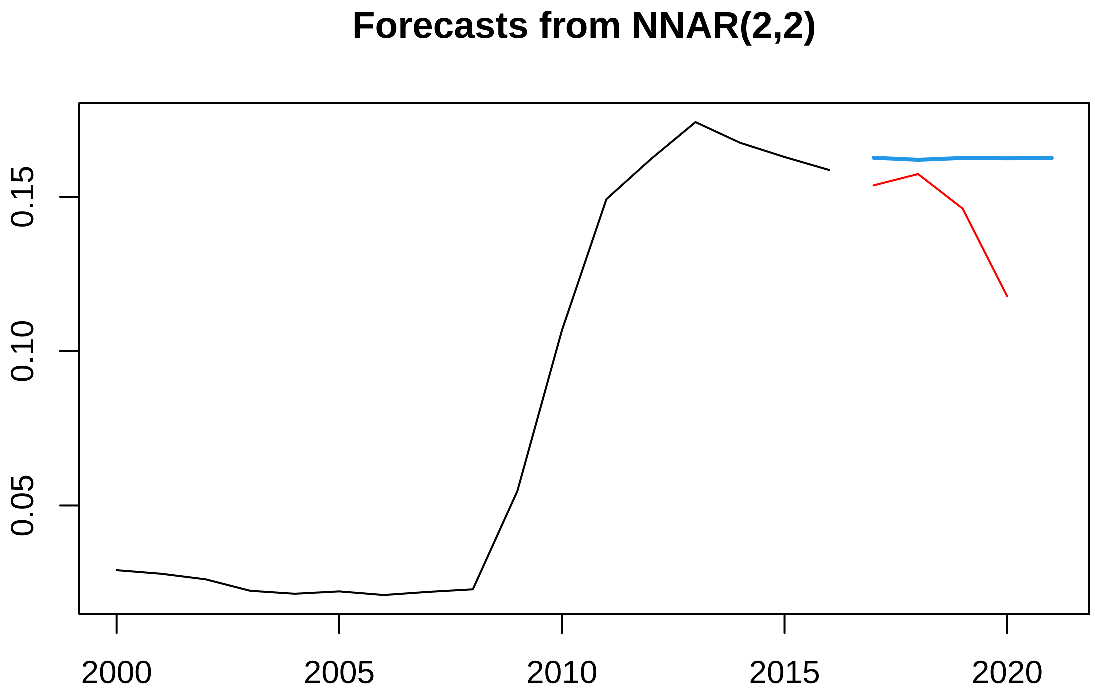

Forecasts from NNAR(1,1)
55
50
45
40
35
30
2000
2005
2010
2015
2020

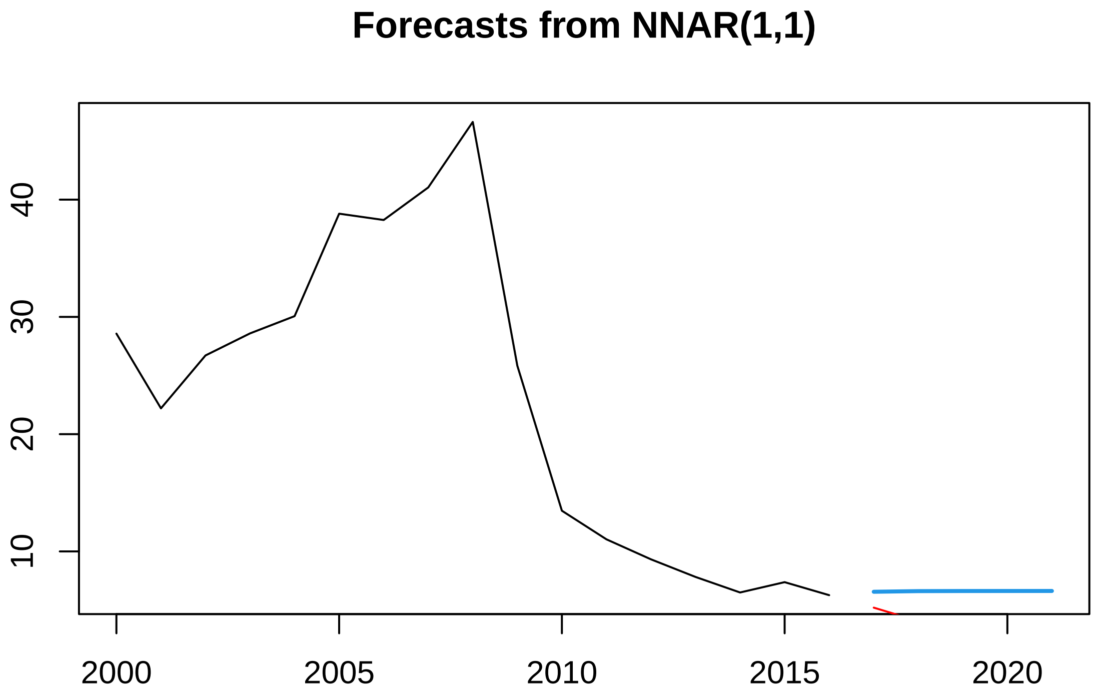

Forecasts from NNAR(2,2)
90
80
70
60
50
40
2000
2005
2010
2015
2020

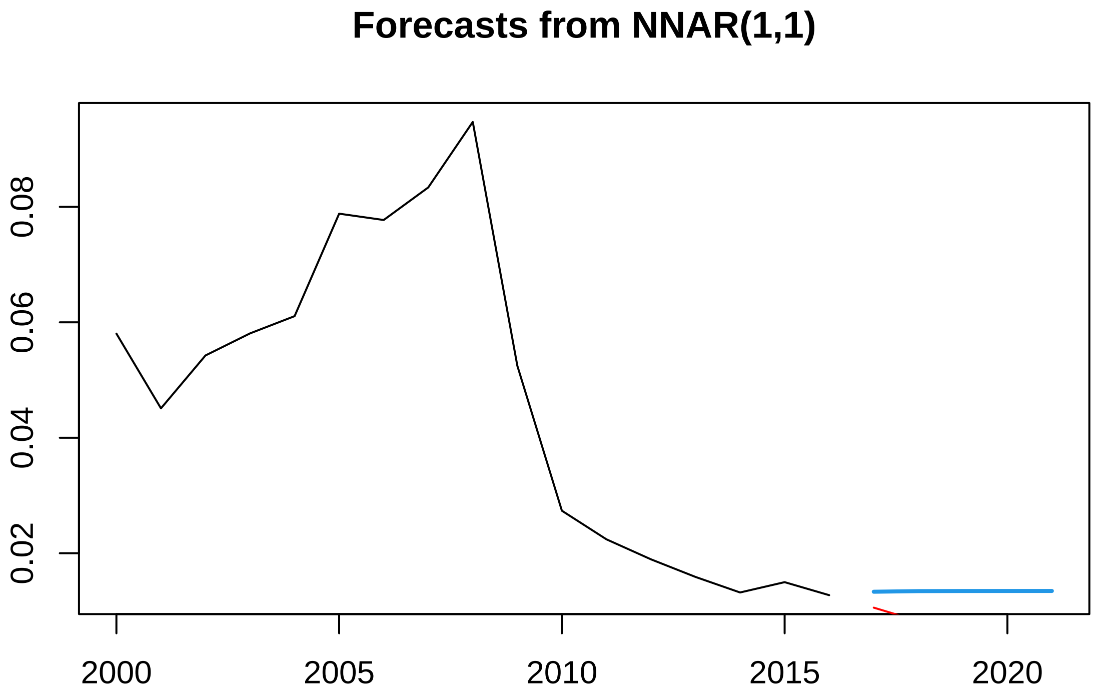

Forecasts from NNAR(2,2)
0.08
0.12
0.16
2000
2005
2010
2015
2020

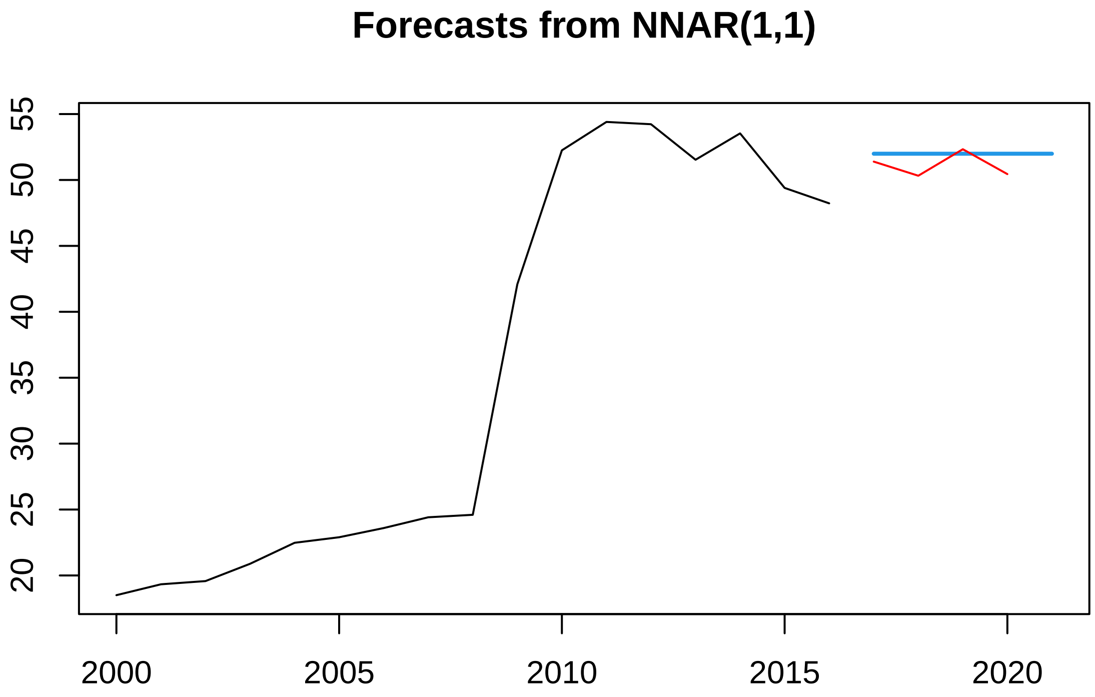
Forecasts from NNAR(1,1)

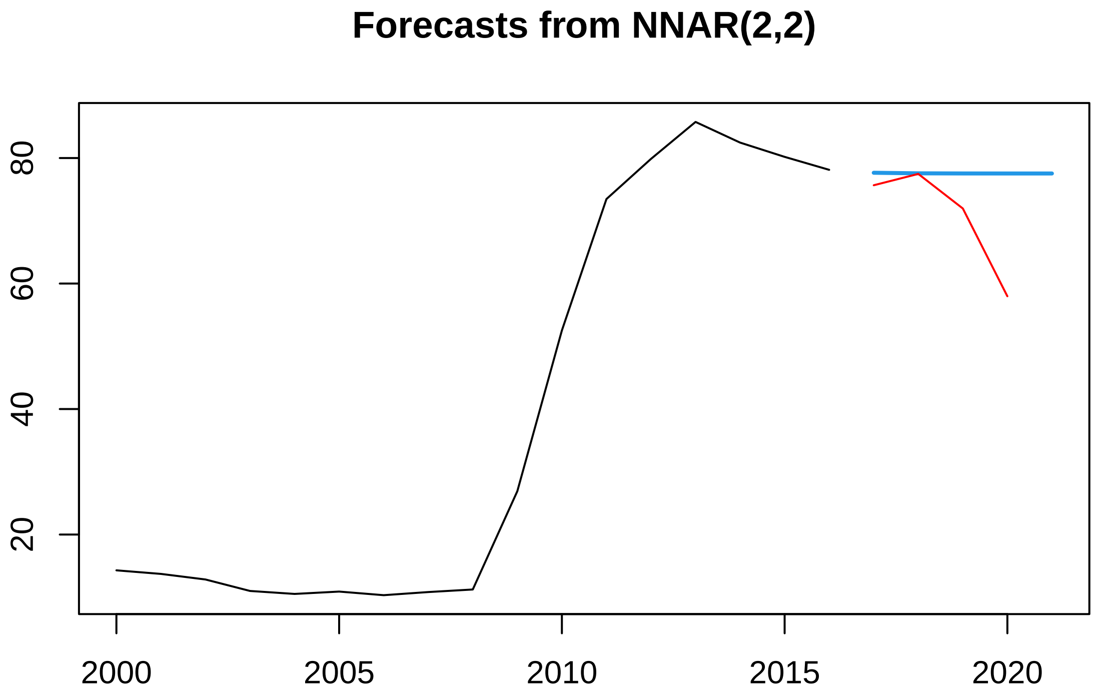


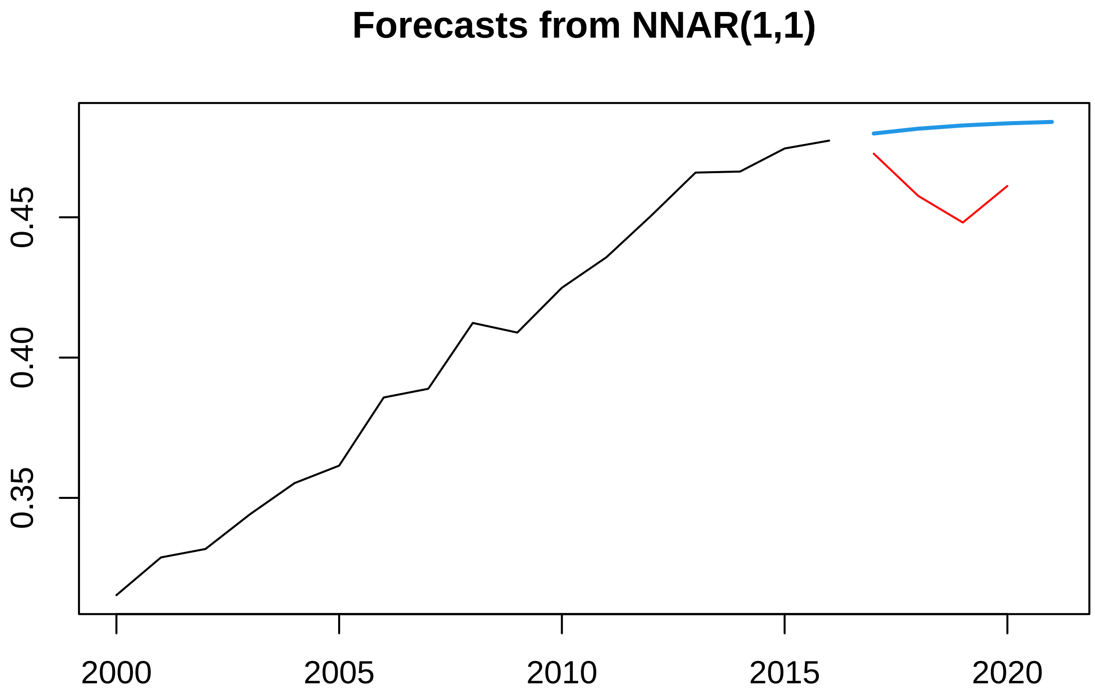

Forecasts from NNAR(2,2)
0.15
0.10
0.05
2000
2005
2010
2015
2020

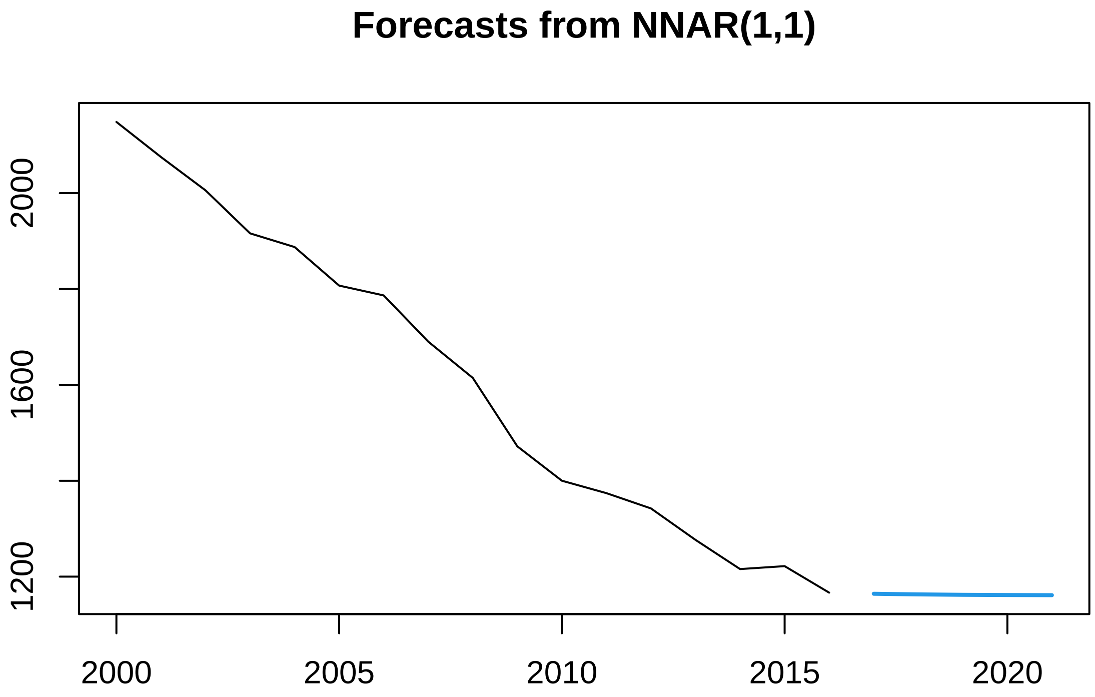

Forecasts from NNAR(1,1)
40
30
20
10
2000
2005
2010
2015
2020


Forecasts from NNAR(1,1)
0.02
0.04
0.06
0.08
2000
2005
2010
2015
2020


Forecasts from NNAR(1,1)
55
50
45
40
35
30
25
20
2000
2005
2010
2015
2020


Forecasts from NNAR(2,2)
80
60
40
20
2000
2005
2010
2015
2020

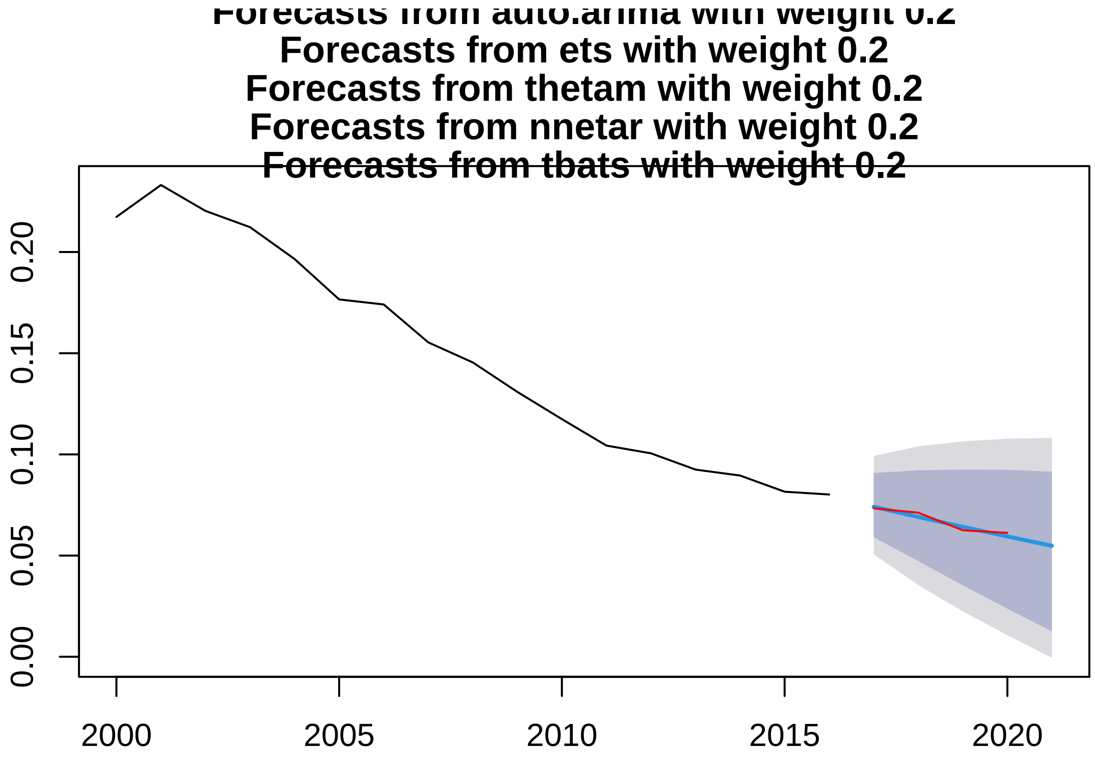

Forecasts from NNAR(1,1)
0.35
0.40
0.45
2000
2005
2010
2015
2020



Hybrid Model

```
hybErrors<-c()
for(i in c(1:13)){
  #Training and making forecast
  hyb_mod<- hybridModel(trainTotals[[i]])
  hyb_forecast <- forecast::forecast(hyb_mod,5)

  #Plotting prediction and testing data (red for testing data)
  plot(hyb_forecast)
  lines(testTotals[[i]],col='red')

  #Getting MSE (the head and tail are used to get from 2017-2020)
  prediction<-hyb_forecast$fitted%>%as.numeric()%>%tail(5)%>%head(4)
  test<-testTotals[[i]]%>%as.numeric()

  #Saving MSE in hybError vector
  hybErrors<-c(hybErrors,MSE(prediction,test))
}
```

```
## Warning in removeModels(y = y, models = expandedModels): The stlm model requires
## that the input data be a seasonal ts object. The stlm model will not be used.

## Fitting the auto.arima model
## Fitting the ets model

## Fitting the thetam model

## Fitting the nnetar model

## Fitting the tbats model

## Warning in removeModels(y = y, models = expandedModels): The stlm model requires
## that the input data be a seasonal ts object. The stlm model will not be used.

## Fitting the auto.arima model

## Fitting the ets model

## Fitting the thetam model

## Fitting the nnetar model

## Fitting the tbats model
```



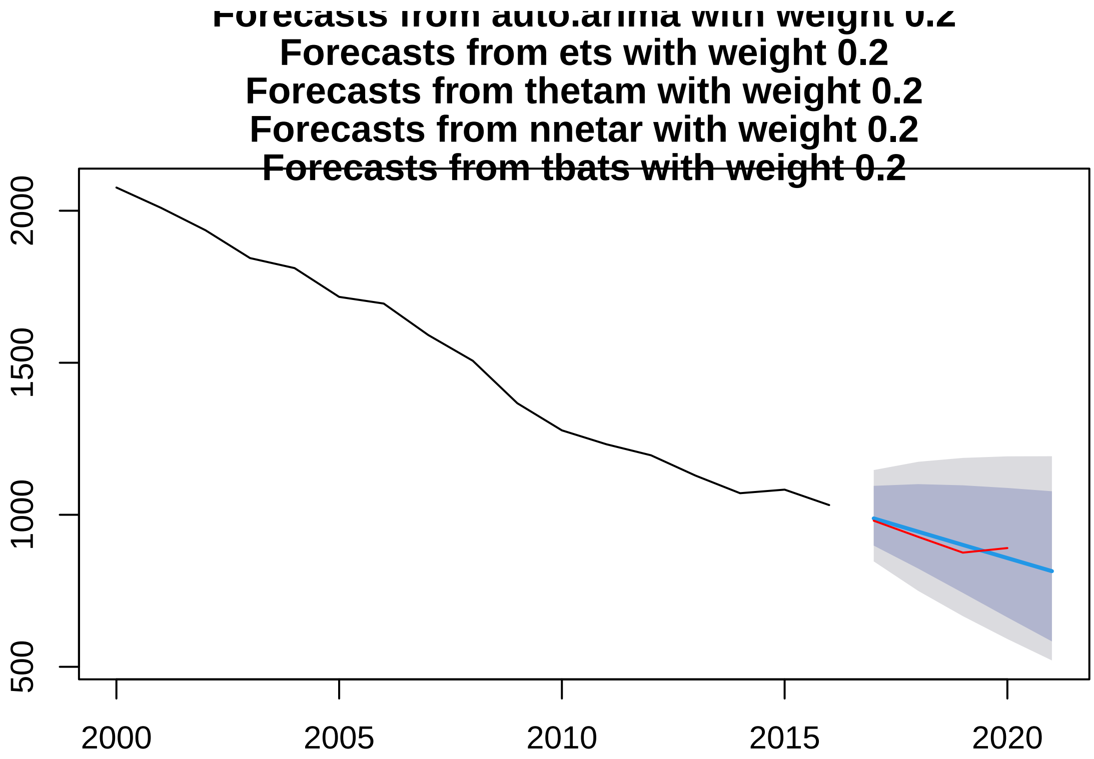
```
## Warning in removeModels(y = y, models = expandedModels): The stlm model requires
## that the input data be a seasonal ts object. The stlm model will not be used.
## Fitting the auto.arima model
```

```
## Fitting the ets model
```

```
## Fitting the thetam model
```

```
## Fitting the nnetar model
```

```
## Fitting the tbats model
```



```
## Warning in removeModels(y = y, models = expandedModels): The stlm model requires
## that the input data be a seasonal ts object. The stlm model will not be used.
```

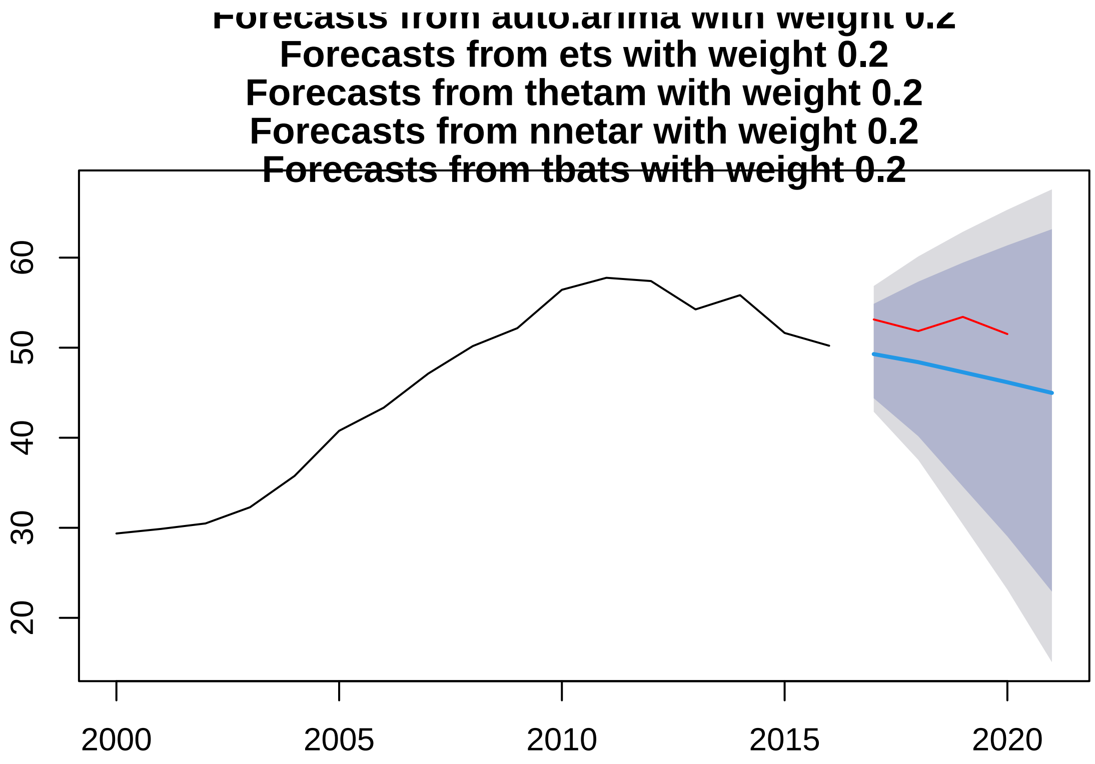
```
## Fitting the auto.arima model
```

```
## Fitting the ets model
```

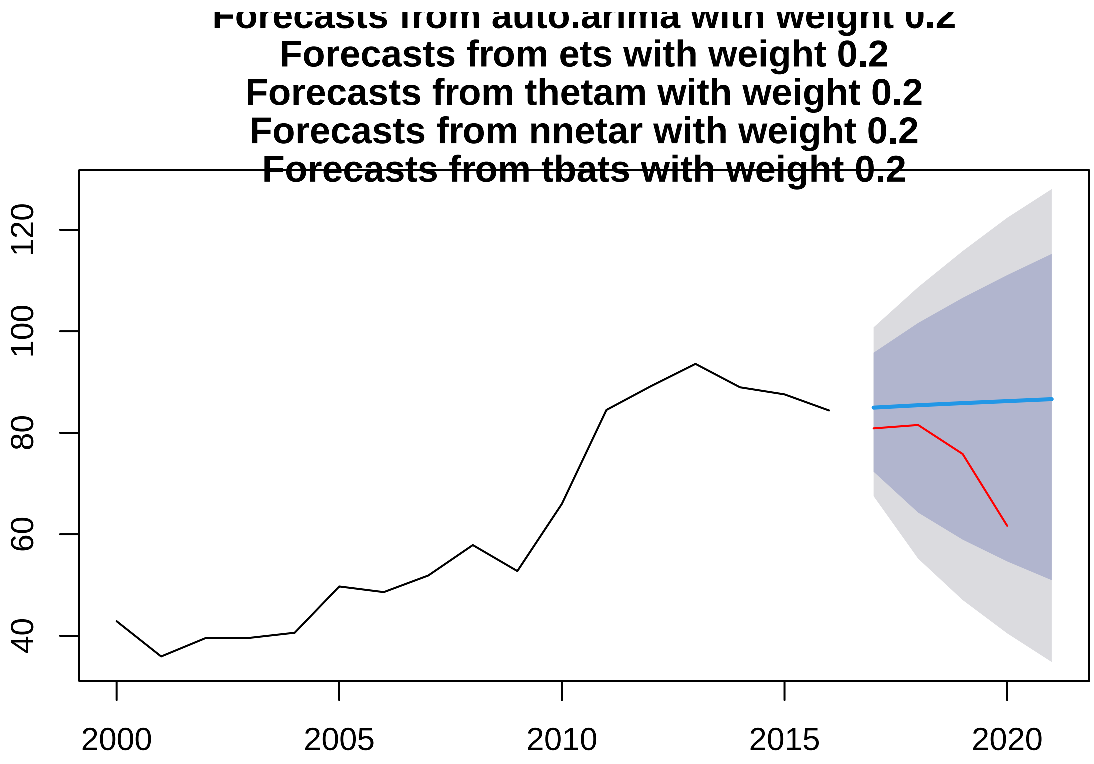
```
## Fitting the thetam model
```

```
## Fitting the nnetar model
```

```
## Fitting the tbats model
```



```
## Warning in removeModels(y = y, models = expandedModels): The stlm model requires
## that the input data be a seasonal ts object. The stlm model will not be used.

## Fitting the auto.arima model

## Fitting the ets model

## Fitting the thetam model

## Fitting the nnetar model

## Fitting the tbats model
```



```
## Warning in removeModels(y = y, models = expandedModels): The stlm model requires
## that the input data be a seasonal ts object. The stlm model will not be used.
```

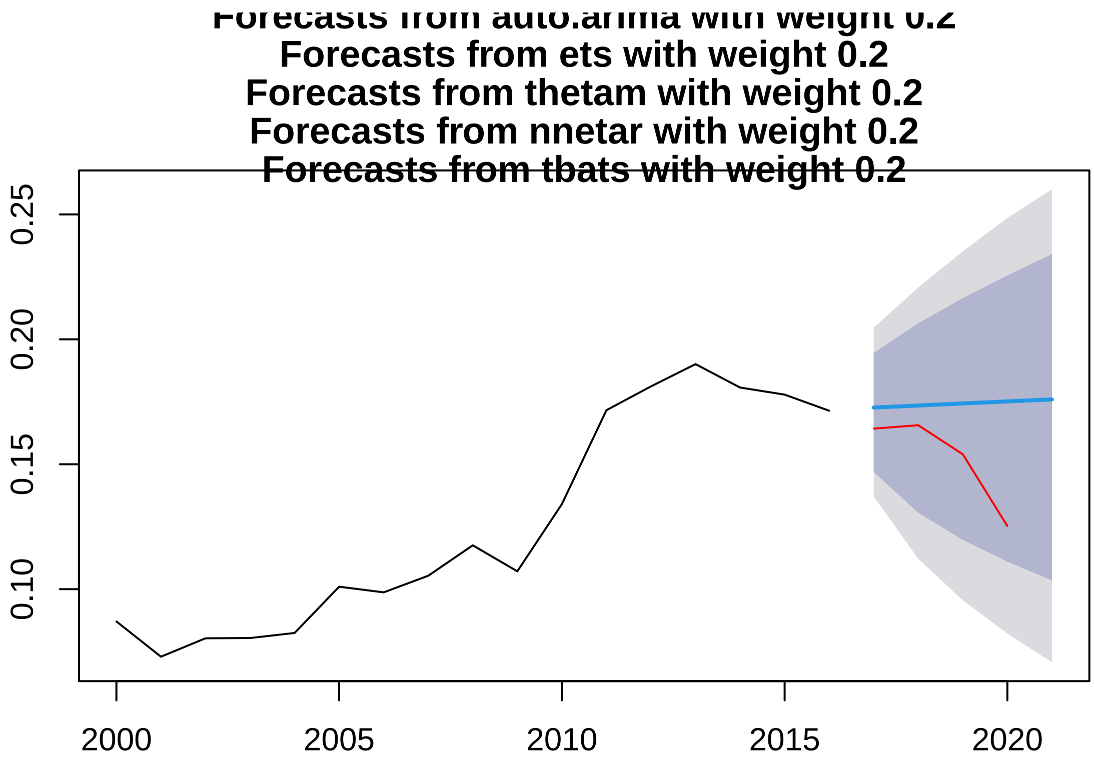
```
## Fitting the auto.arima model
```

```
## Fitting the ets model
```

```
## Fitting the thetam model
```

```
## Fitting the nnetar model
```

```
## Fitting the tbats model
```



```
## Warning in removeModels(y = y, models = expandedModels): The stlm model requires
## that the input data be a seasonal ts object. The stlm model will not be used.
```

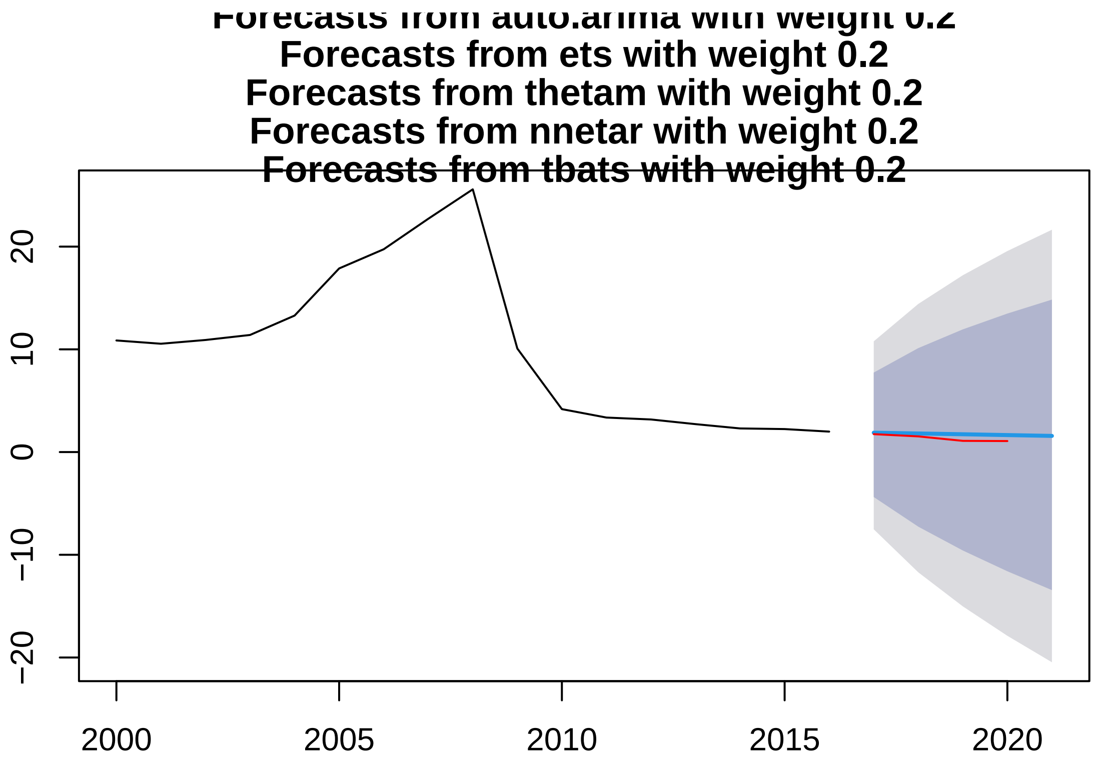
```
## Fitting the auto.arima model
```

```
## Fitting the ets model
```

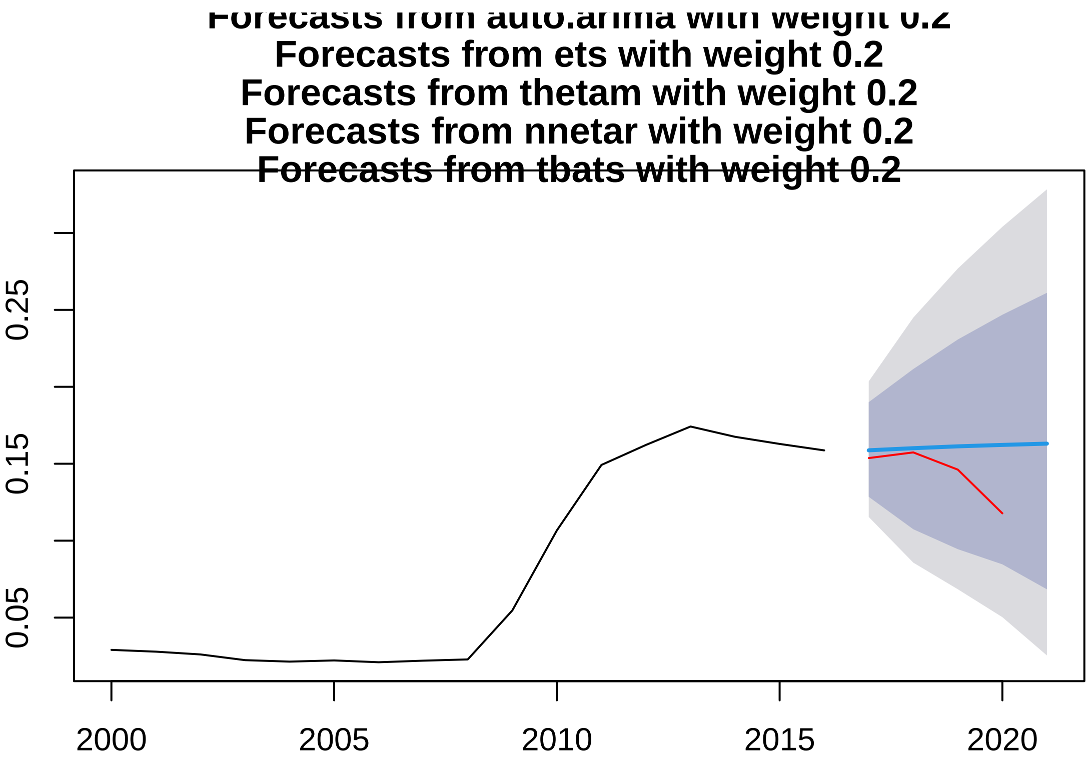
```
## Fitting the thetam model
```

```
## Fitting the nnetar model
```

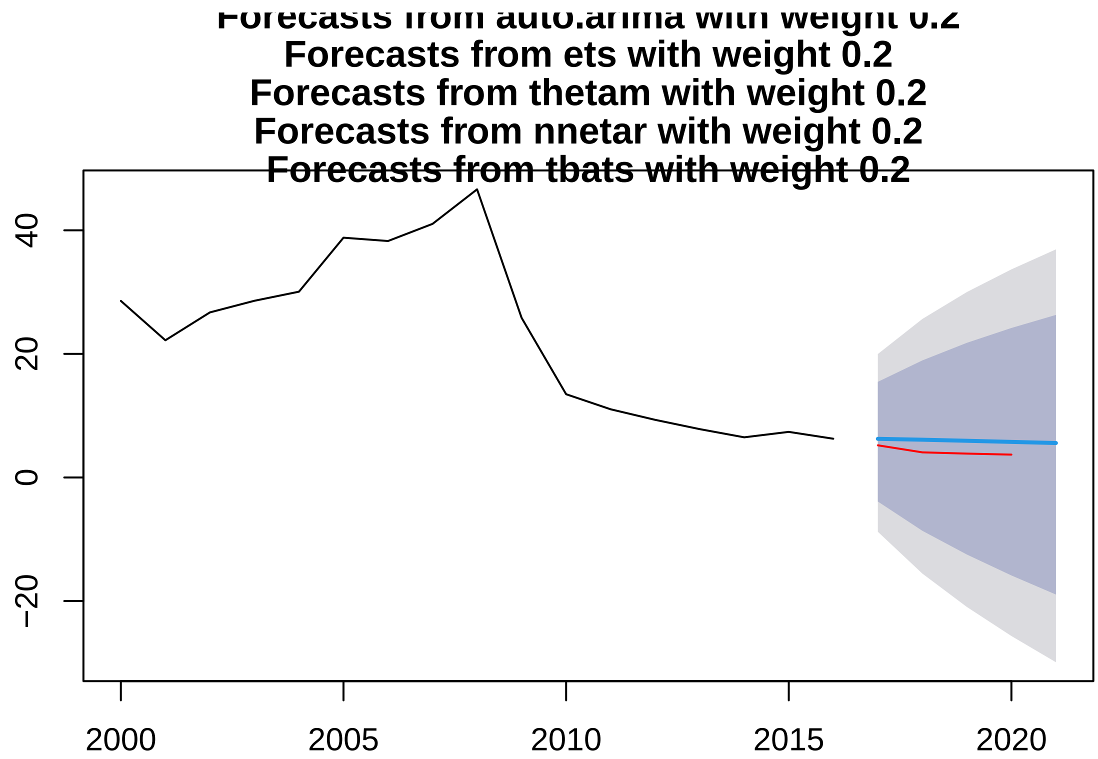
```
## Fitting the tbats model
```



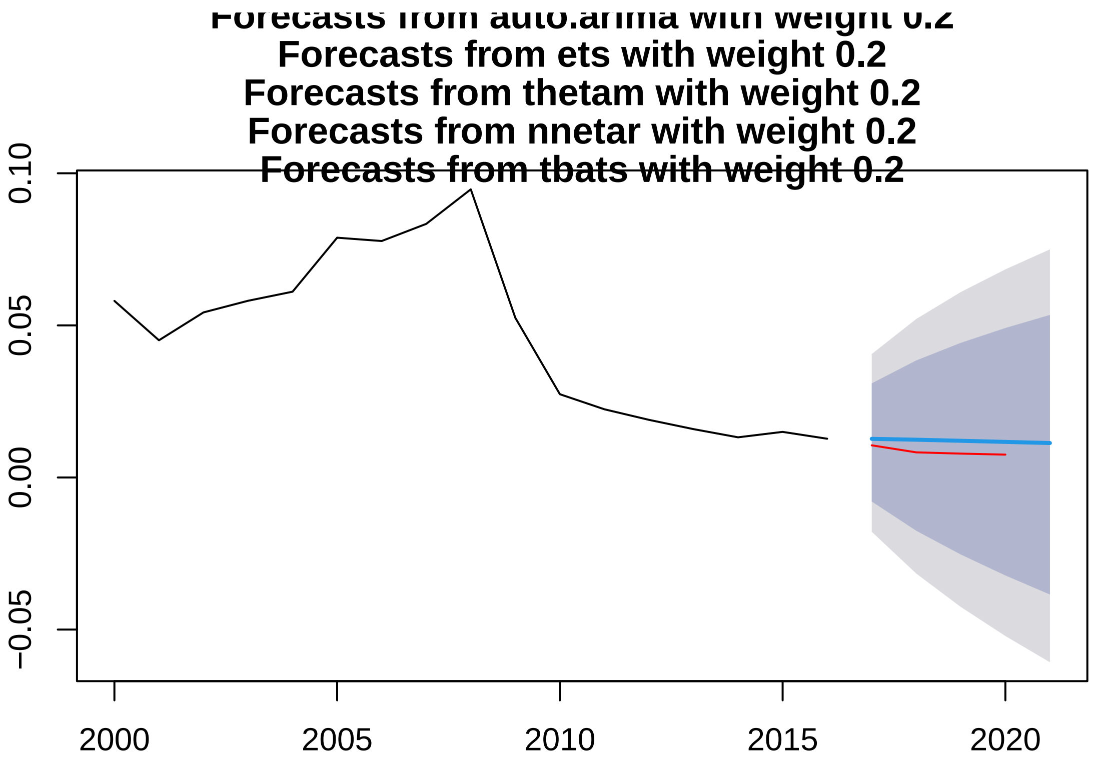
```
## Warning in removeModels(y = y, models = expandedModels): The stlm model requires
## that the input data be a seasonal ts object. The stlm model will not be used.

## Fitting the auto.arima model

## Fitting the ets model

## Fitting the thetam model

## Fitting the nnetar model

## Fitting the tbats model
```



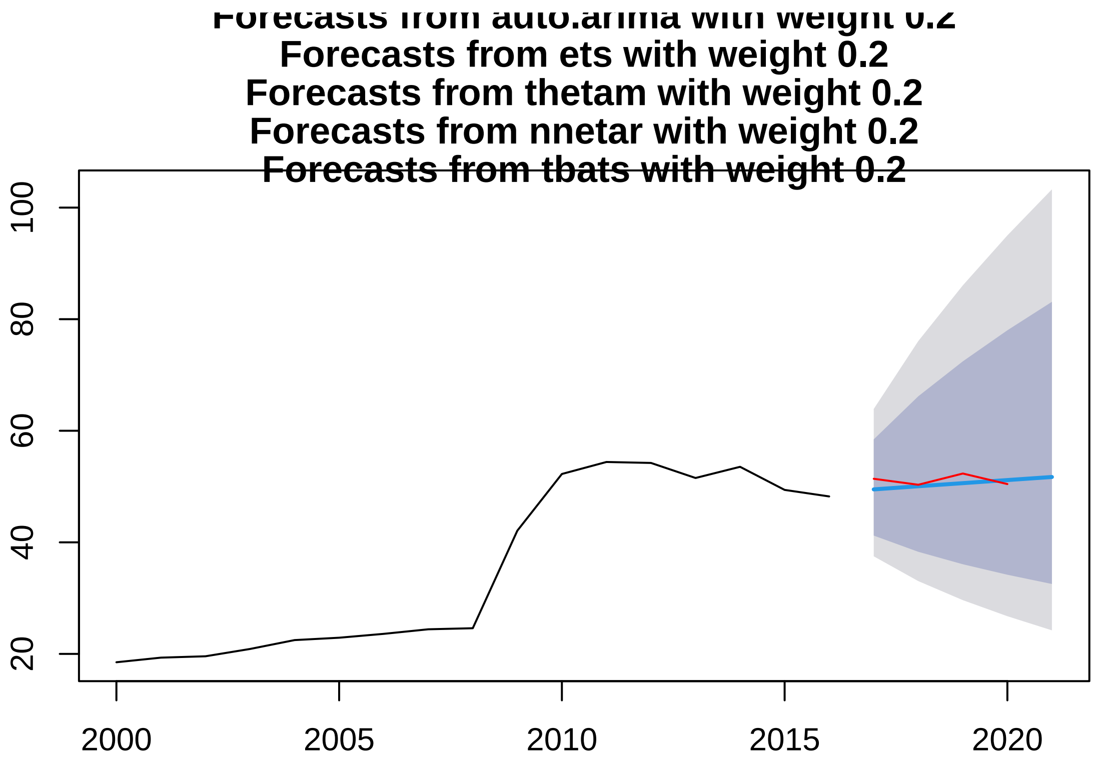
```
## Warning in removeModels(y = y, models = expandedModels): The stlm model requires
## that the input data be a seasonal ts object. The stlm model will not be used.

## Fitting the auto.arima model

## Fitting the ets model

## Fitting the thetam model

## Fitting the nnetar model

## Fitting the tbats model
```



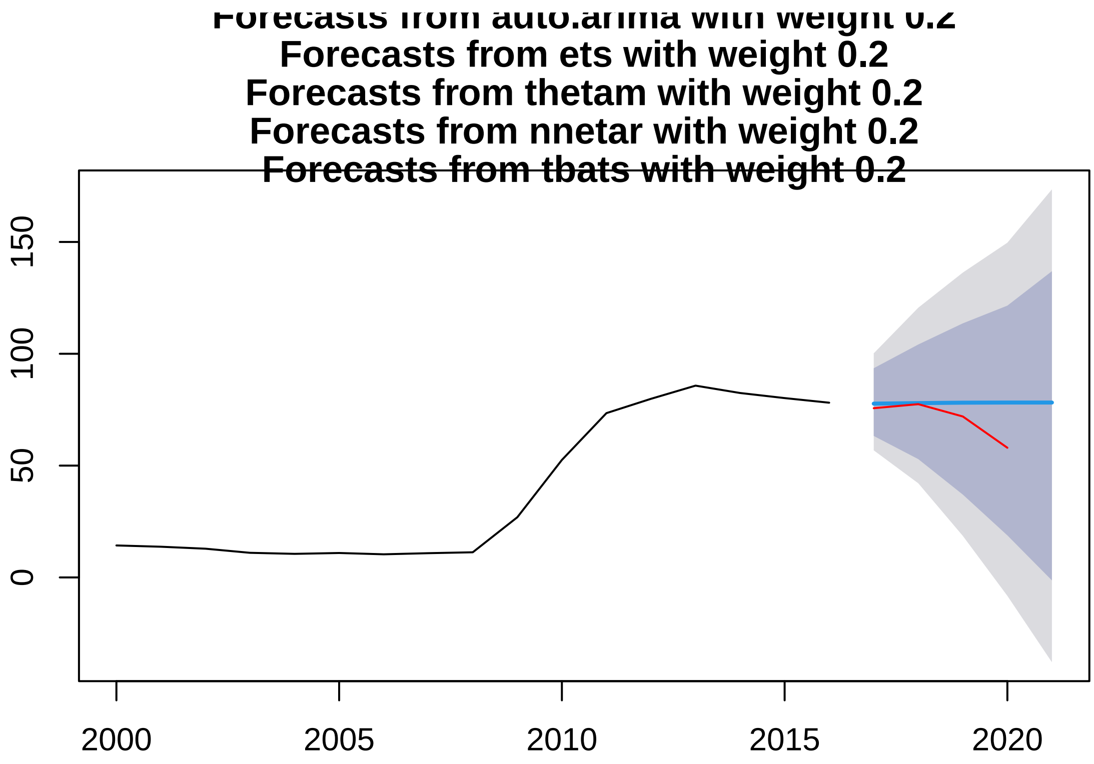
```
## Warning in removeModels(y = y, models = expandedModels): The stlm model requires
## that the input data be a seasonal ts object. The stlm model will not be used.

## Fitting the auto.arima model

## Fitting the ets model

## Fitting the thetam model

## Fitting the nnetar model

## Fitting the tbats model
```



```
## Warning in removeModels(y = y, models = expandedModels): The stlm model requires
## that the input data be a seasonal ts object. The stlm model will not be used.

## Fitting the auto.arima model

## Fitting the ets model

## Fitting the thetam model

## Fitting the nnetar model

## Fitting the tbats model
```



```
## Warning in removeModels(y = y, models = expandedModels): The stlm model requires
## that the input data be a seasonal ts object. The stlm model will not be used.

## Fitting the auto.arima model

## Fitting the ets model

## Fitting the thetam model

## Fitting the nnetar model

## Fitting the tbats model
```



```
## Warning in removeModels(y = y, models = expandedModels): The stlm model requires
## that the input data be a seasonal ts object. The stlm model will not be used.
```

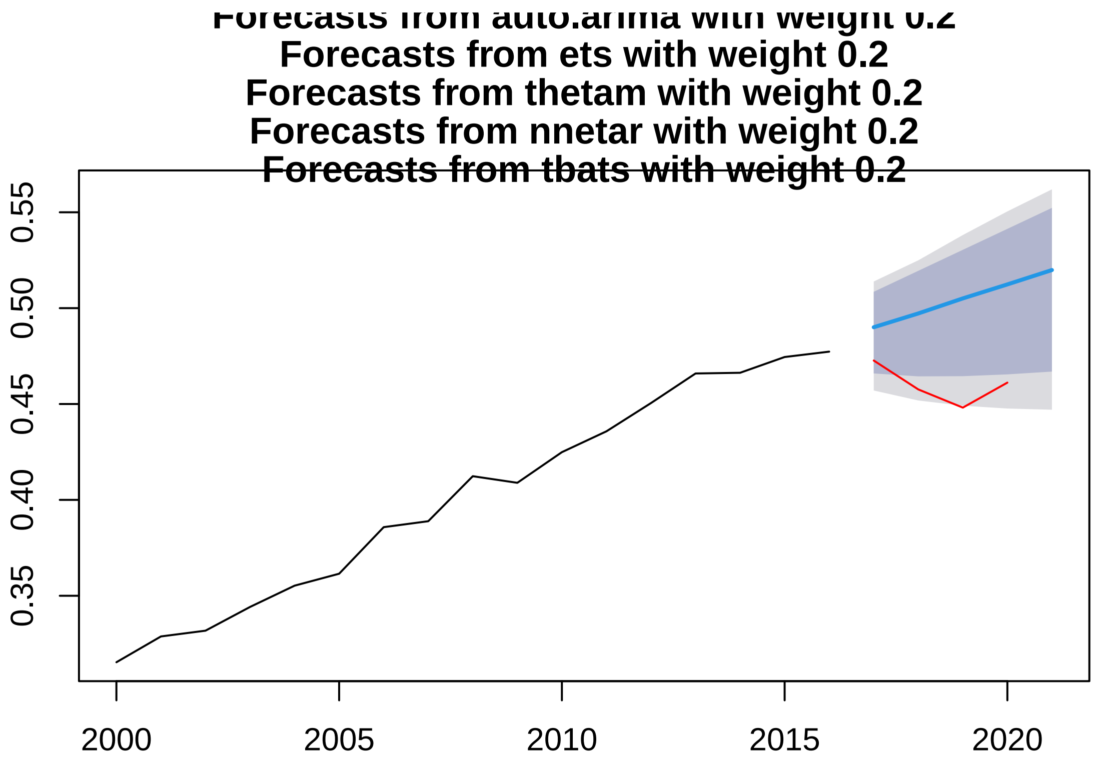
```
## Fitting the auto.arima model
```

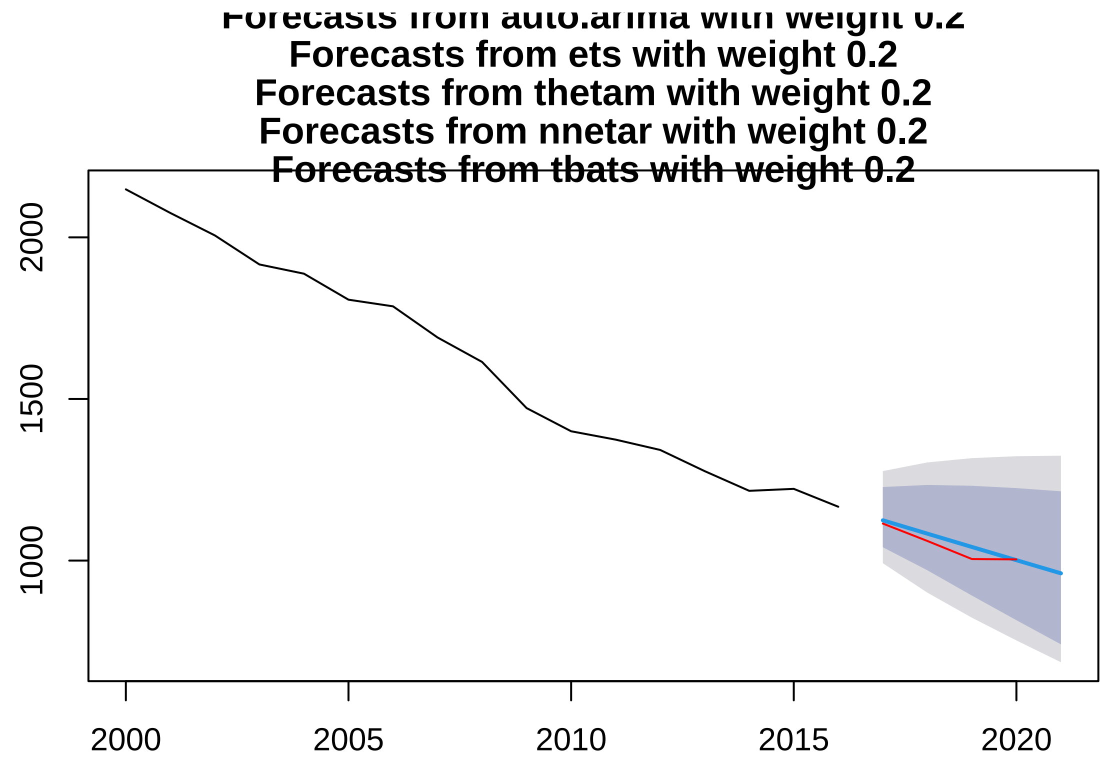
```
## Fitting the ets model
```

```
## Fitting the thetam model
```

```
## Fitting the nnetar model
```

```
## Fitting the tbats model
```

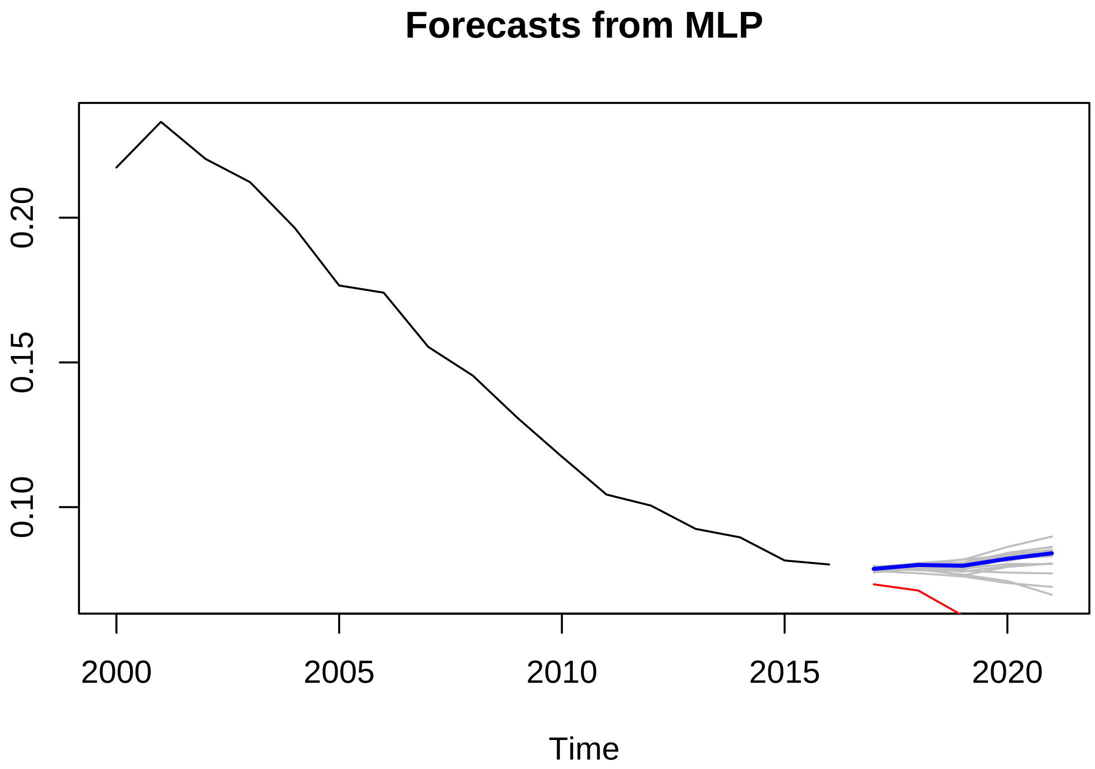

Forecasts from auto.arima with weight 0.2
Forecasts from ets with weight 0.2
Forecasts from thetam with weight 0.2
Forecasts from nnetar with weight 0.2
Forecasts from tbats with weight 0.2
0.35
0.40
0.45
0.50
0.55
2000
2005
2010
2015
2020

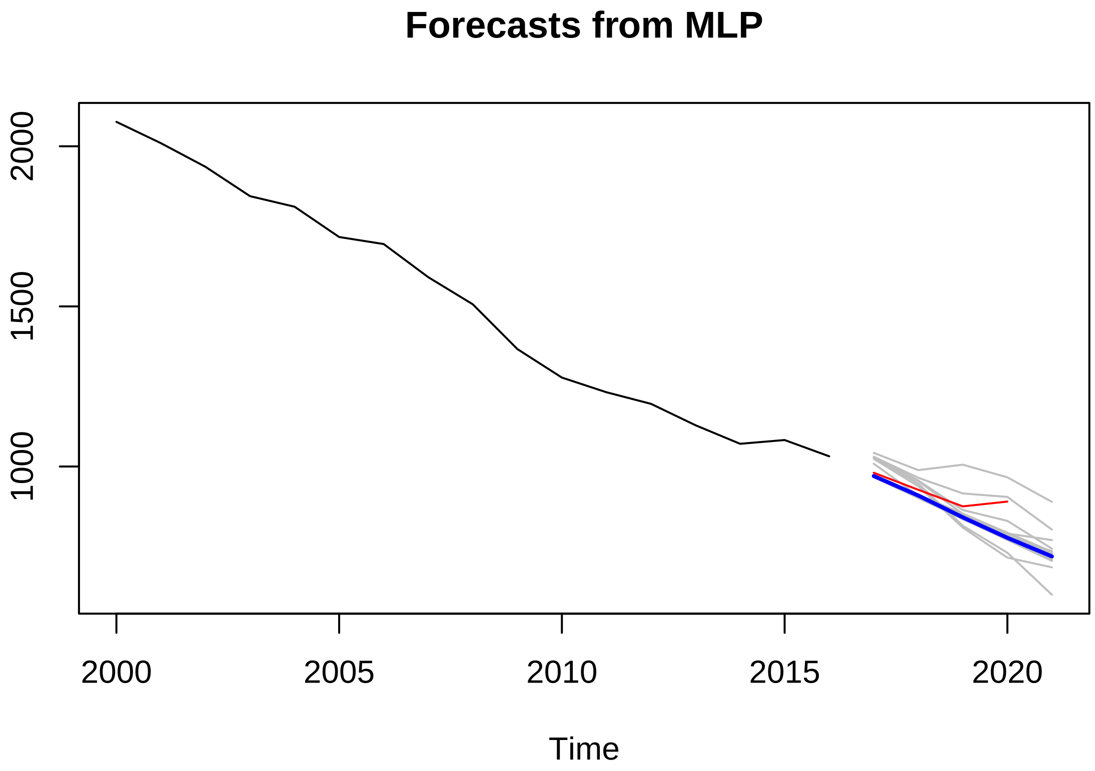


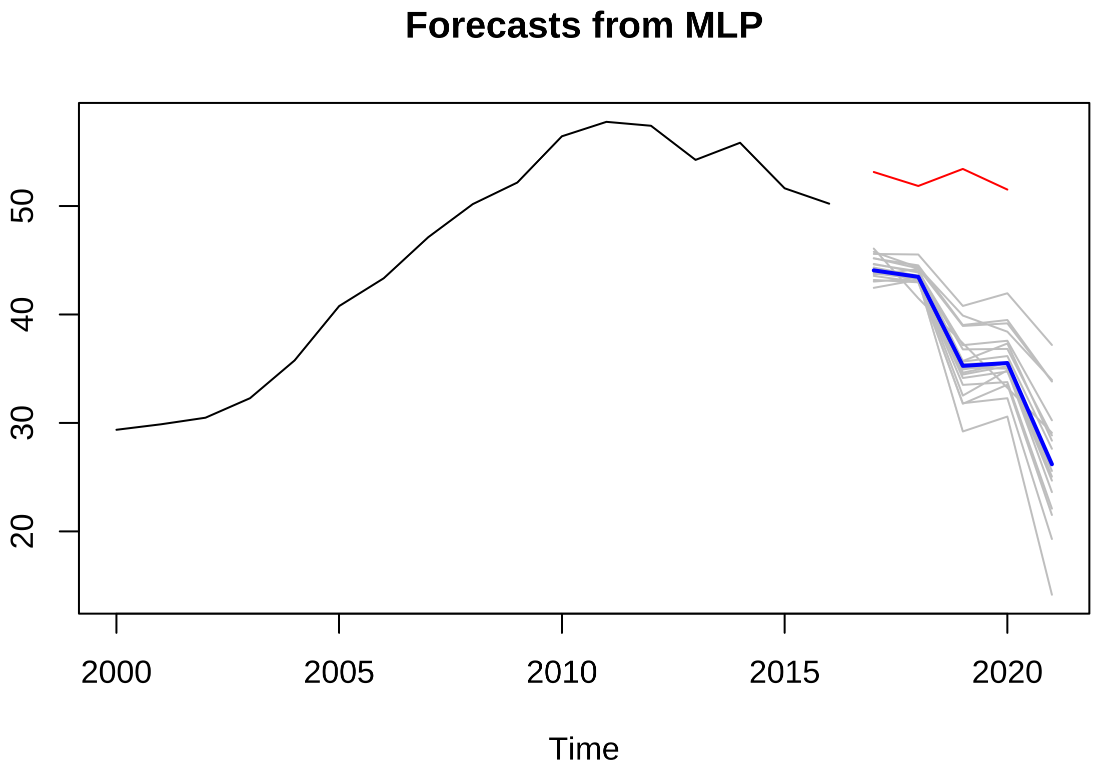
Multilayer Perceptron Model

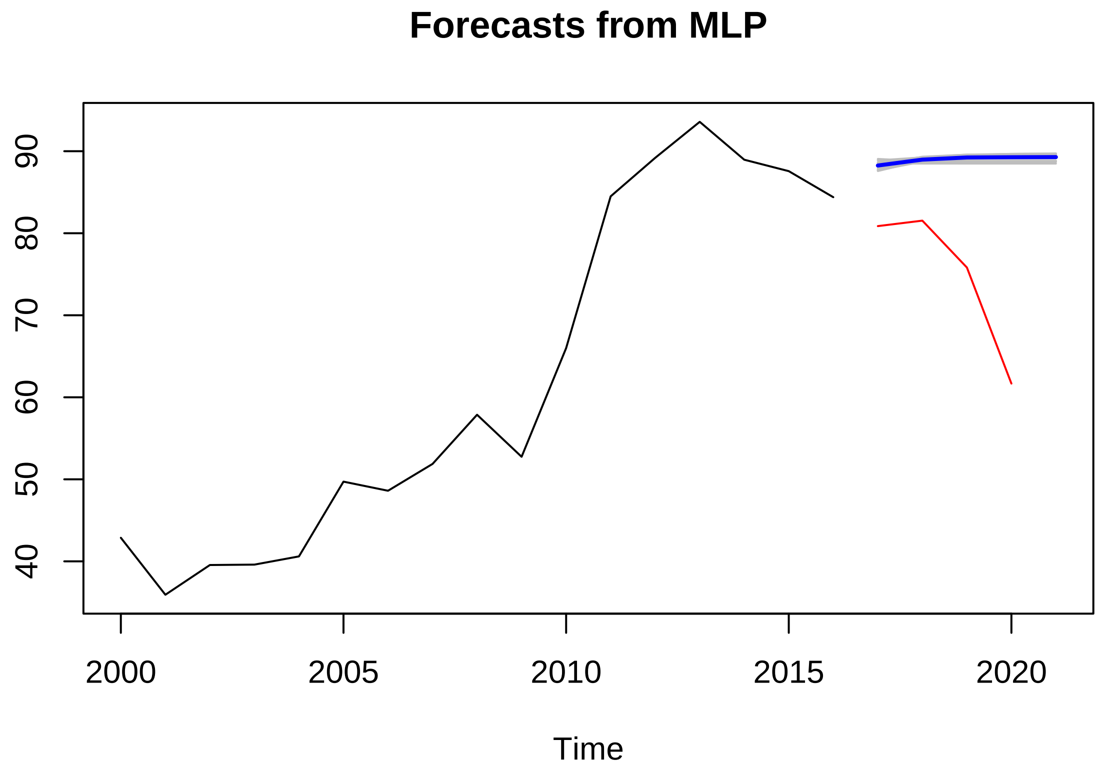
```
mlpErrors<-c()
for(i in c(1:13)){
  #Training
  mlp_fit<-mlp(trainTotals[[i]])
  mlp_model<-forecast::forecast(mlp_fit,5)

  #Plotting
  plot(mlp_model)
  lines(testTotals[[i]],col='red')

  #Getting MSE (the head and tail are used to get from 2017-2020)
  prediction<-mlp_model$fitted%>%as.numeric()%>%tail(5)%>%head(4)
  test<-testTotals[[i]]%>%as.numeric()

  ##Saving MSE in mlpError vector
  mlpErrors<-c(mlpErrors,MSE(prediction,test))
}
```

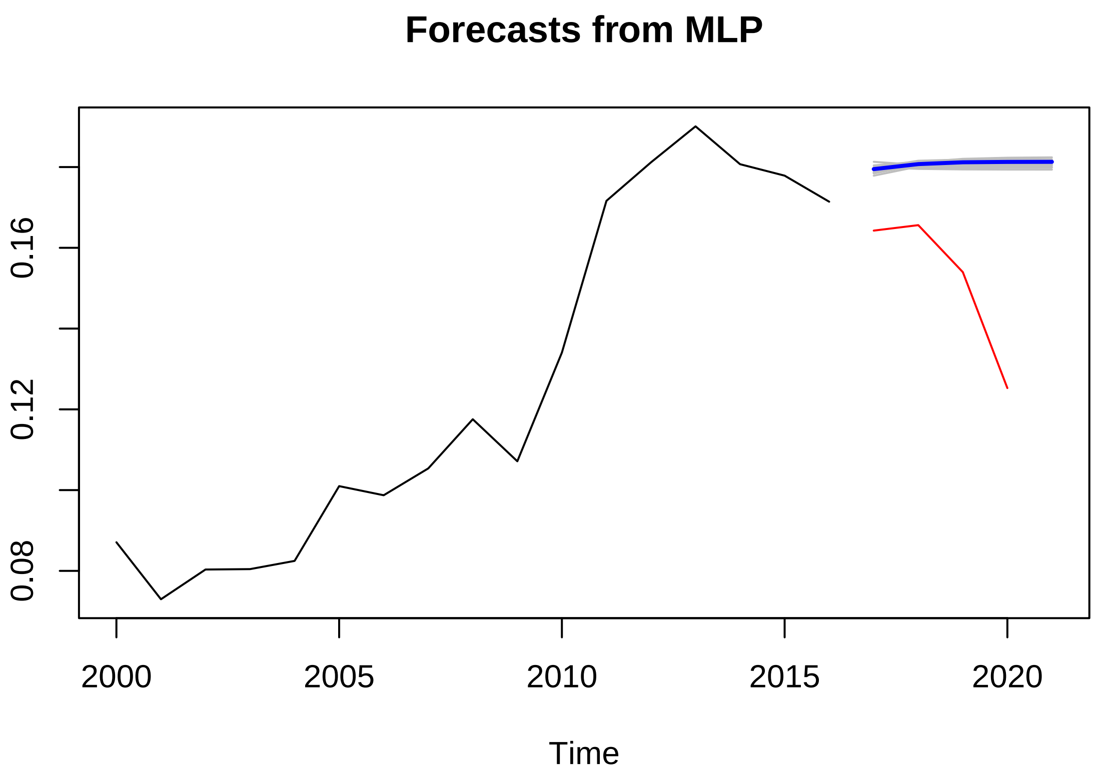
```
## Warning in preprocess(y, m, lags, keep, difforder, sel.lag, allow.det.season, :
## No inputs left in the network after pre-selection, forcing AR(1).
```

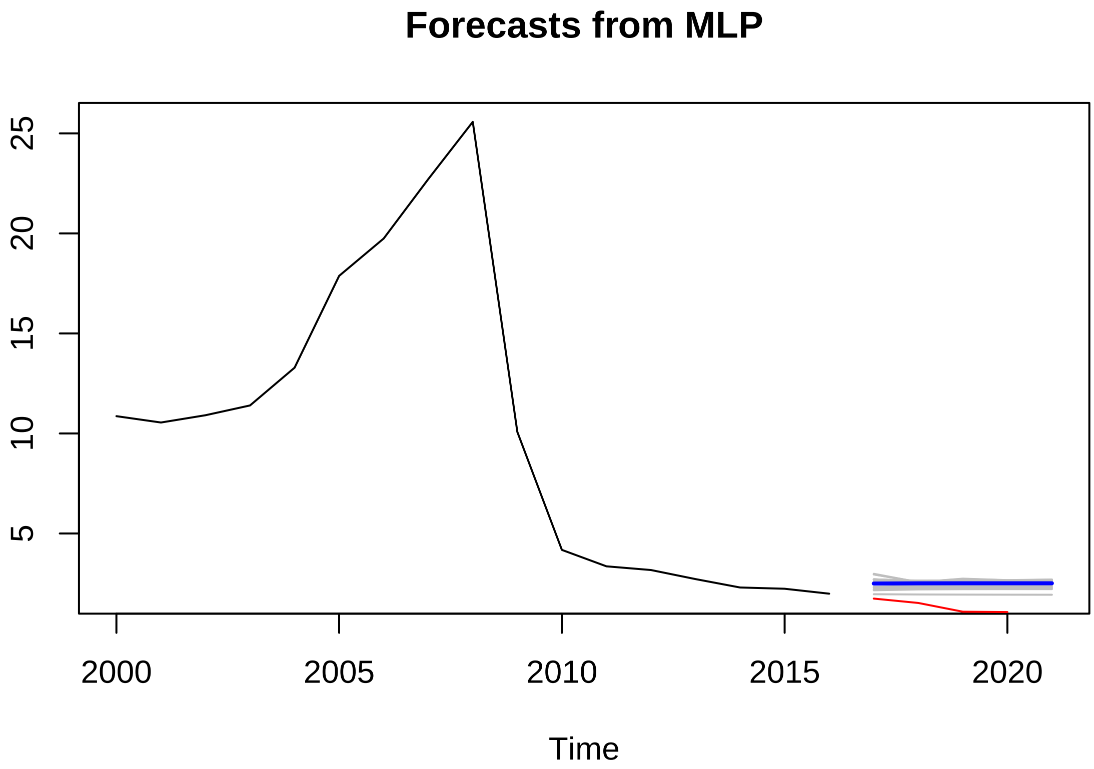

Forecasts from MLP
0.10
0.15
0.20
2000
2005
2010
2015
2020
Time

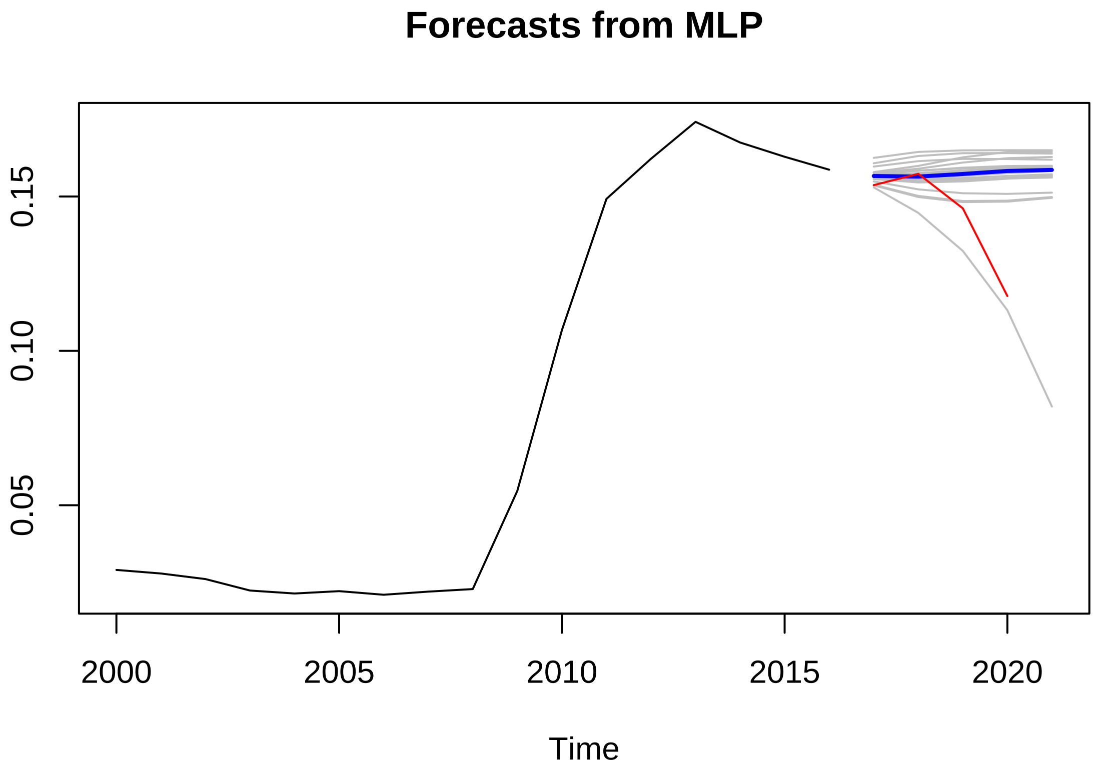

Forecasts from MLP
2000
1500
1000
2000
2005
2010
2015
2020
Time

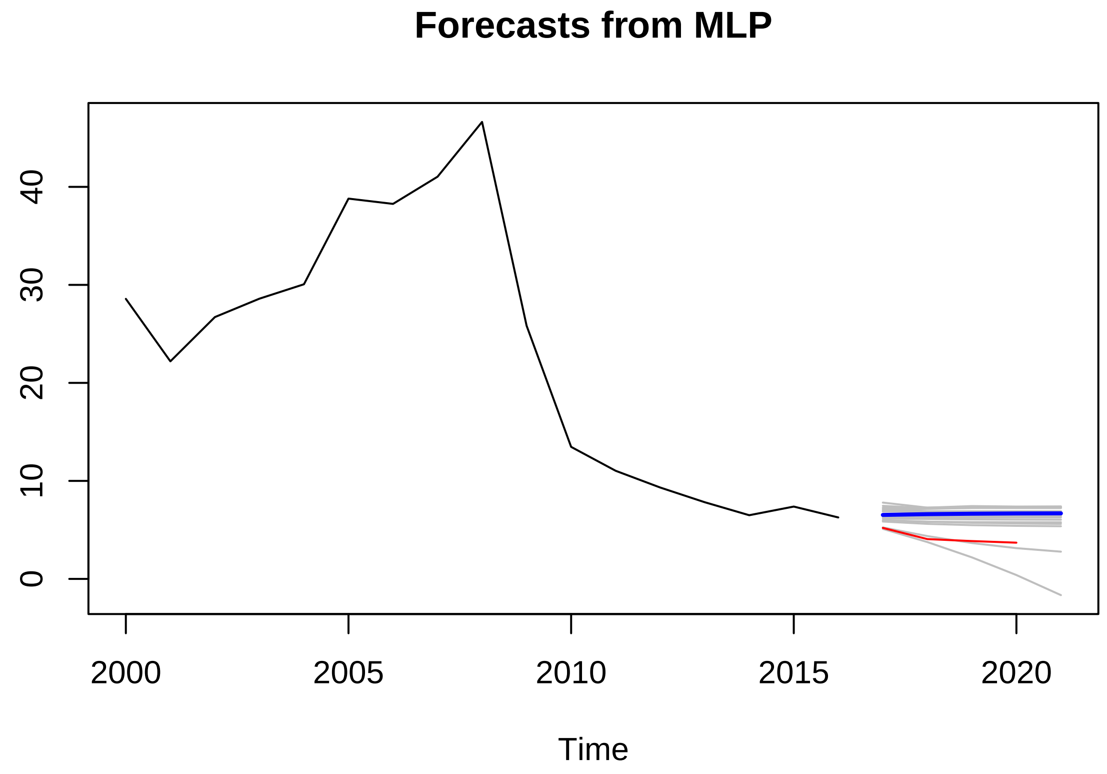

Forecasts from MLP
50
40
30
20
2000
2005
2010
2015
2020
Time

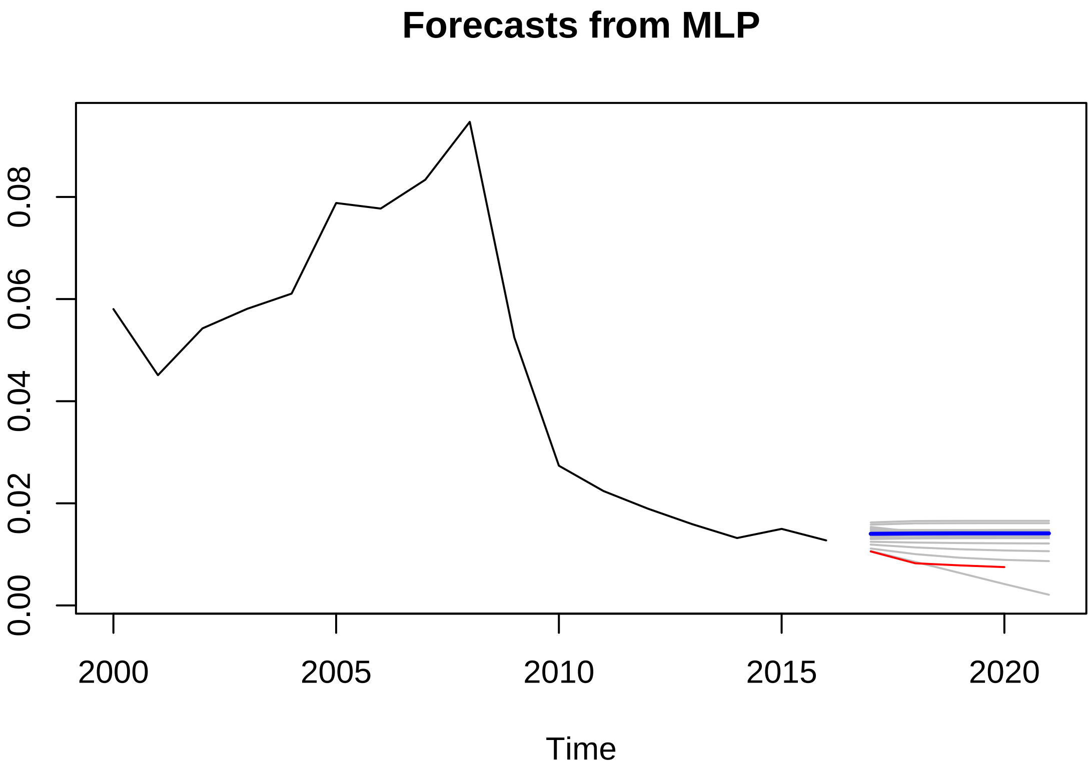

Forecasts from MLP
90
80
70
60
50
40
2000
2005
2010
2015
2020
Time

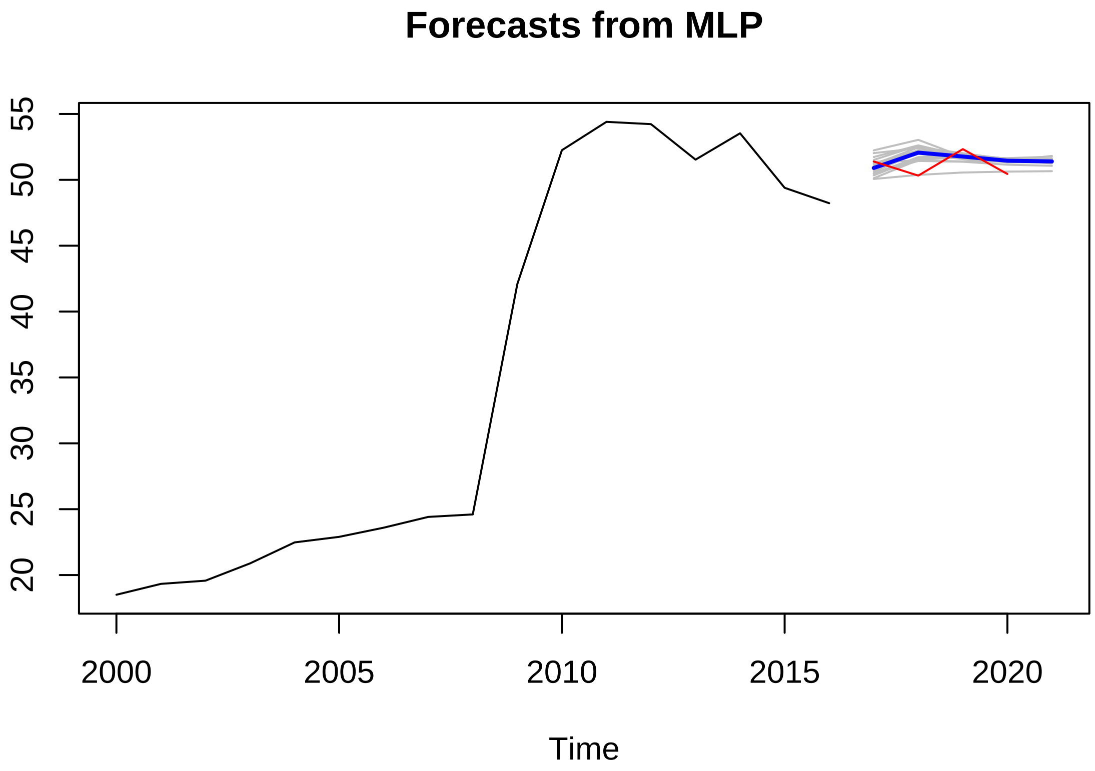

Forecasts from MLP
0.08
0.12
0.16
2000
2005
2010
2015
2020
Time

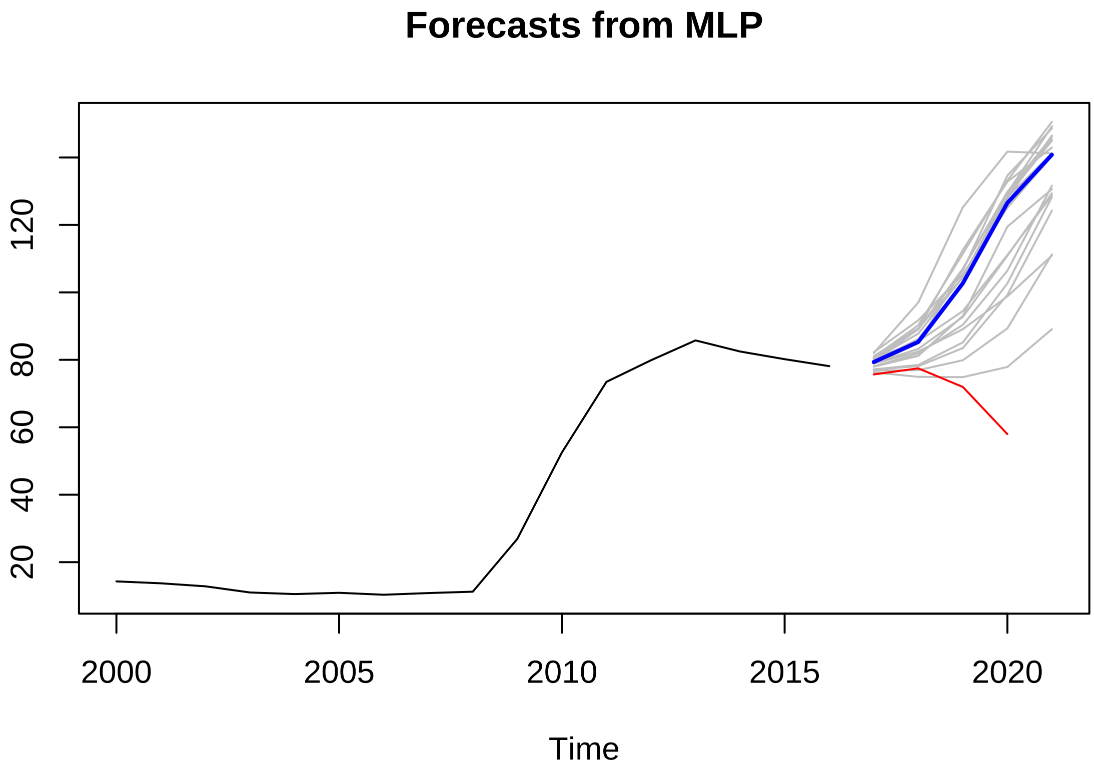

Forecasts from MLP
25
20
15
10
5
2000
2005
2010
2015
2020
Time


Forecasts from MLP
0.05
0.10
0.15
2000
2005
2010
2015
2020
Time

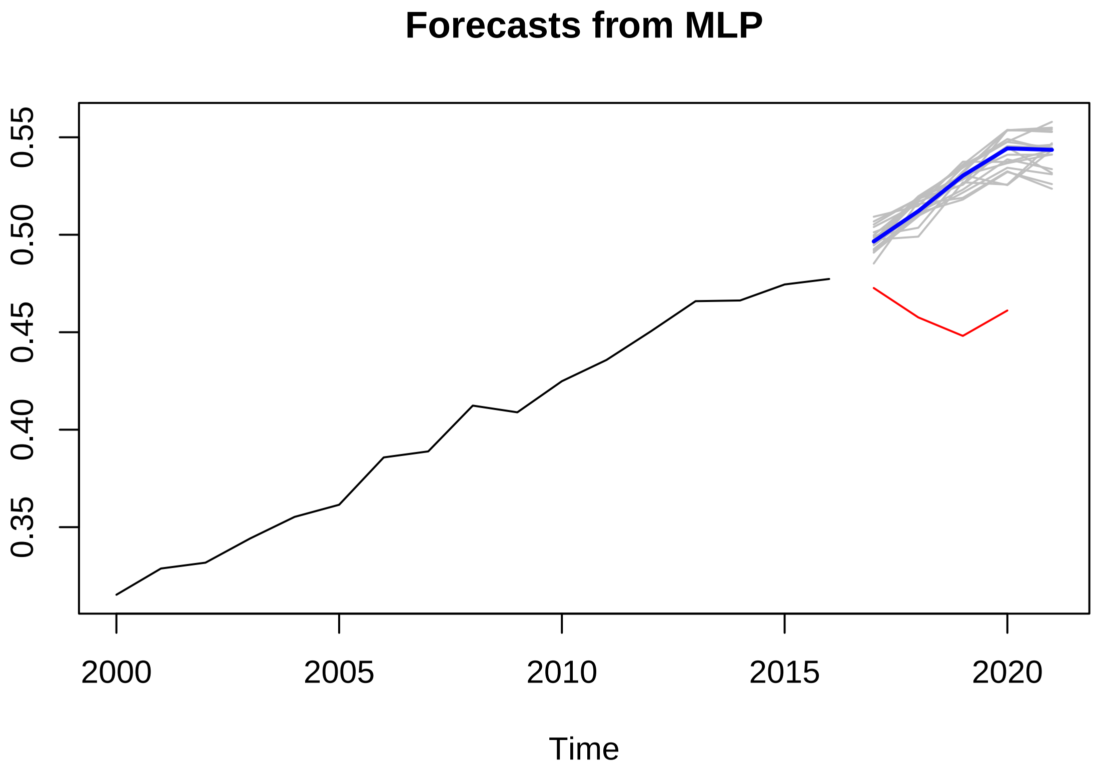

Forecasts from MLP
0
10
20
30
40
2000
2005
2010
2015
2020
Time

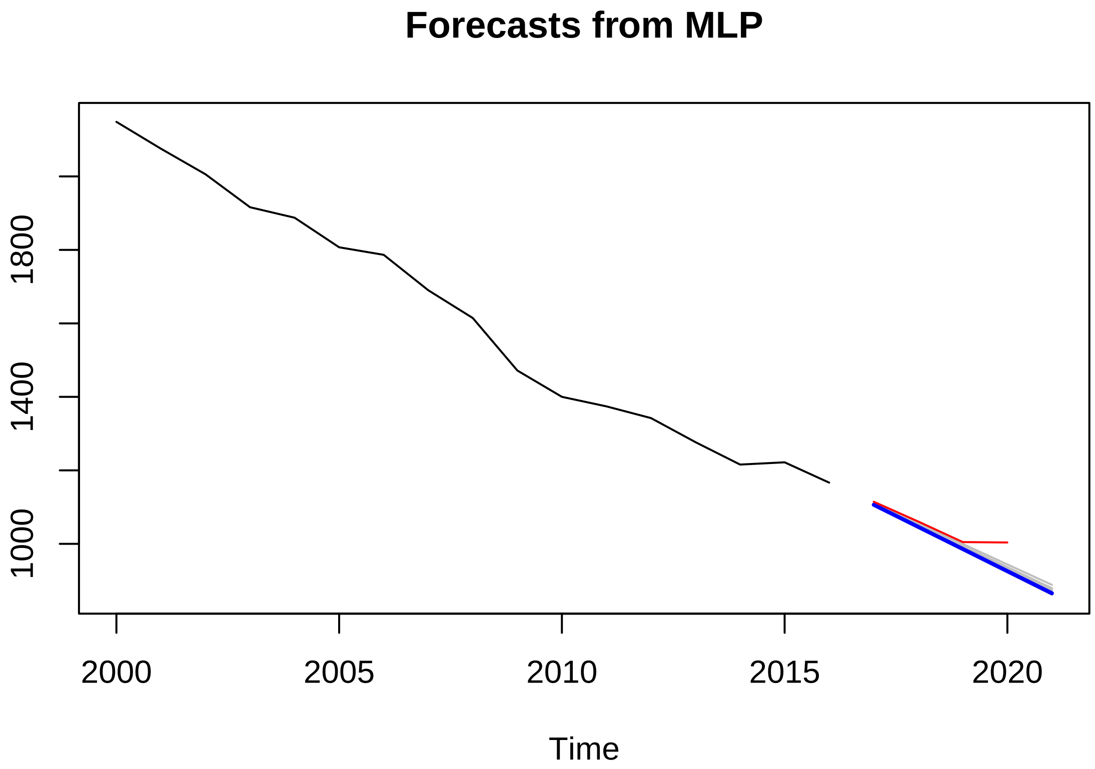

Forecasts from MLP
0.00
0.02
0.04
0.06
0.08
2000
2005
2010
2015
2020
Time


Forecasts from MLP
55
50
45
40
35
30
25
20
2000
2005
2010
2015
2020
Time

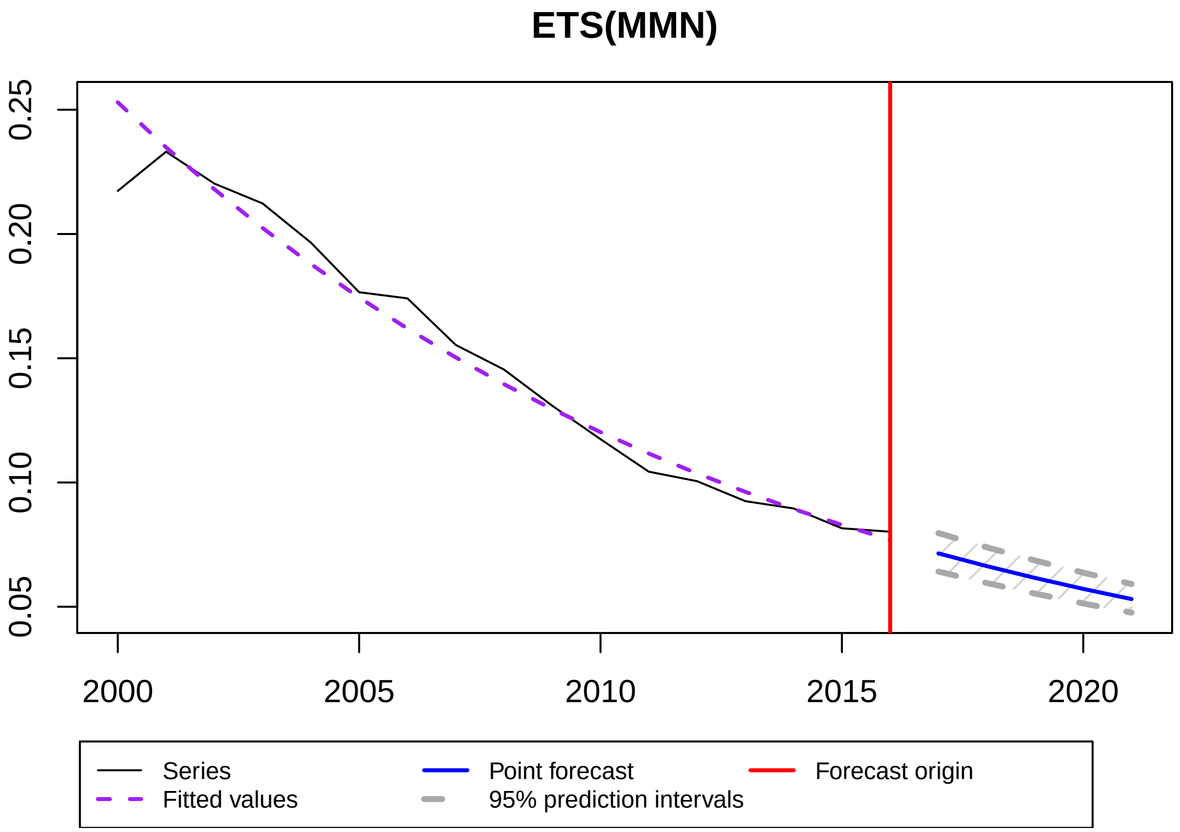


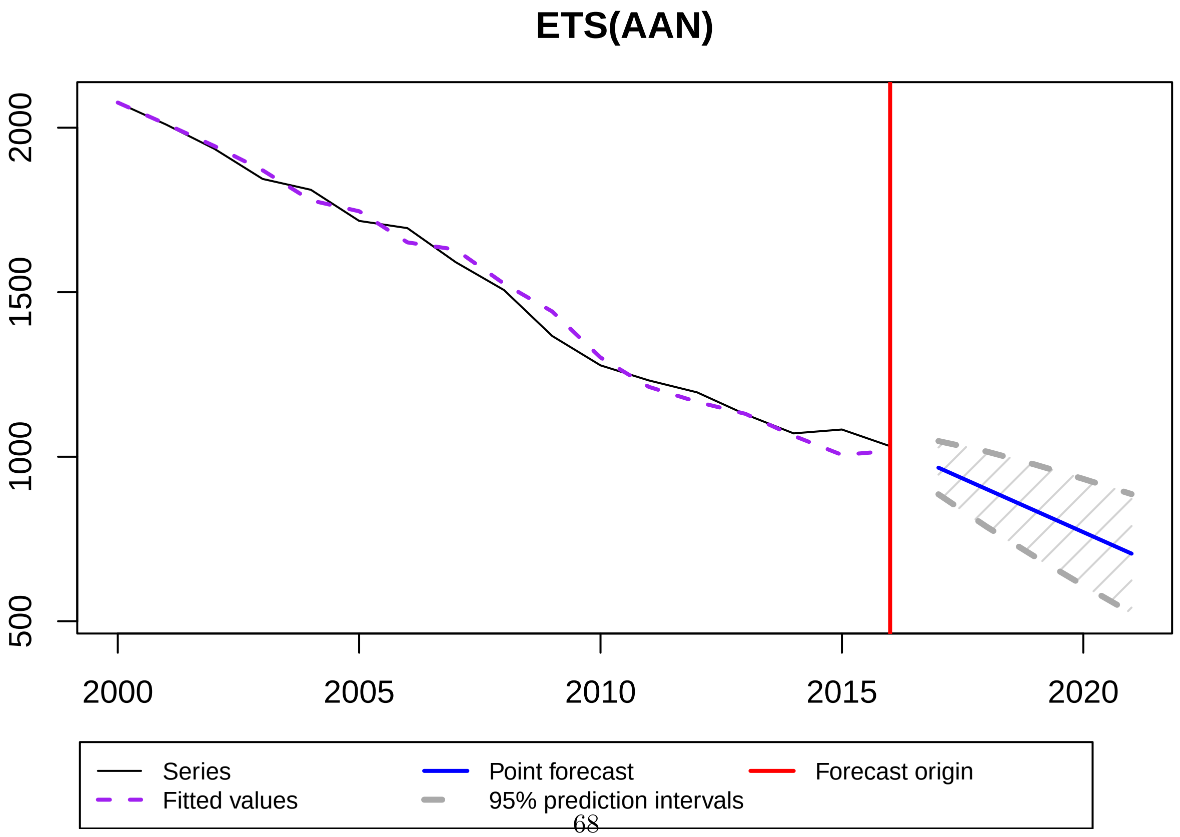
```
## Warning in preprocess(y, m, lags, keep, difforder, sel.lag, allow.det.season, :
## No inputs left in the network after pre-selection, forcing AR(1).
```

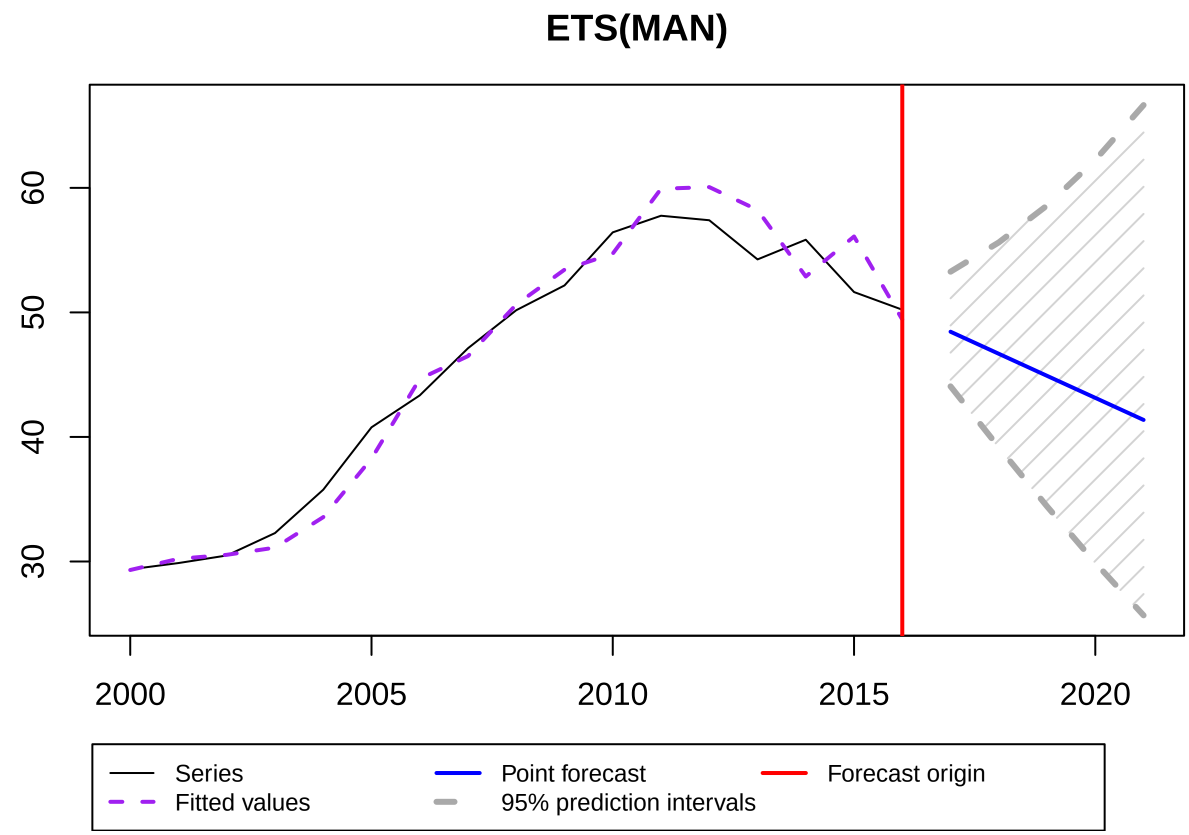

Forecasts from MLP
0.35
0.40
0.45
0.50
0.55
2000
2005
2010
2015
2020
Time

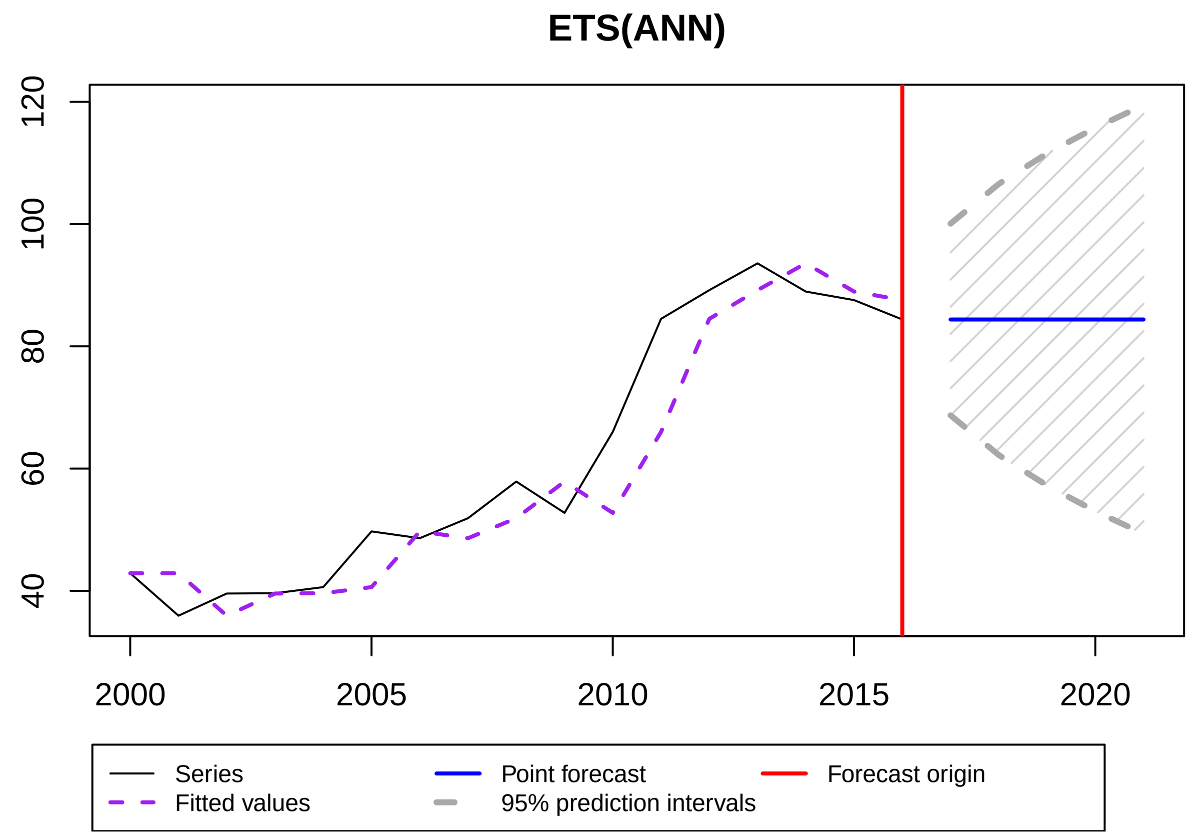

Forecasts from MLP
1000
1400
1800
2000
2005
2010
2015
2020
Time

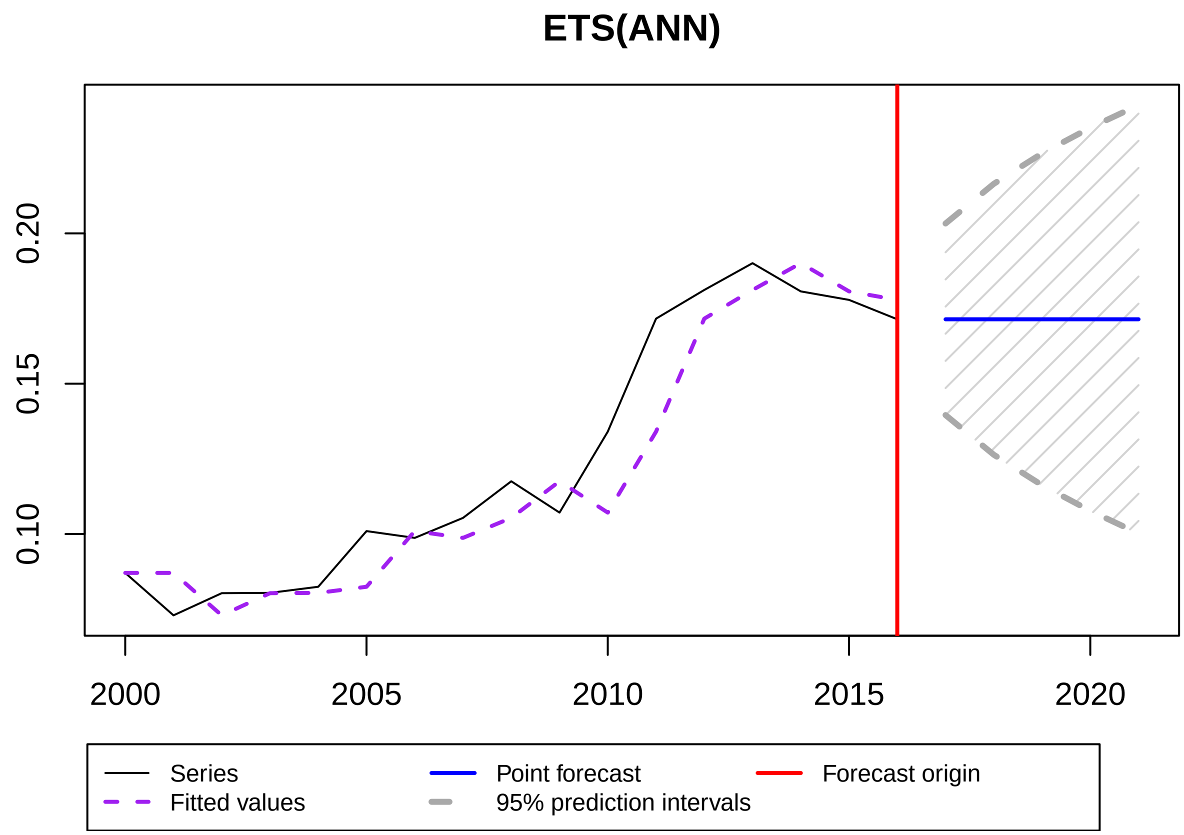
Exponential Smoothing

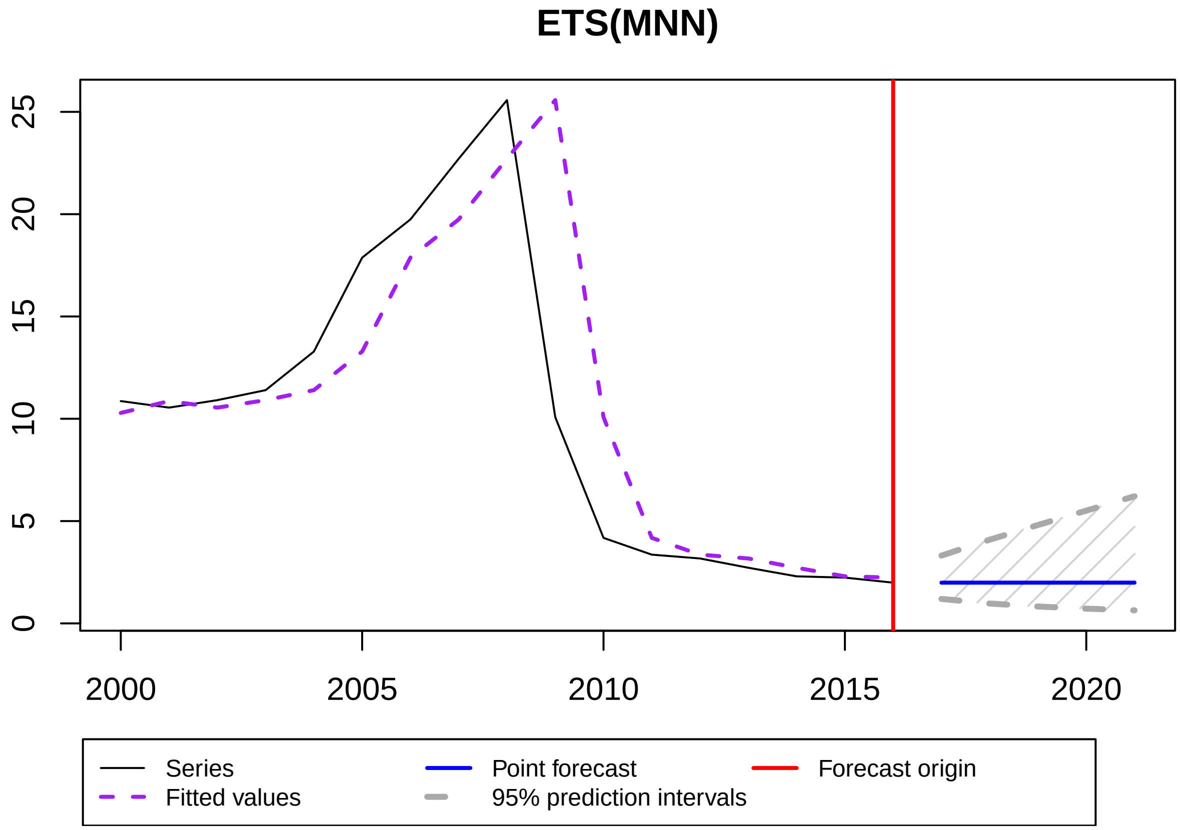


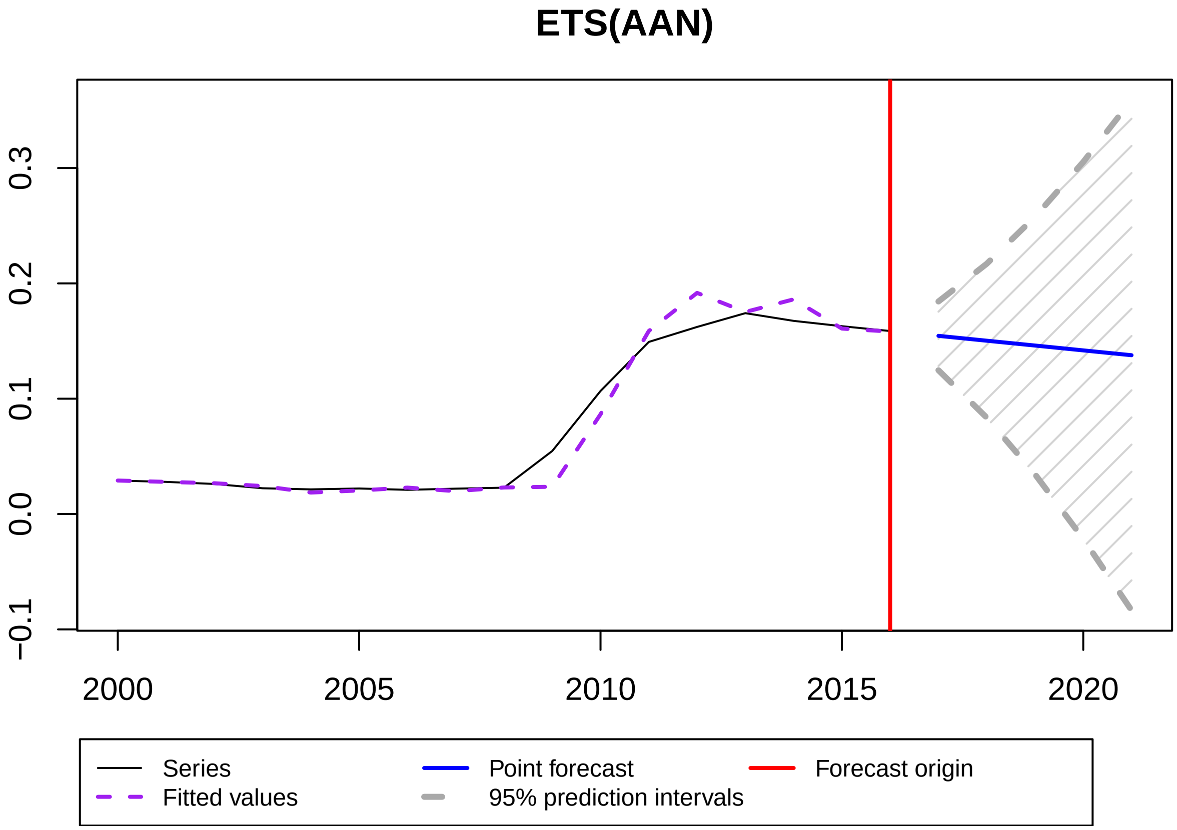


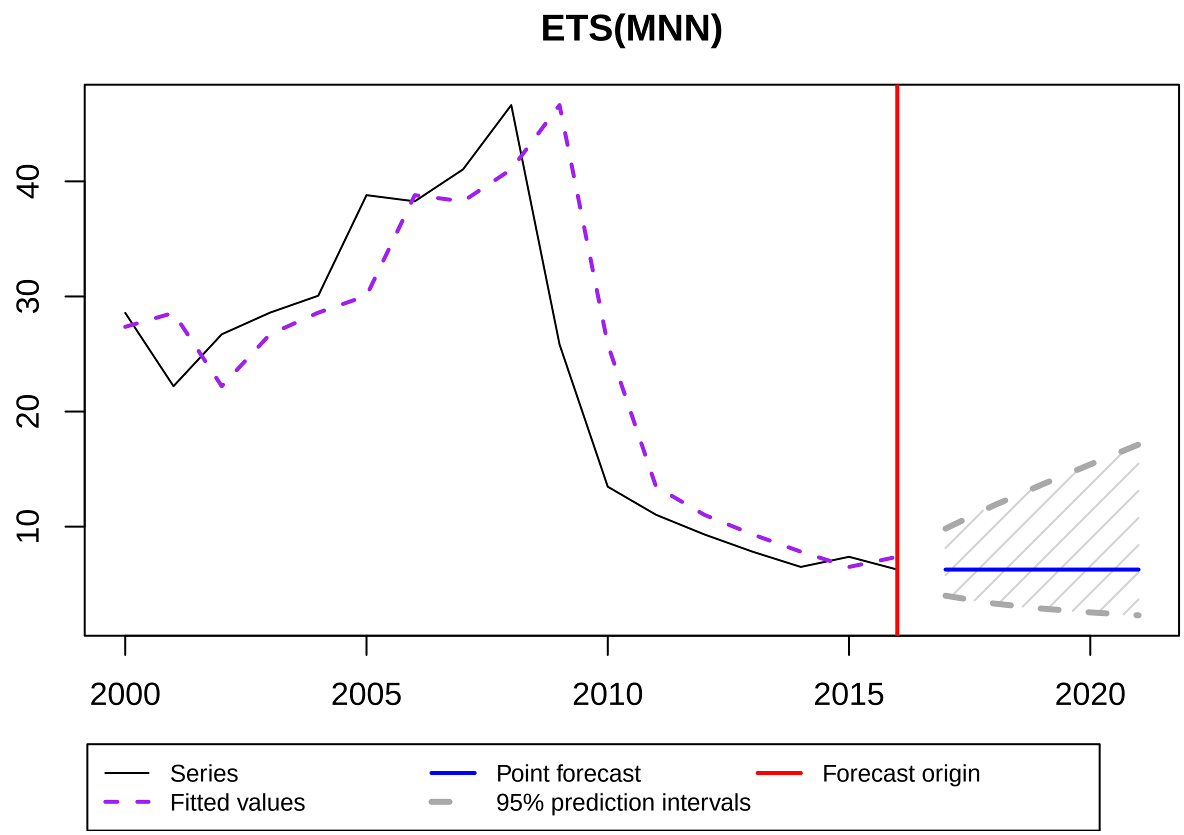

ETS(MAN)
30
40
50
60
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin

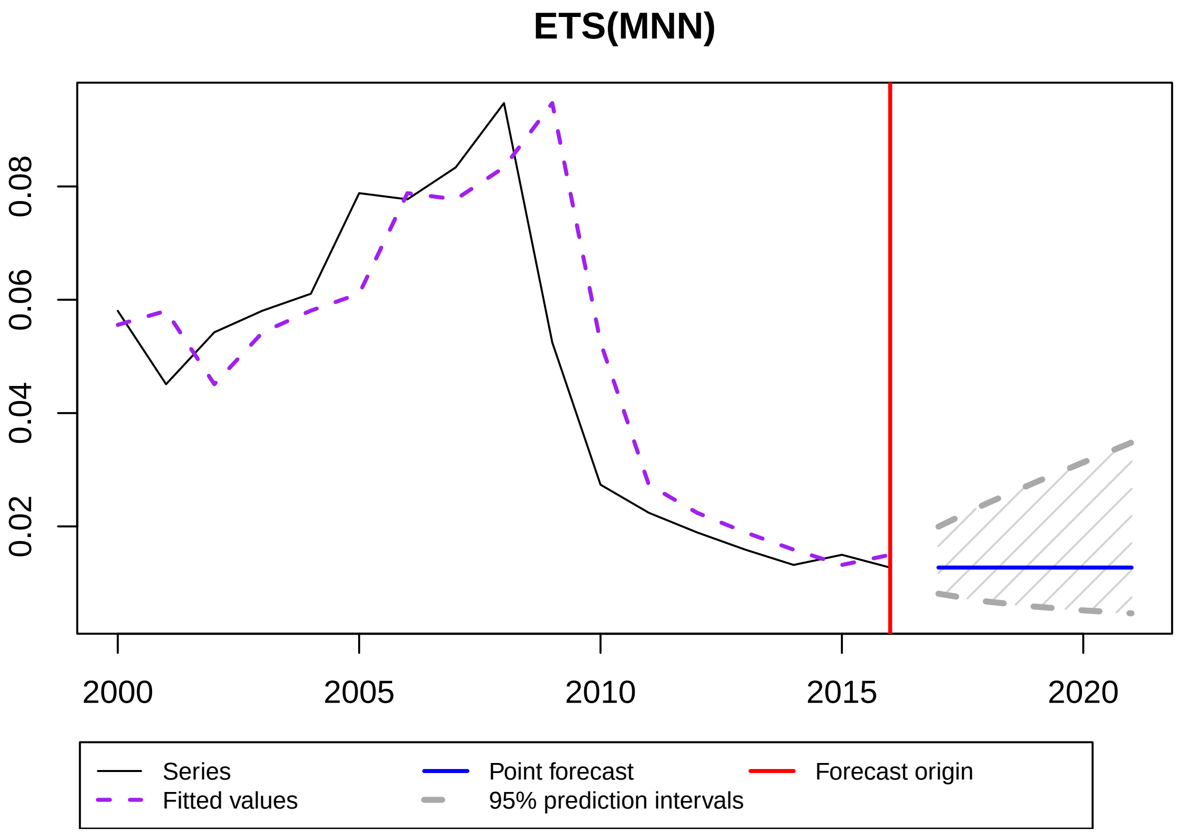

ETS(ANN)
120
100
80
60
40
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin

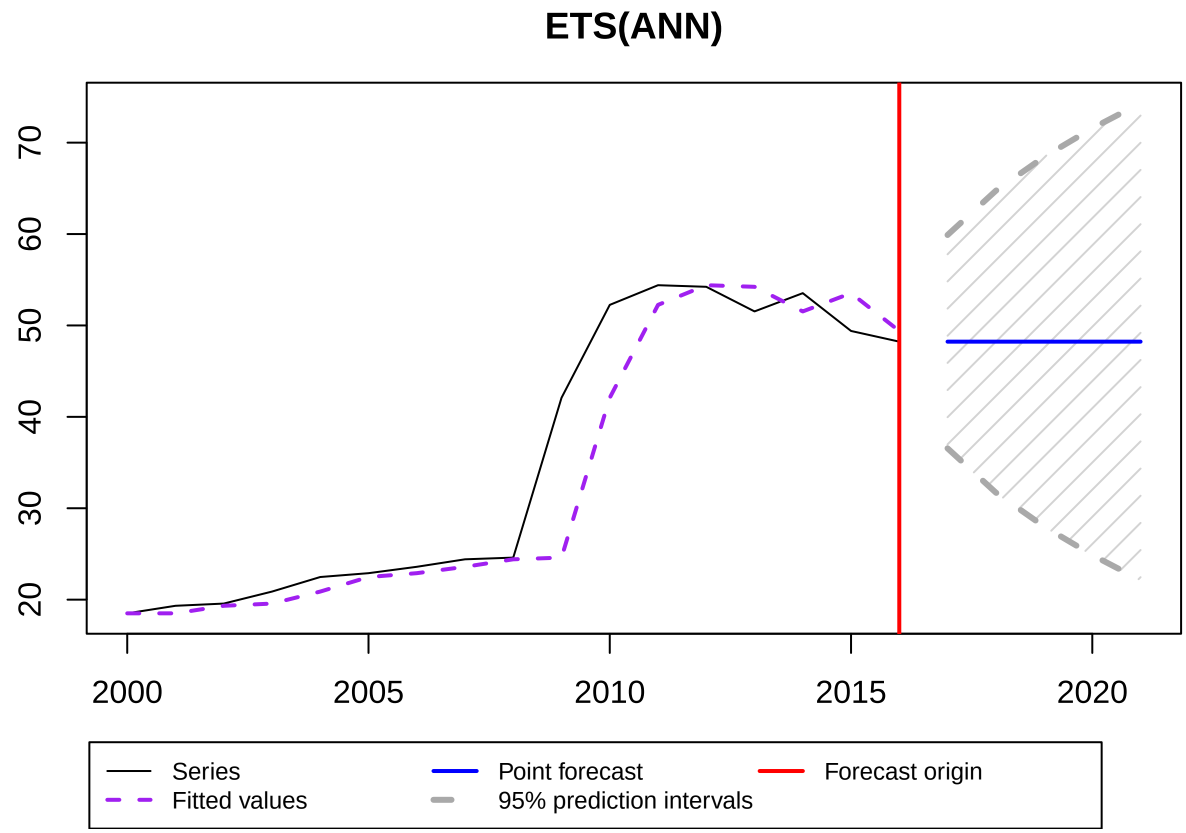

ETS(ANN)
0.10
0.15
0.20
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin

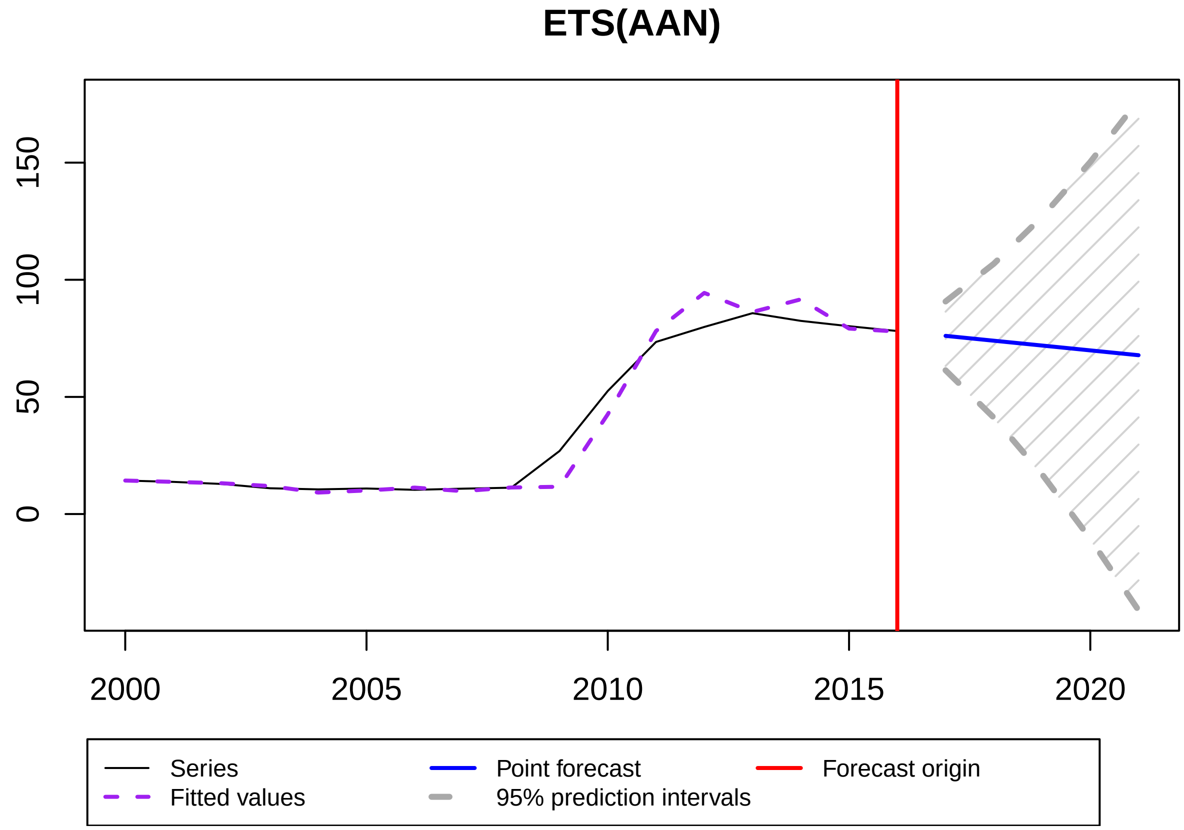

ETS(MNN)
0
5
10
15
20
25
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin

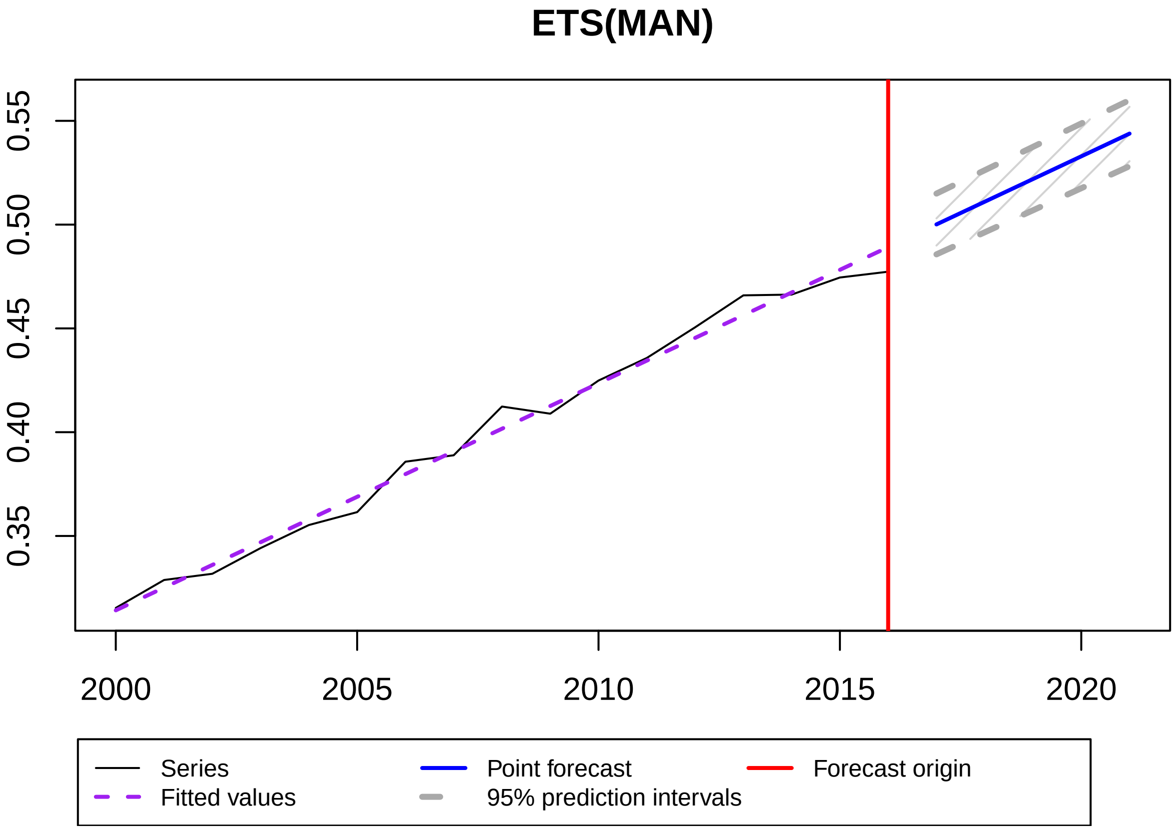

ETS(AAN)
0.3
0.2
0.1
0.0
−0.1
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin

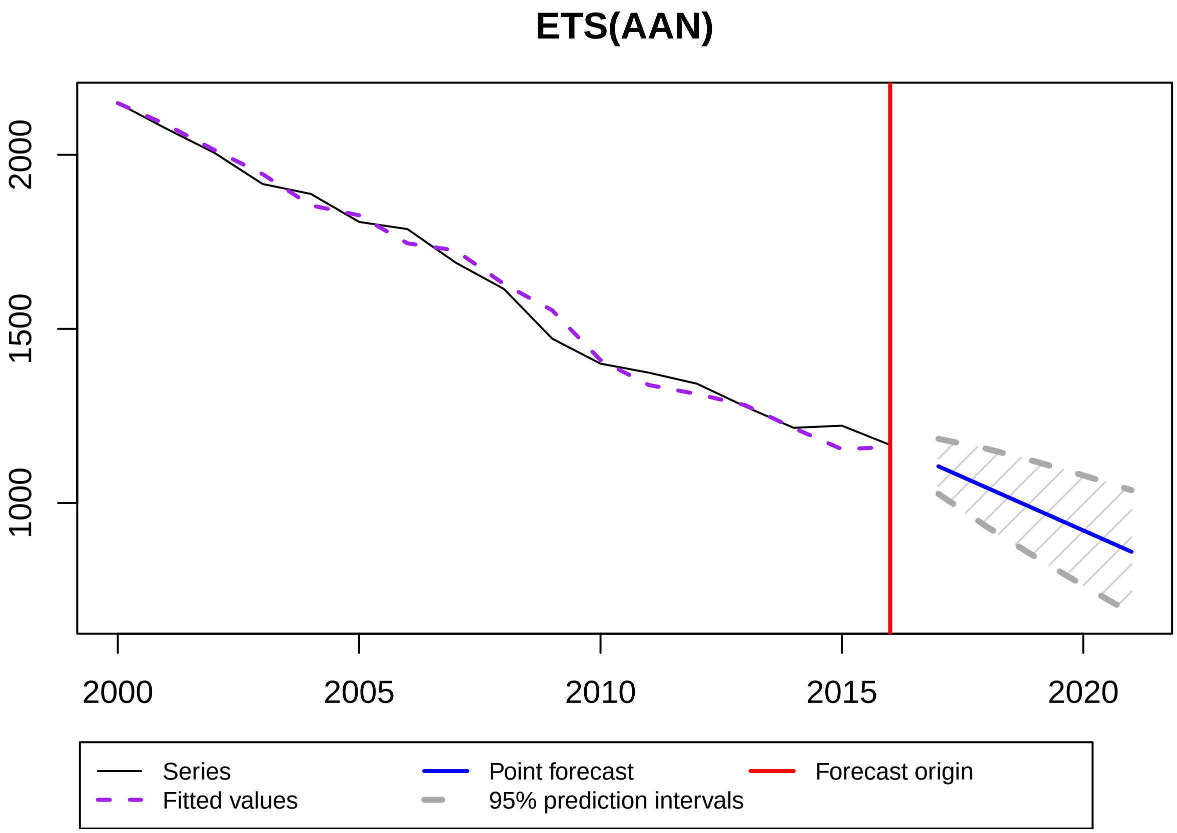

ETS(MNN)
40
30
20
10
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin


ETS(MNN)
0.02
0.04
0.06
0.08
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin


ETS(ANN)
70
60
50
40
30
20
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin

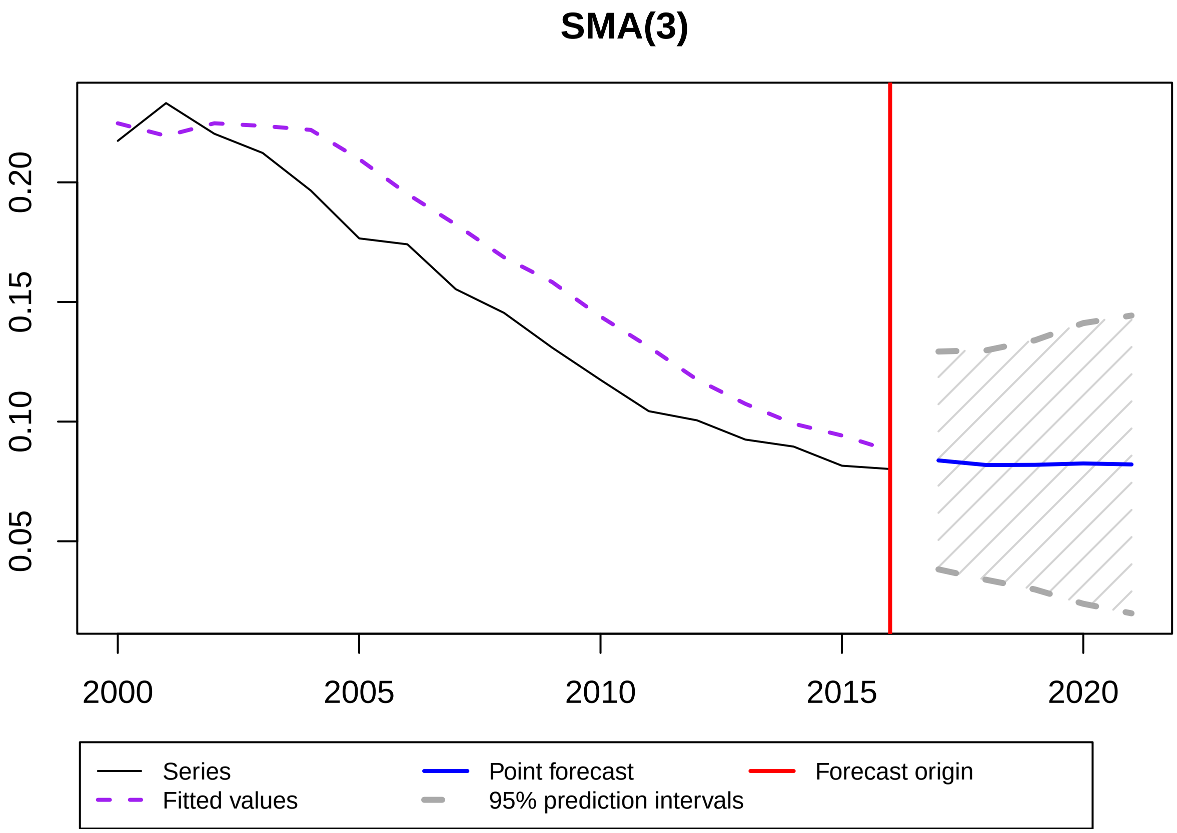

ETS(AAN)
150
100
50
0
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin

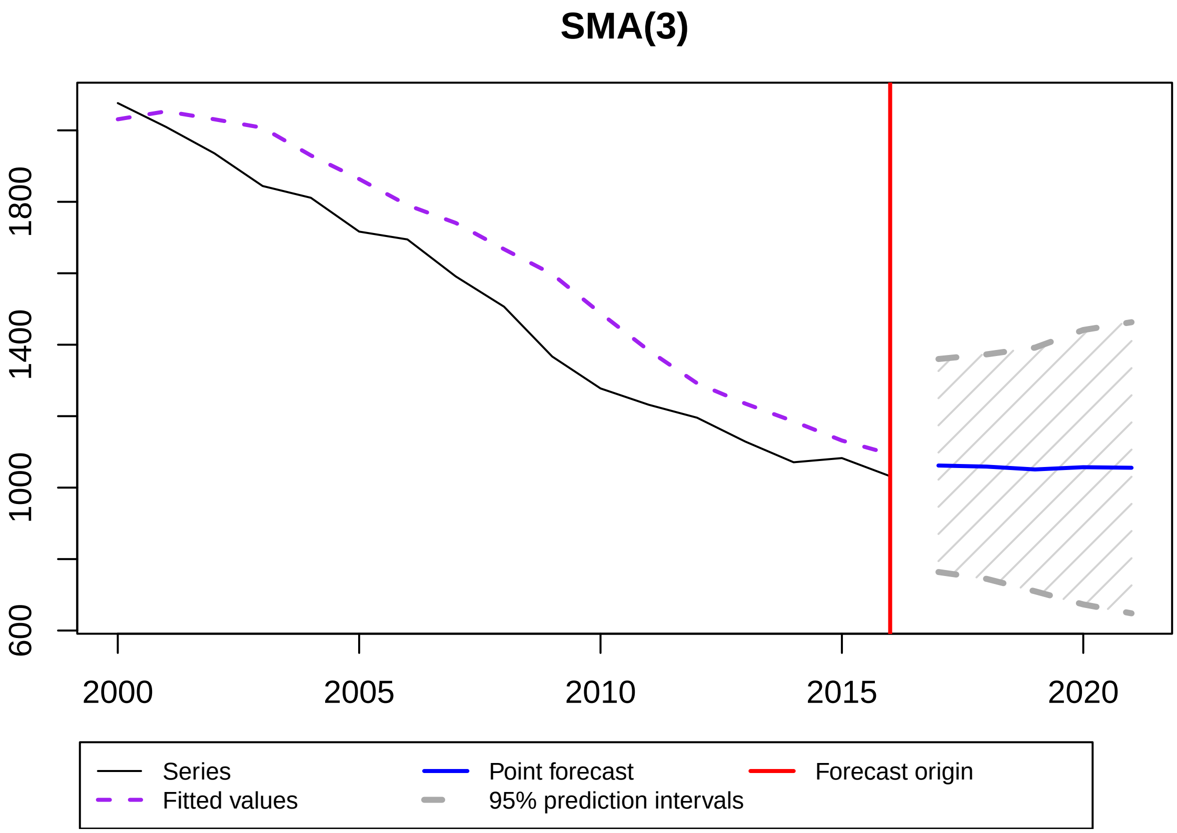

ETS(MAN)
0.35
0.40
0.45
0.50
0.55
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin

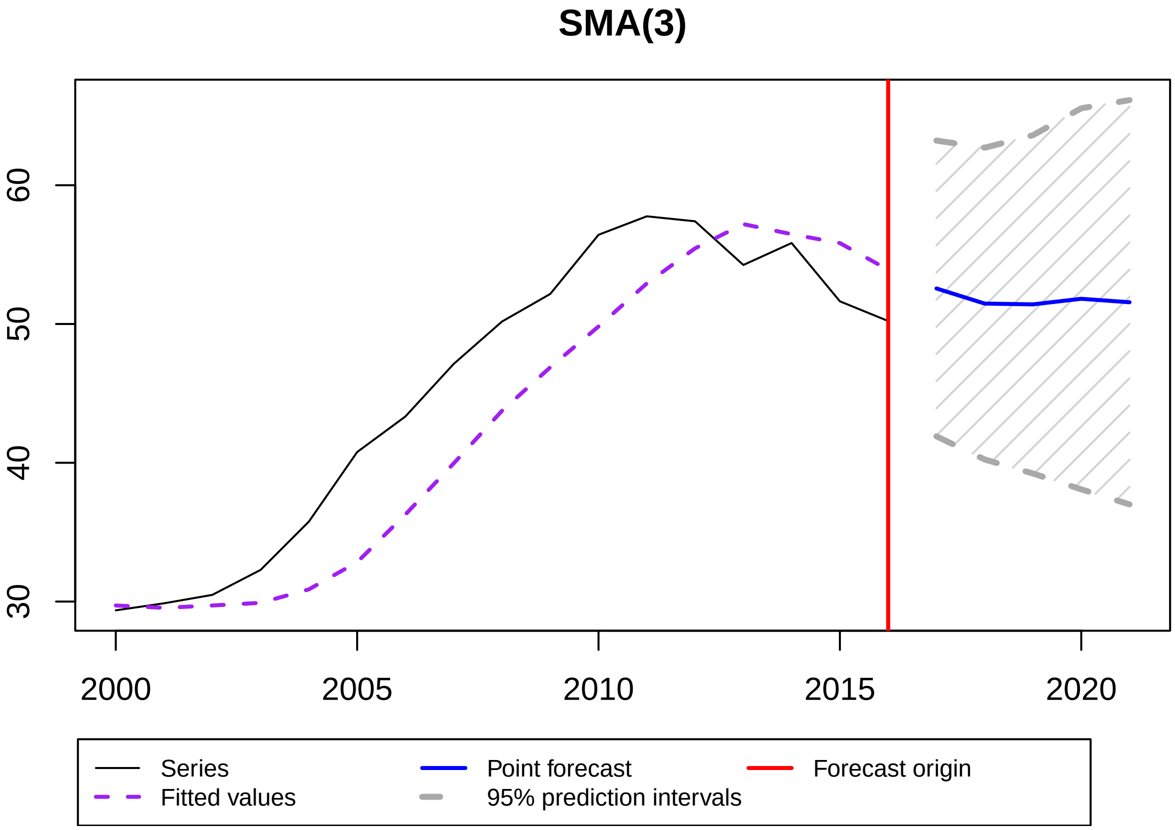


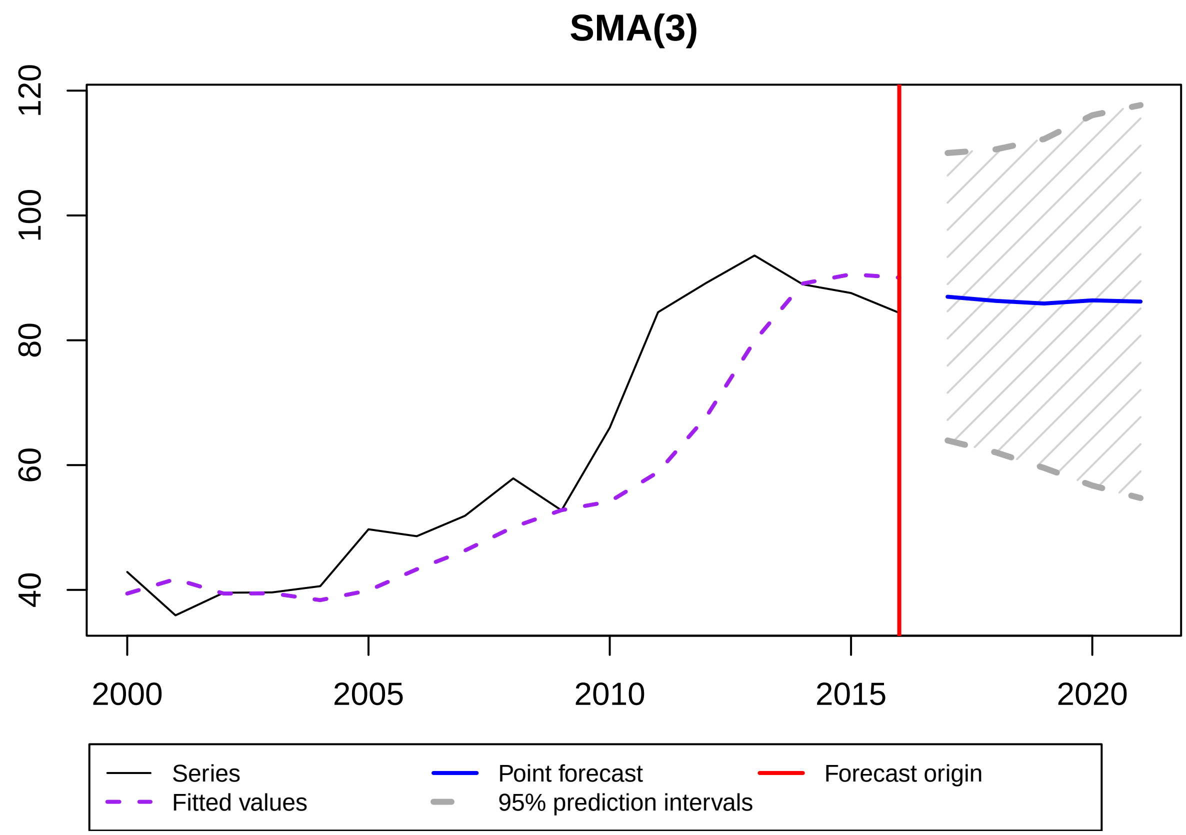
Simple Moving Average

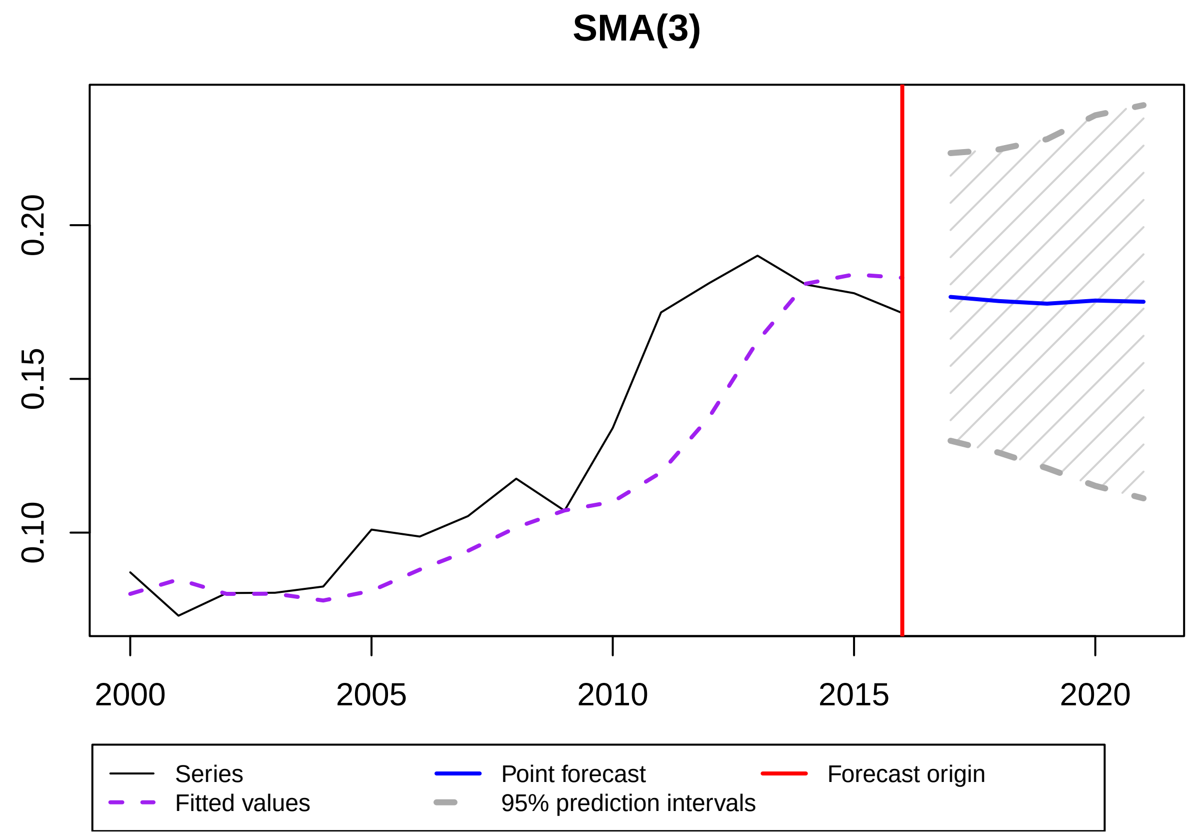
```
smaErrors<-c()
for (i in c(1:13)){
  #Generating and plotting model
  sma_model<-sma(trainTotals[[i]], h=5, order=3, holdout=FALSE, interval=TRUE, silent='output')

  #Getting MSE (the head and tail are used to get from 2017-2020)
  prediction<-sma_model$forecast%>%as.numeric()%>%tail(5)%>%head(4)
  test<-testTotals[[i]]%>%as.numeric()

  ##Saving MSE in smaError vector
  smaErrors<-c(smaErrors,MSE(prediction,test))
}
```

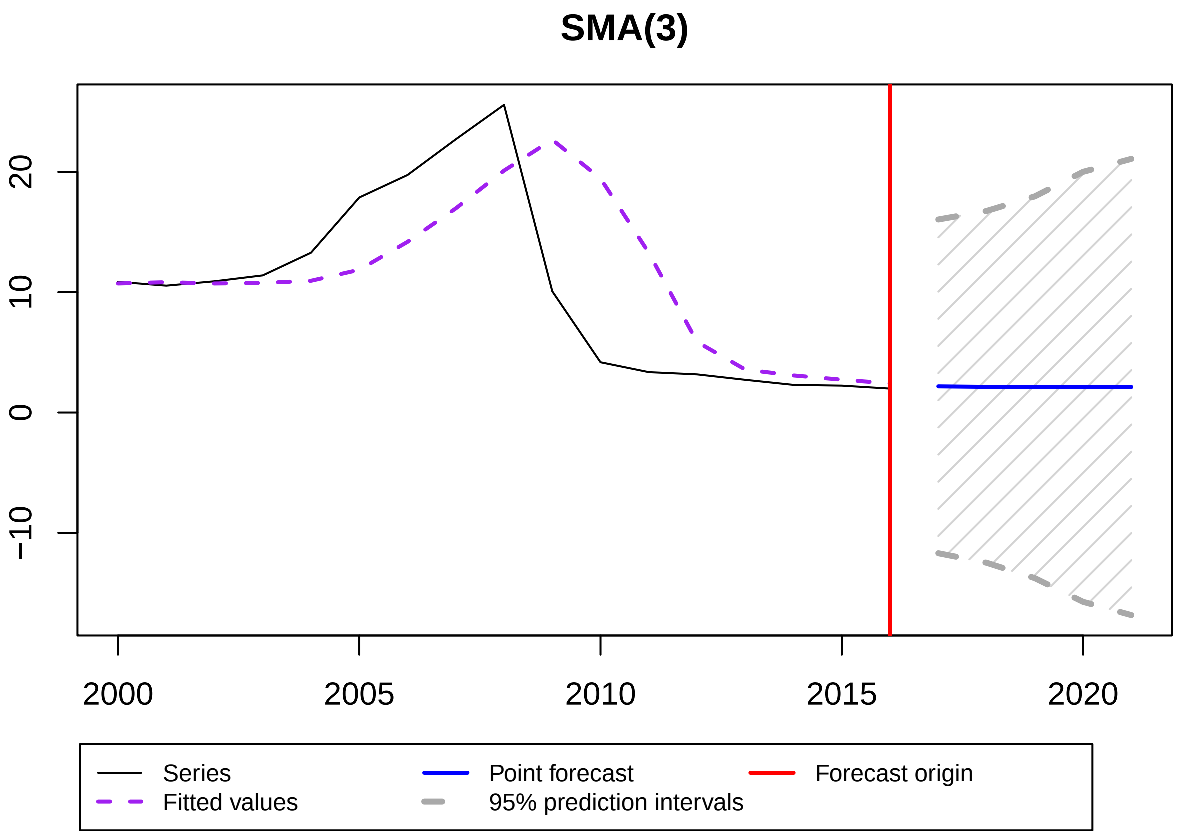

SMA(3)
0.05
0.10
0.15
0.20
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin

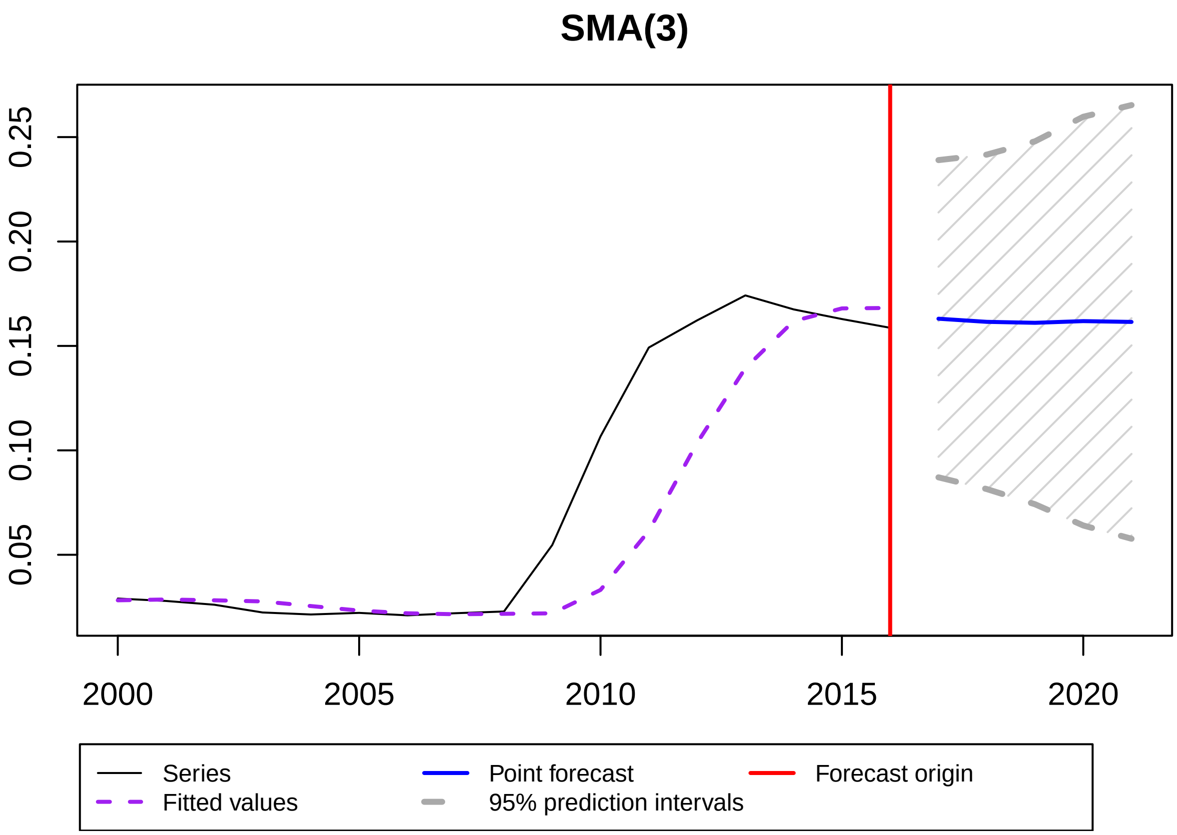

SMA(3)
600
1000
1400
1800
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin


SMA(3)
60
50
40
30
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin

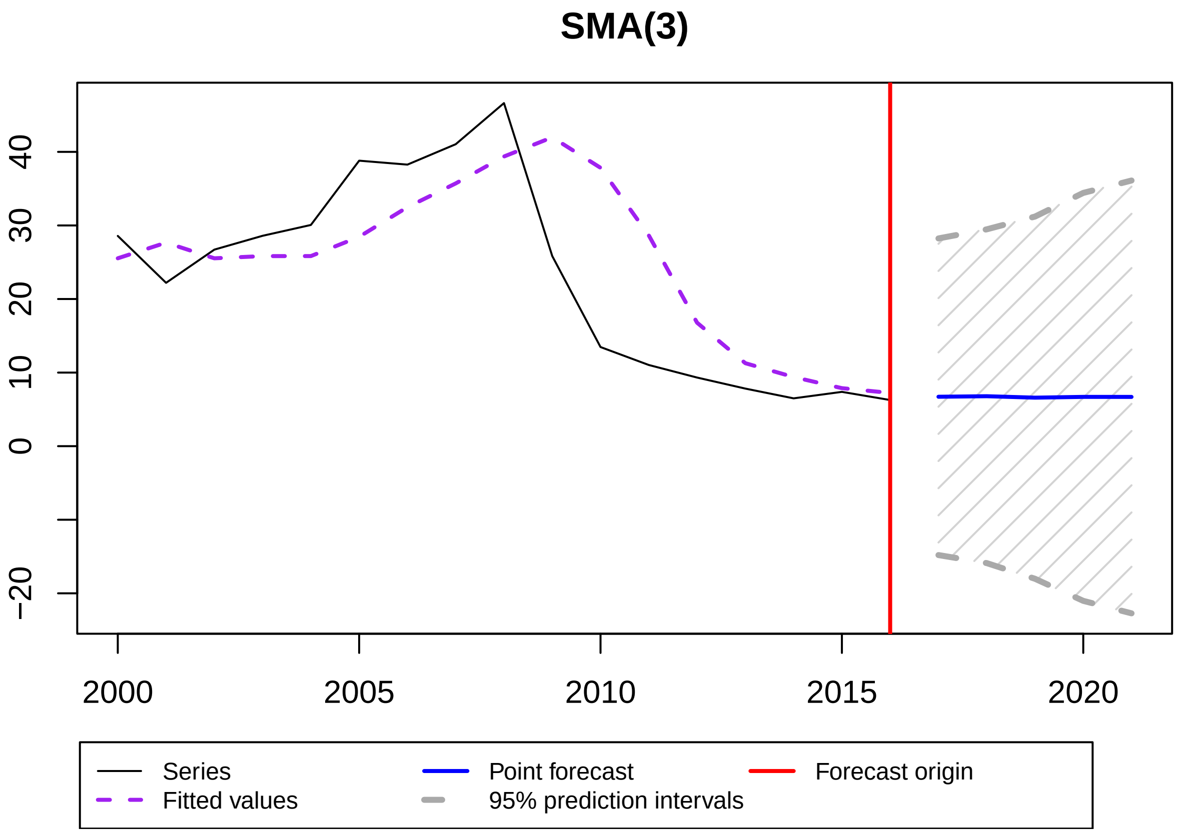

SMA(3)
120
100
80
60
40
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin

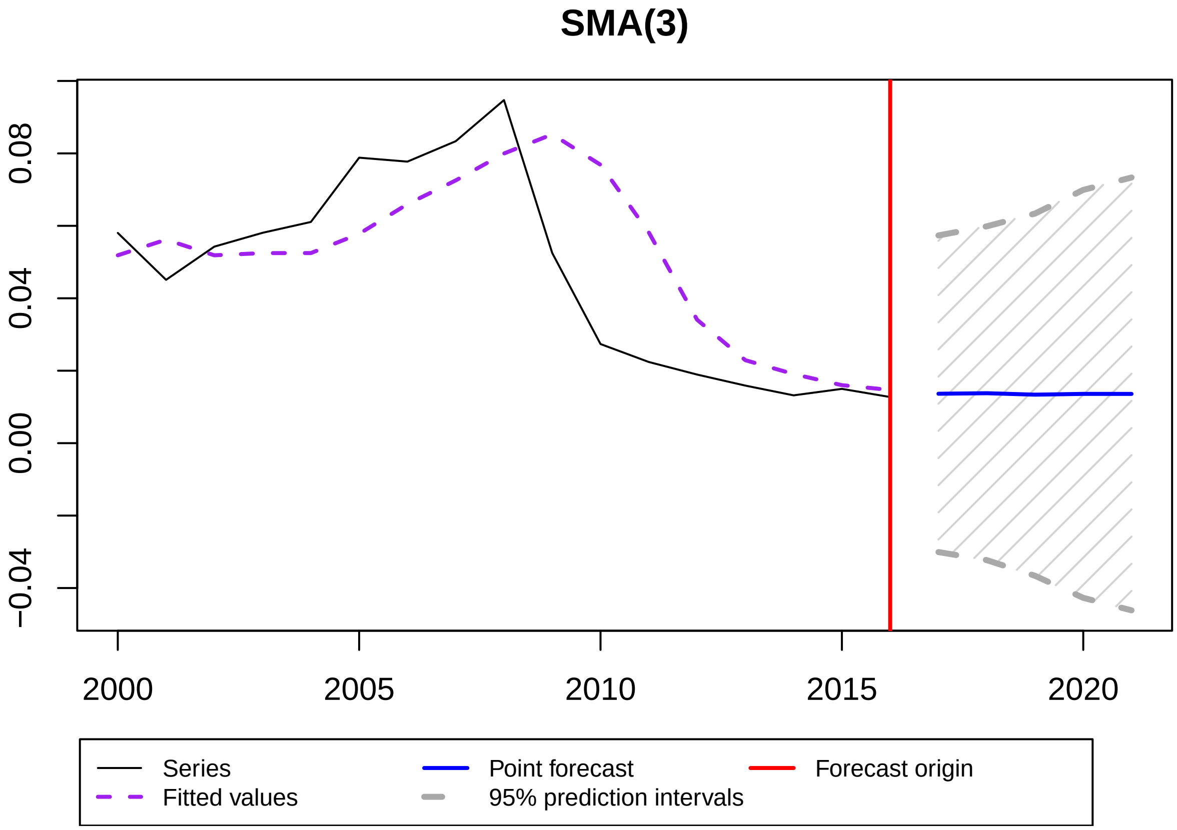

SMA(3)
0.10
0.15
0.20
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin

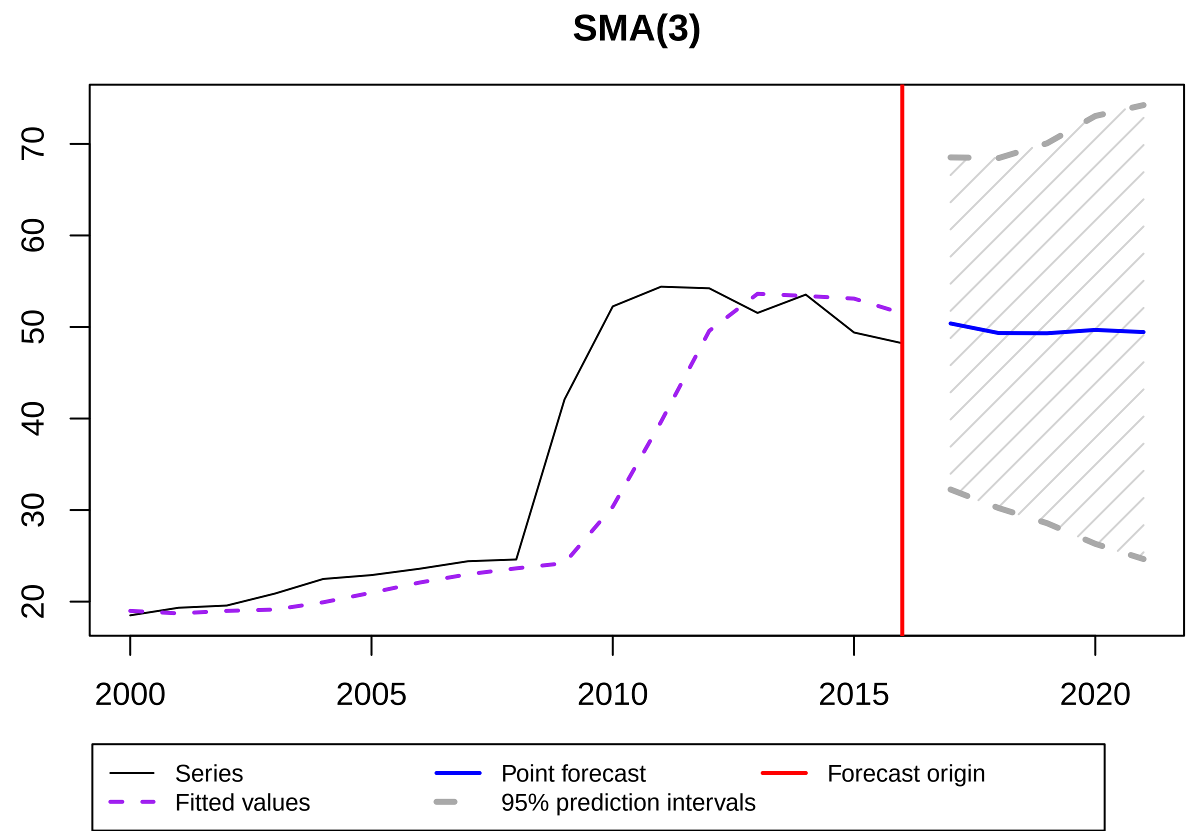

SMA(3)
20
10
0
−10
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin

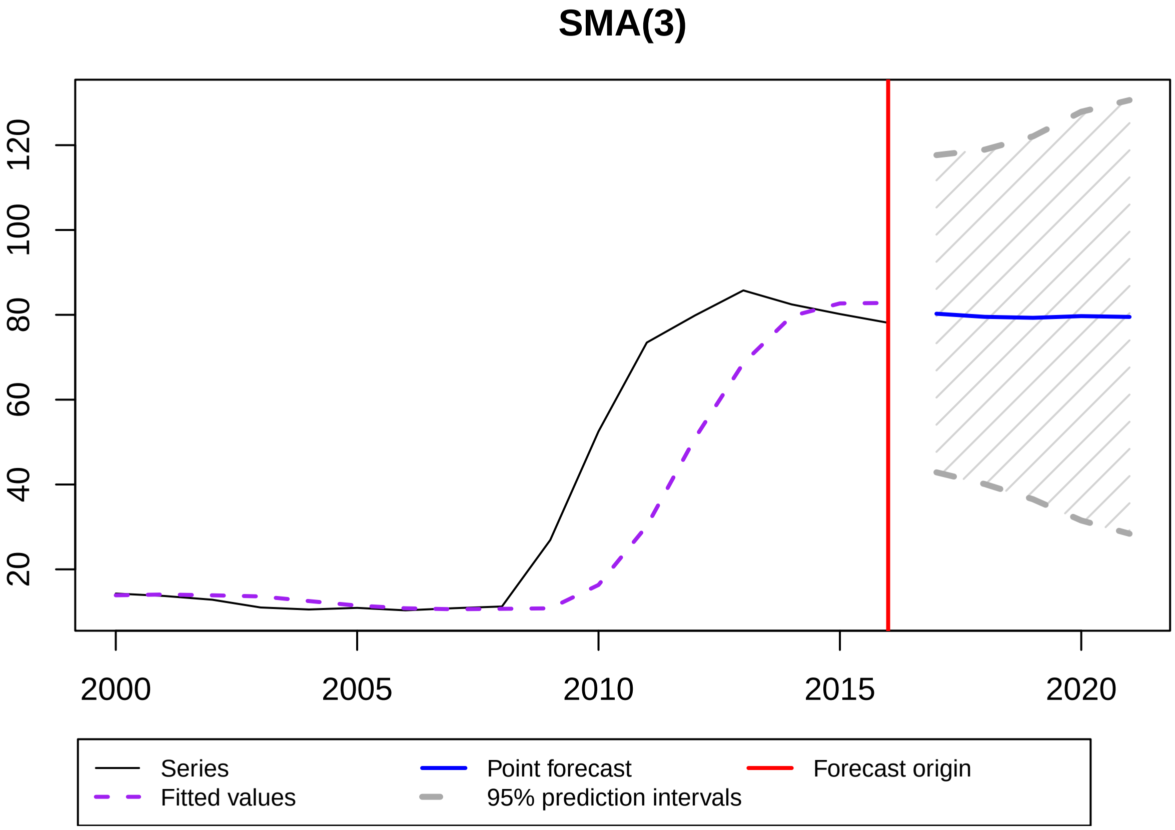


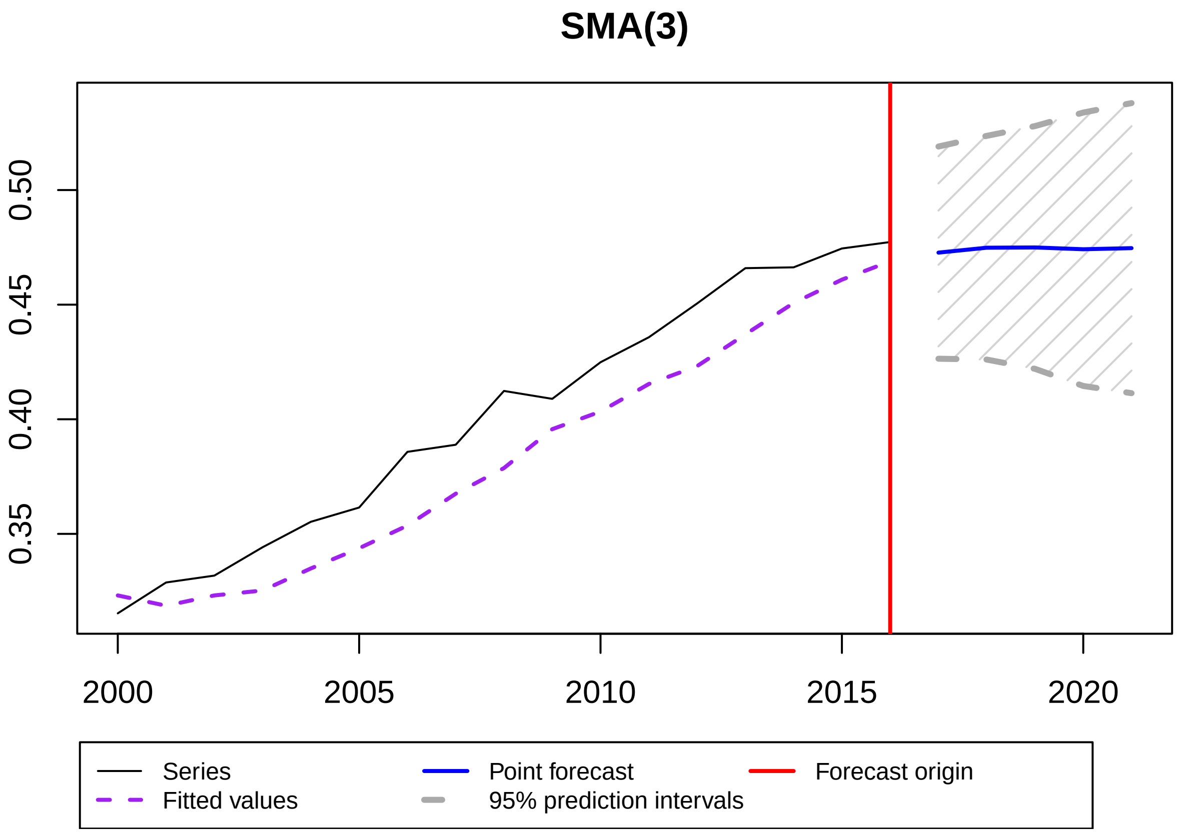
SMA(3)
0.05
0.10
0.15
0.20
0.25
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin

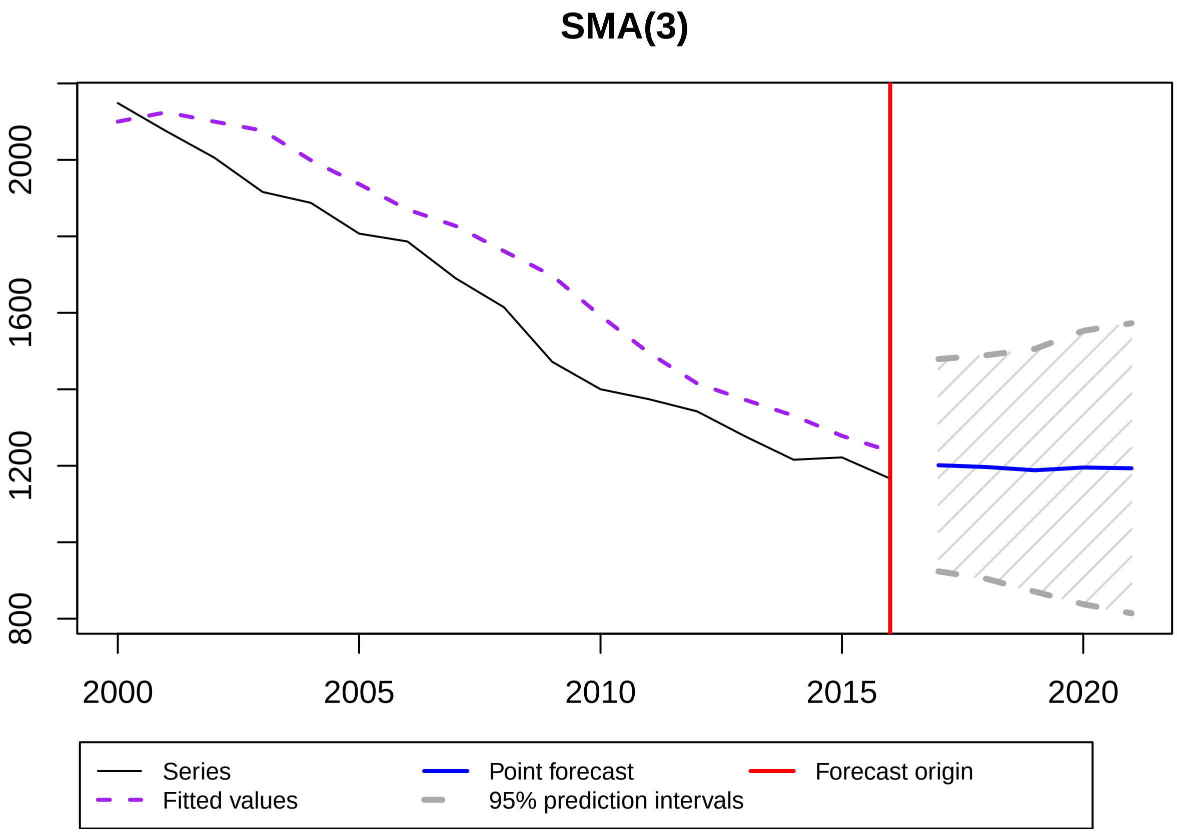

SMA(3)
40
30
20
10
0
−20
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin


SMA(3)
0.08
0.04
0.00
−0.04
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin


SMA(3)
70
60
50
40
30
20
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin

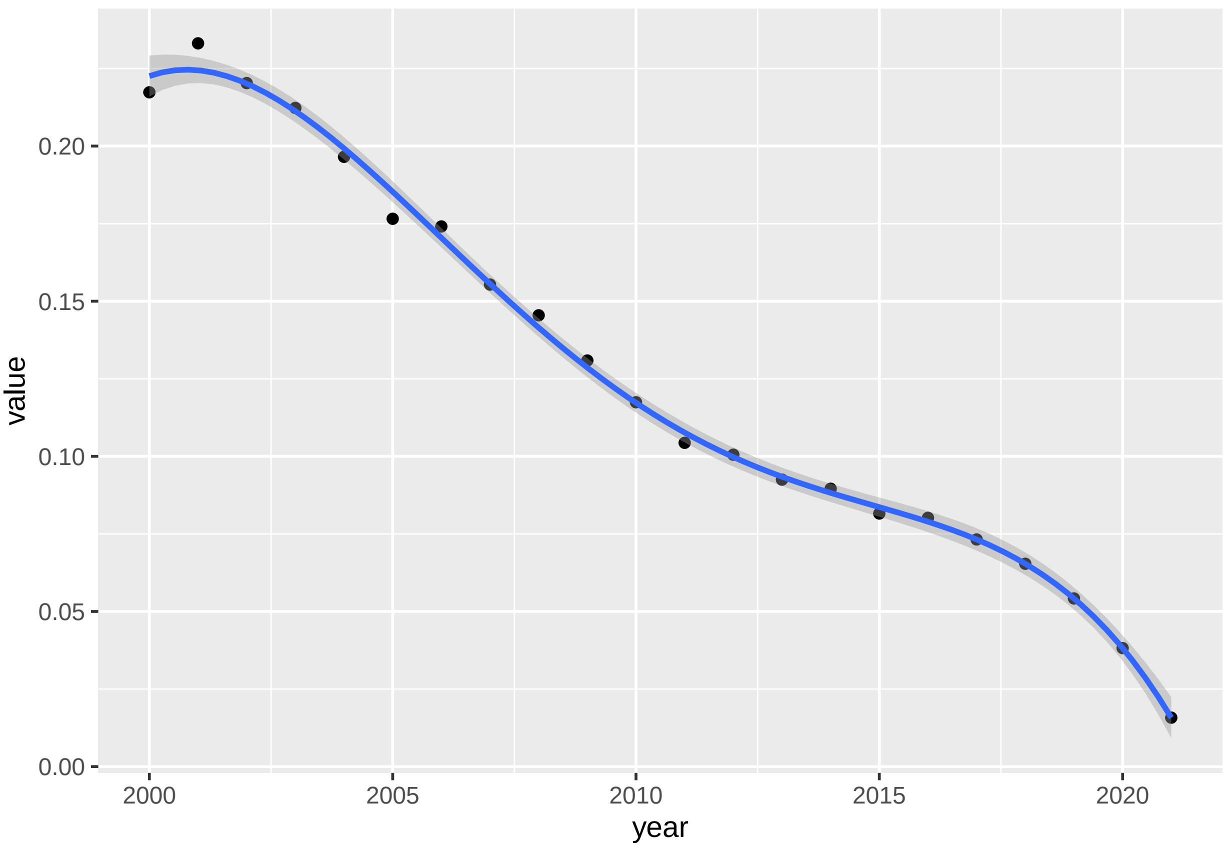

SMA(3)
120
100
80
60
40
20
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin

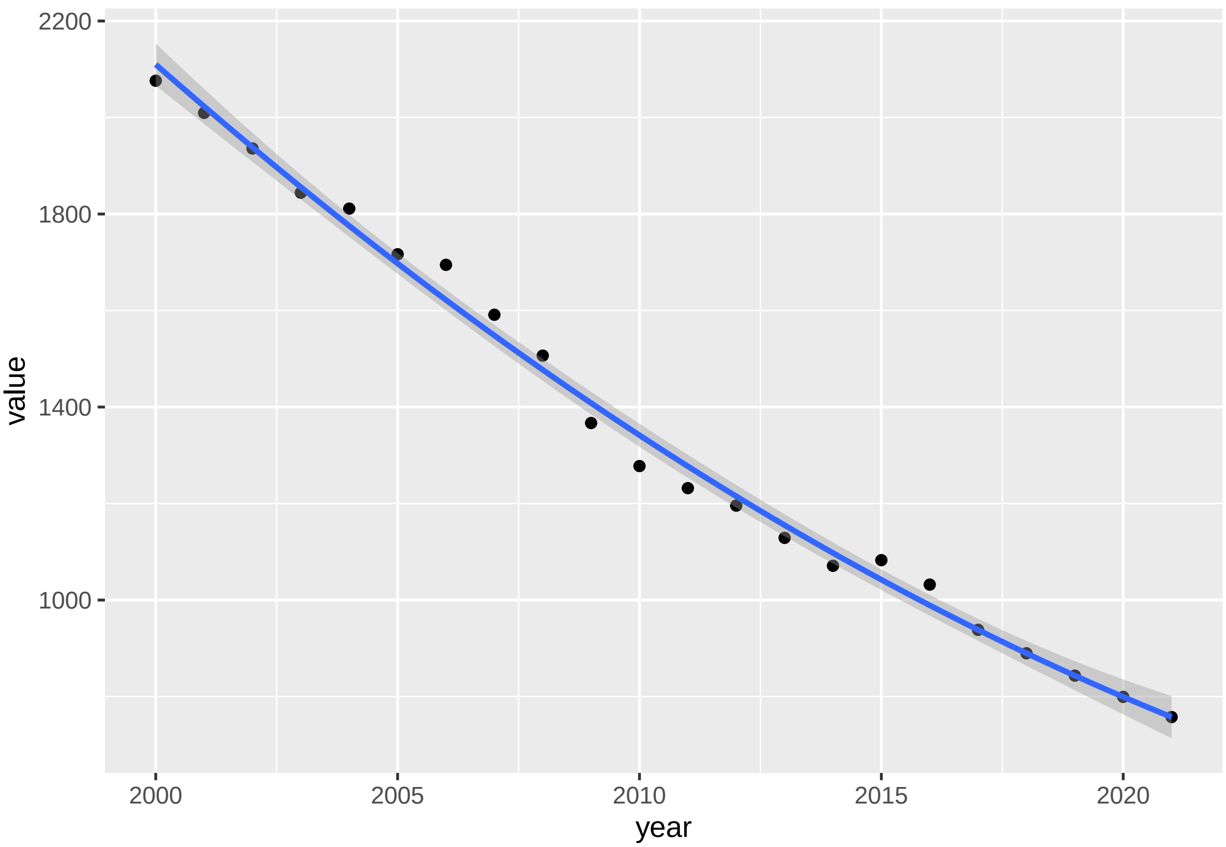

SMA(3)
0.35
0.40
0.45
0.50
2000
2005
2010
2015
2020
Series
Fitted values
Point forecast
95% prediction intervals
Forecast origin



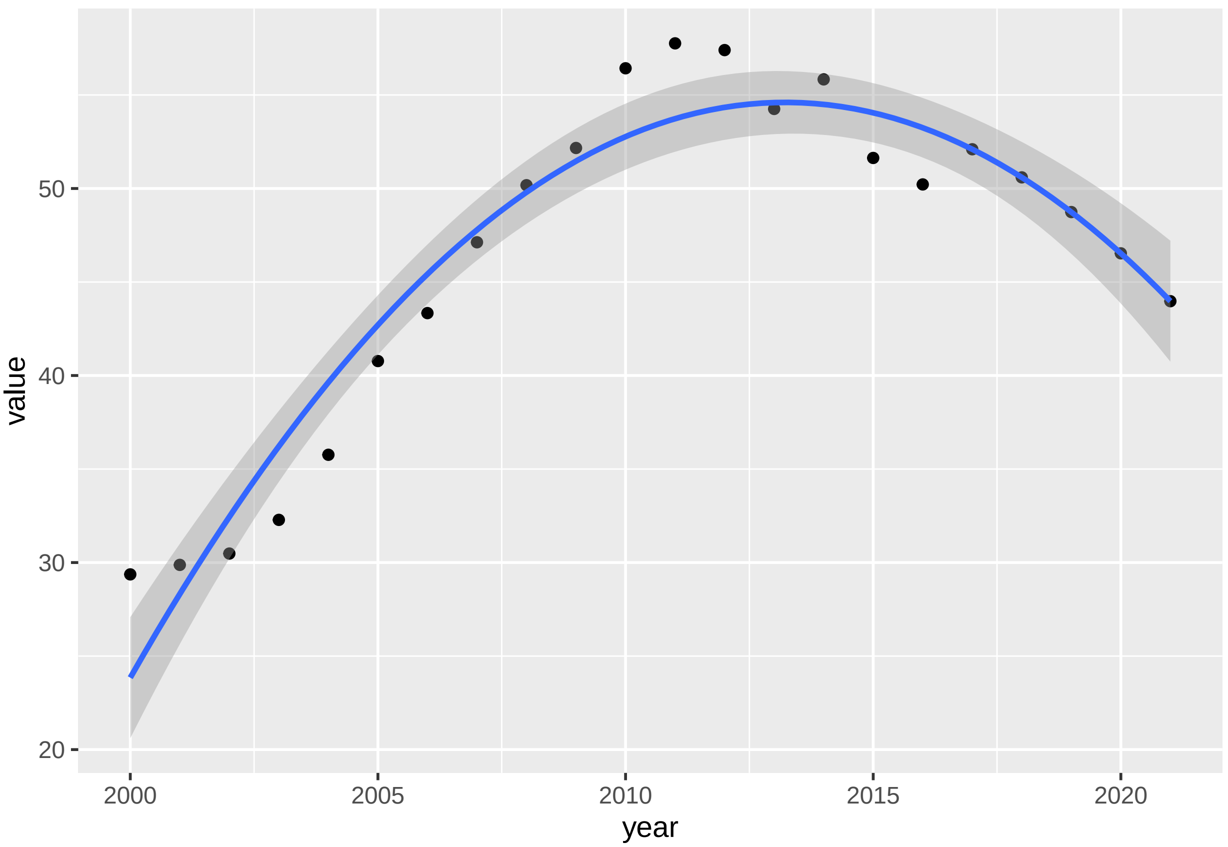
Polinomial Regression

```
polErrors<-c()
for(h in c(1:13)){
  #Load and plot the data
  polydf <- data.frame(year=c(2000:2016),value=trainTotals[[h]]%>%as.numeric())

  #randomly shuffle data
  polydf.shuffled <- polydf[sample(nrow(polydf)),]

  #define number of folds to use for k-fold cross-validation
  K <- 10

  #define degree of polynomials to fit
  degree <- 5

  #create k equal-sized folds
  folds <- cut(seq(1, nrow(polydf.shuffled)) , breaks=K , labels=FALSE)

  #create object to hold MSE's of models
  mse = matrix(data=NA,nrow=K,ncol=degree)

  #Perform K-fold cross validation
  for(i in 1:K){
      #define training and testing data
      testData <- data.frame(year=c(2017:2020),value=testTotals[[h]]%>%as.numeric())
      trainData <- data.frame(year=c(2000:2016),value=trainTotals[[h]]%>%as.numeric())

      #use k-fold cv to evaluate models
      for (j in 1:degree){
          fit.train = lm(value ~ poly(year,j), data=trainData)
          fit.test = predict(fit.train, newdata=testData)
          mse[i,j] = mean((fit.test-testData$value)^2)
      }
  }

  #find MSE for each degree
  mmse =colMeans(mse)
  #determine which is the better degree
  mdegree = which.min(mmse)

  # Make predictions
  model <- lm(value ~ poly(year, mdegree), data = polydf)
  predictions <- model %>% predict(data.frame('year'=c(2017:2021)))
  predictionsdf <- data.frame('year' = c(2017: 2021), 'value' = predictions)
  totaldf <- rbind(polydf, predictionsdf )

  print(ggplot(totaldf, aes(x=year, y=value)) +
            geom_point() +
            stat_smooth(method='lm', formula = y ~ poly(x,mdegree), size = 1)+
            xlab('year') +
            ylab('value'))

  #Saving MSE in vector
  polErrors<-c(polErrors,MSE(predictions[1:4],testTotals[[i]]%>%as.numeric()))
}
```

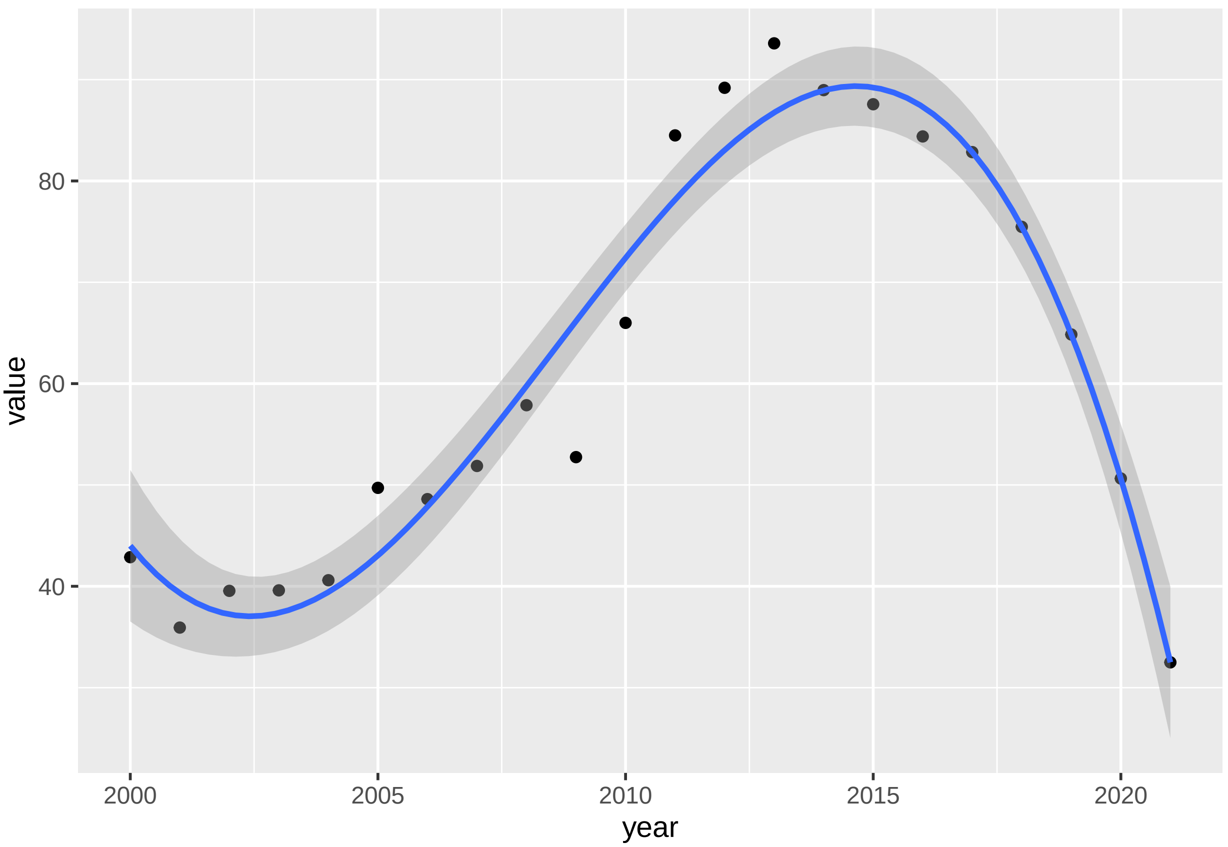

value
0.00
0.05
0.10
0.15
0.20
2000
2005
2010
2015
2020
year

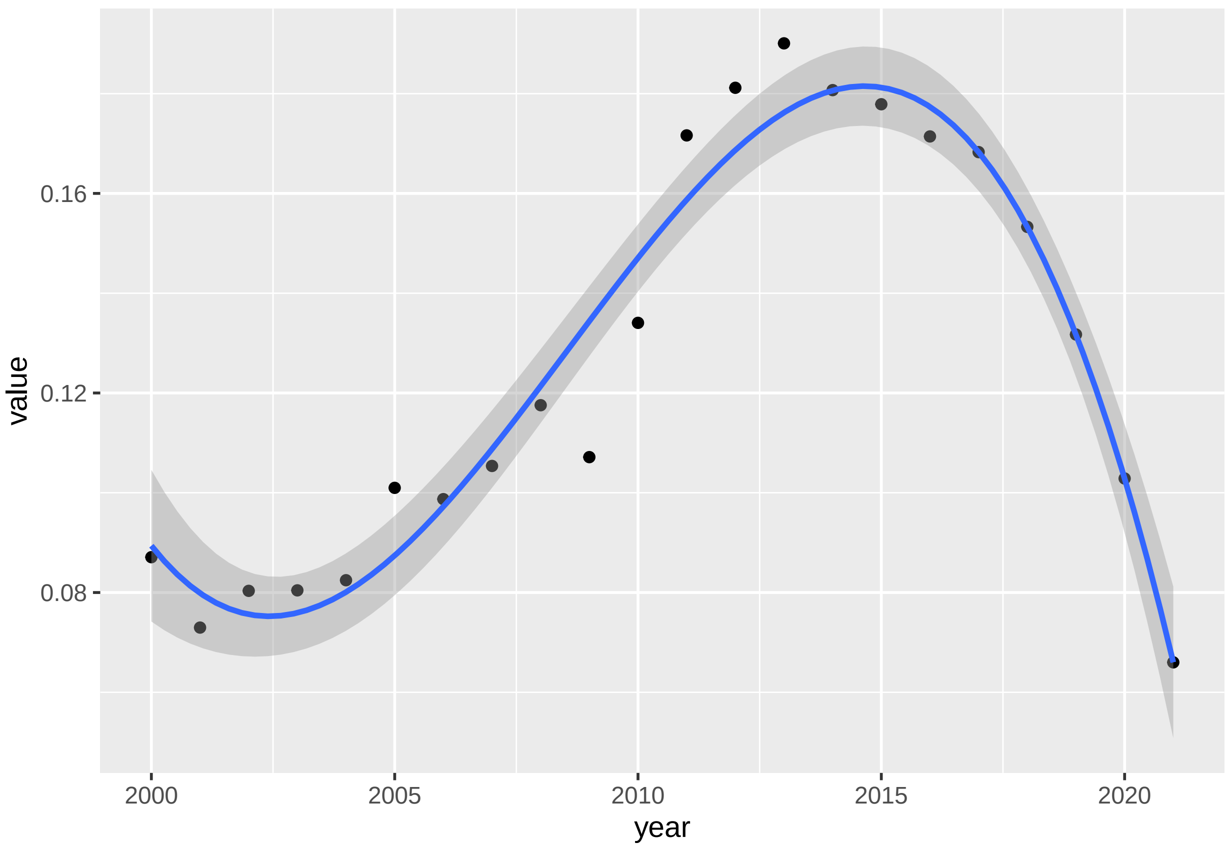


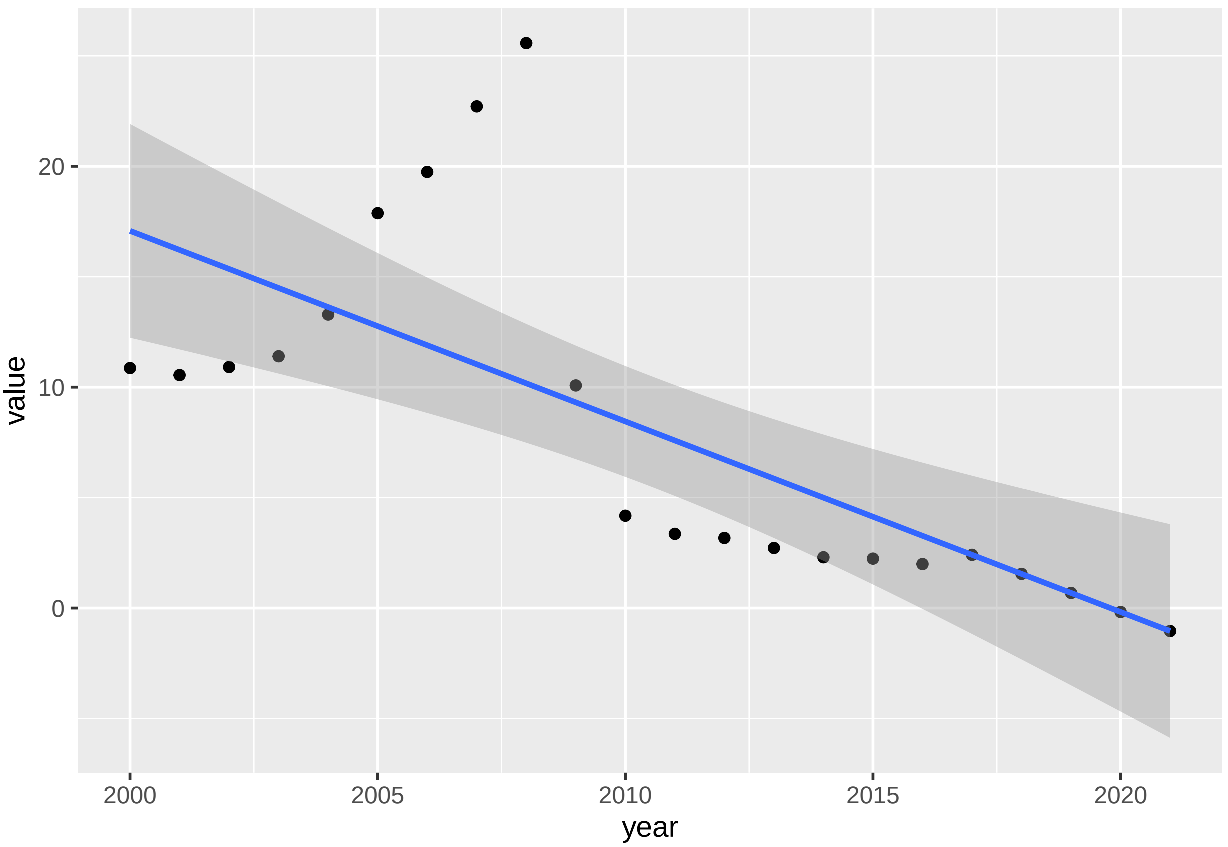
2200
1800
1400
1000
value
2000
2005
2010
2015
2020
year

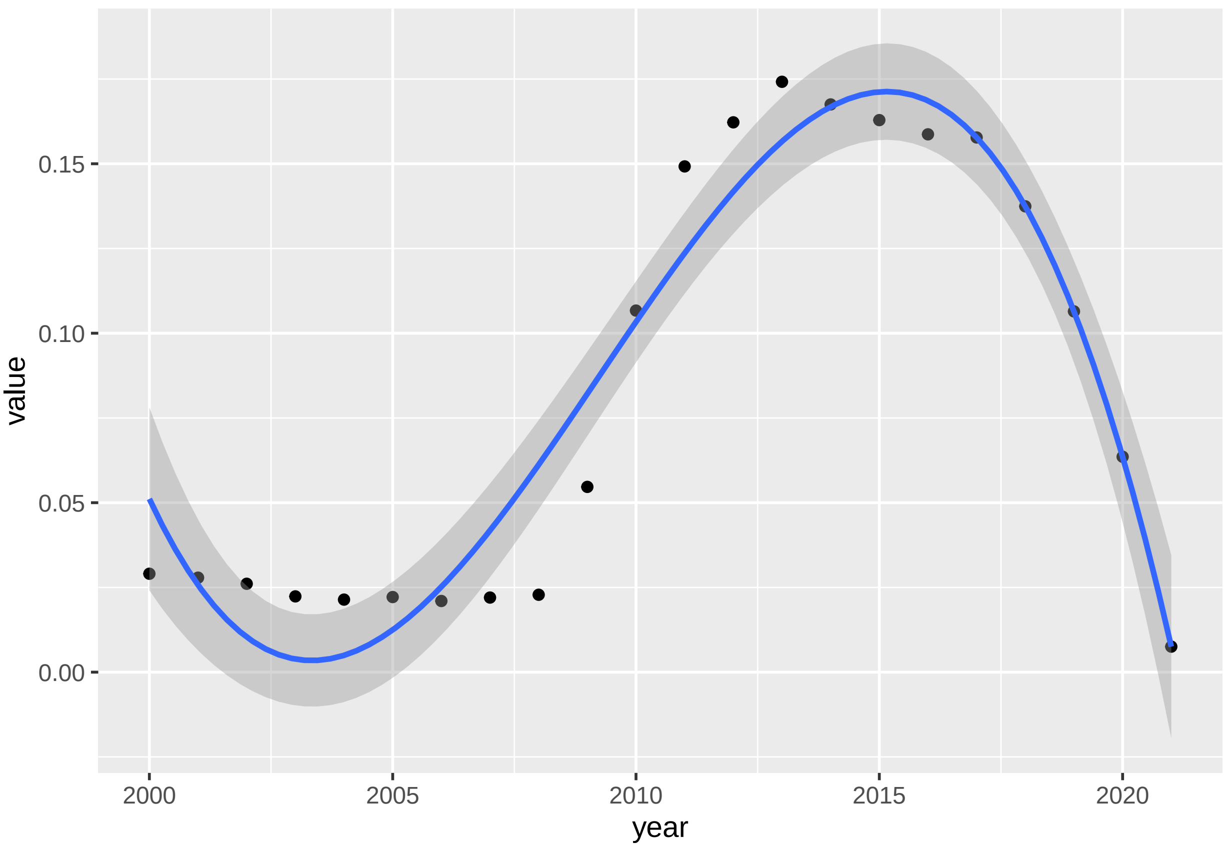


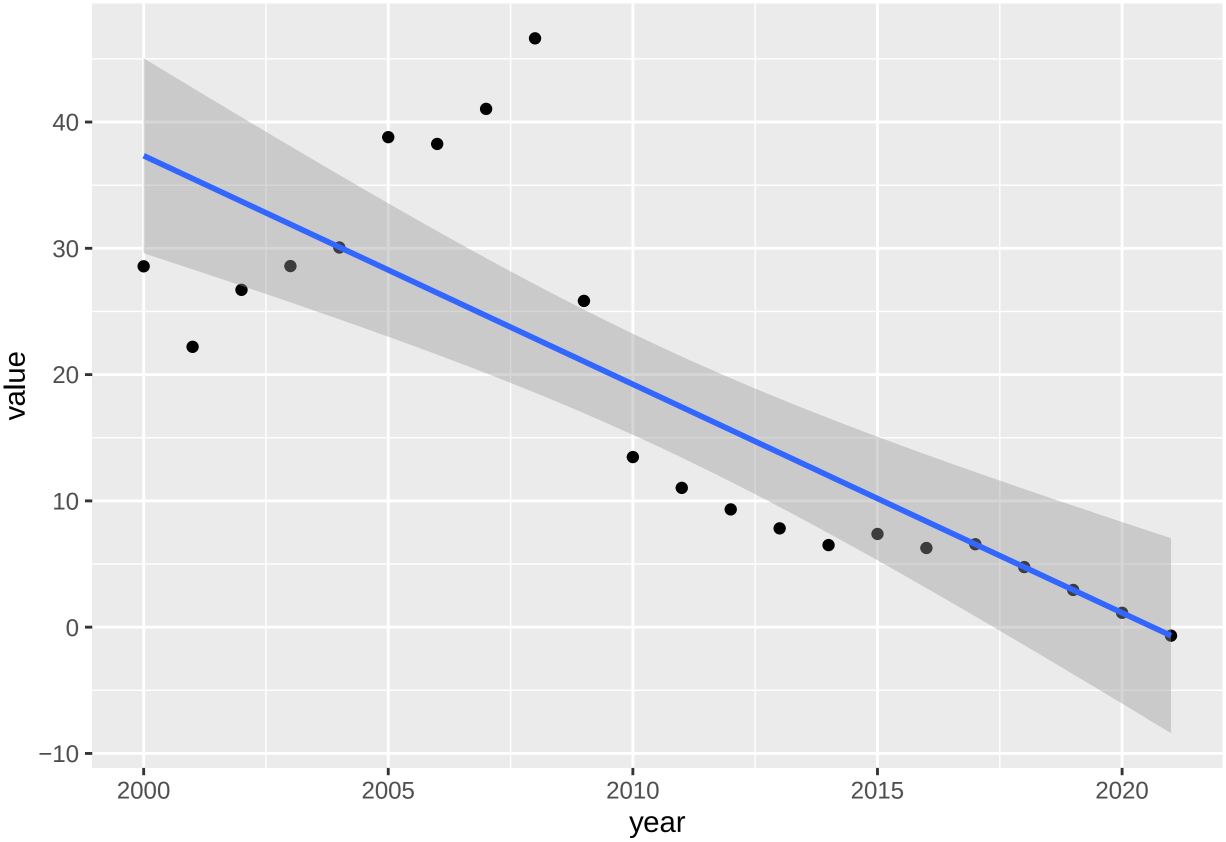
value
50
40
30
20
2000
2005
2010
2015
2020
year

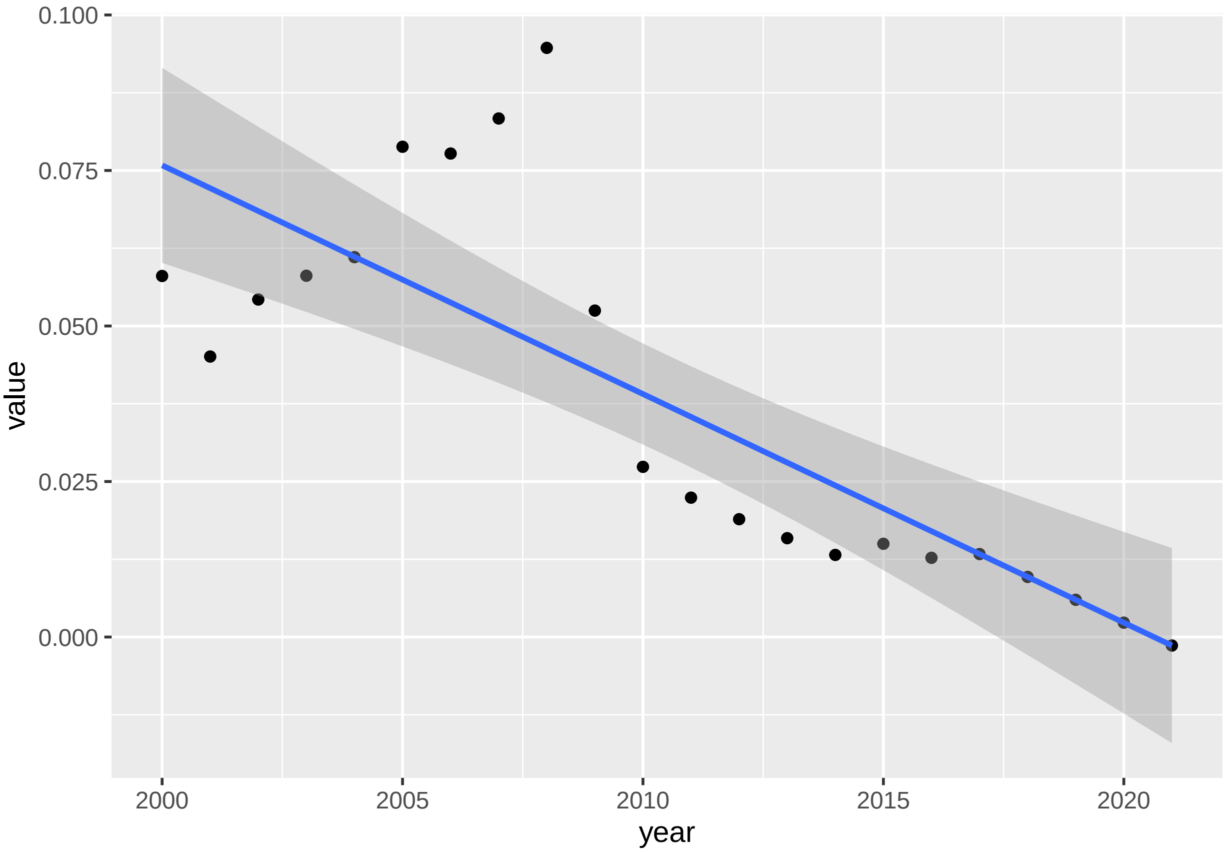

80
60
40
value
2000
2005
2010
2015
2020
year

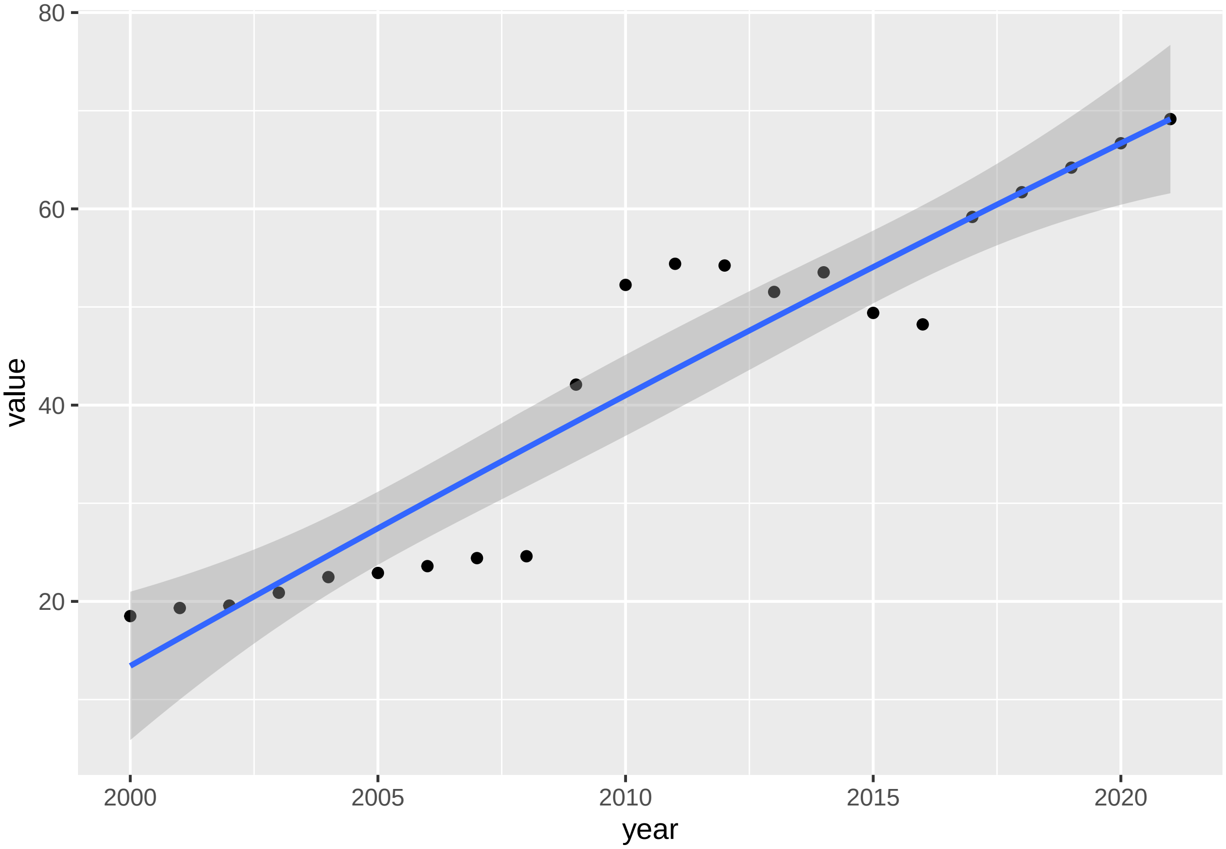

value
0.16
0.12
0.08
2000
2005
2010
2015
2020
year

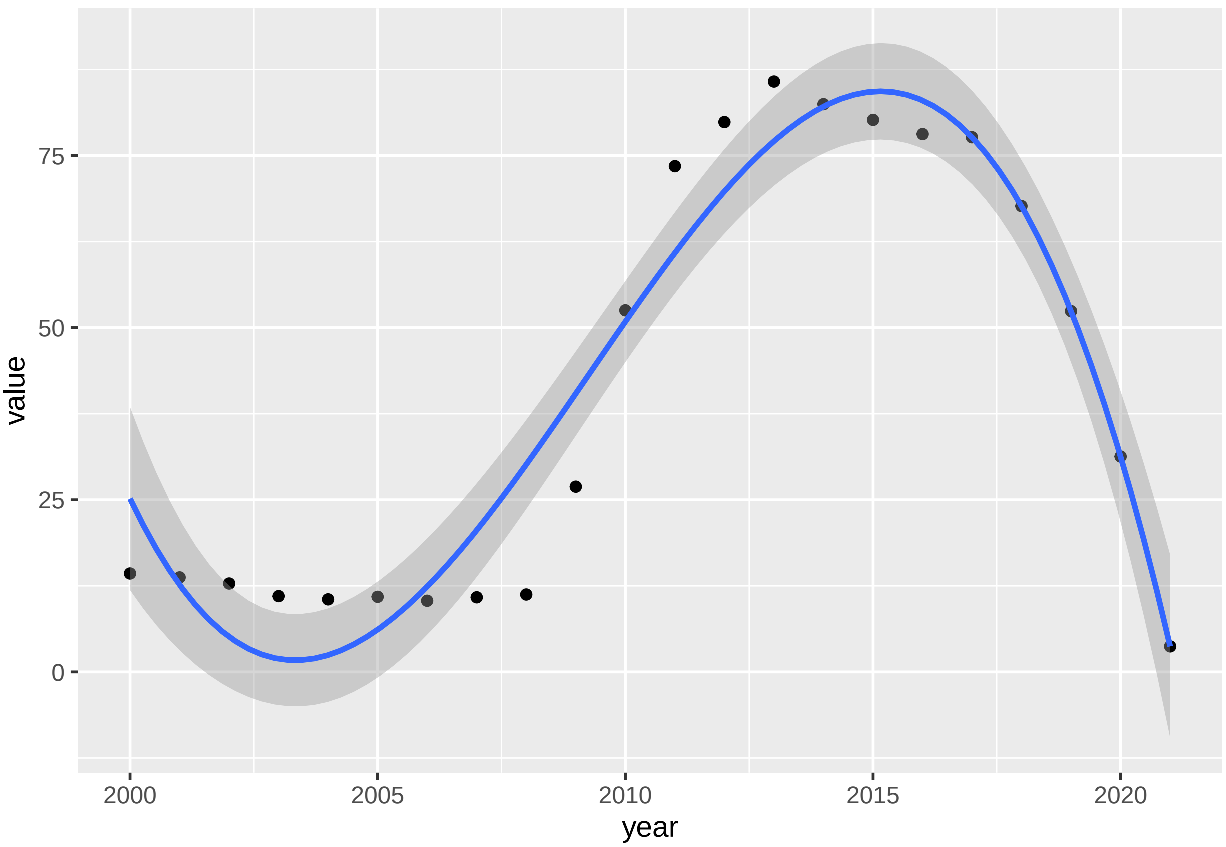

value
20
10
0
2000
2005
2010
2015
2020
year

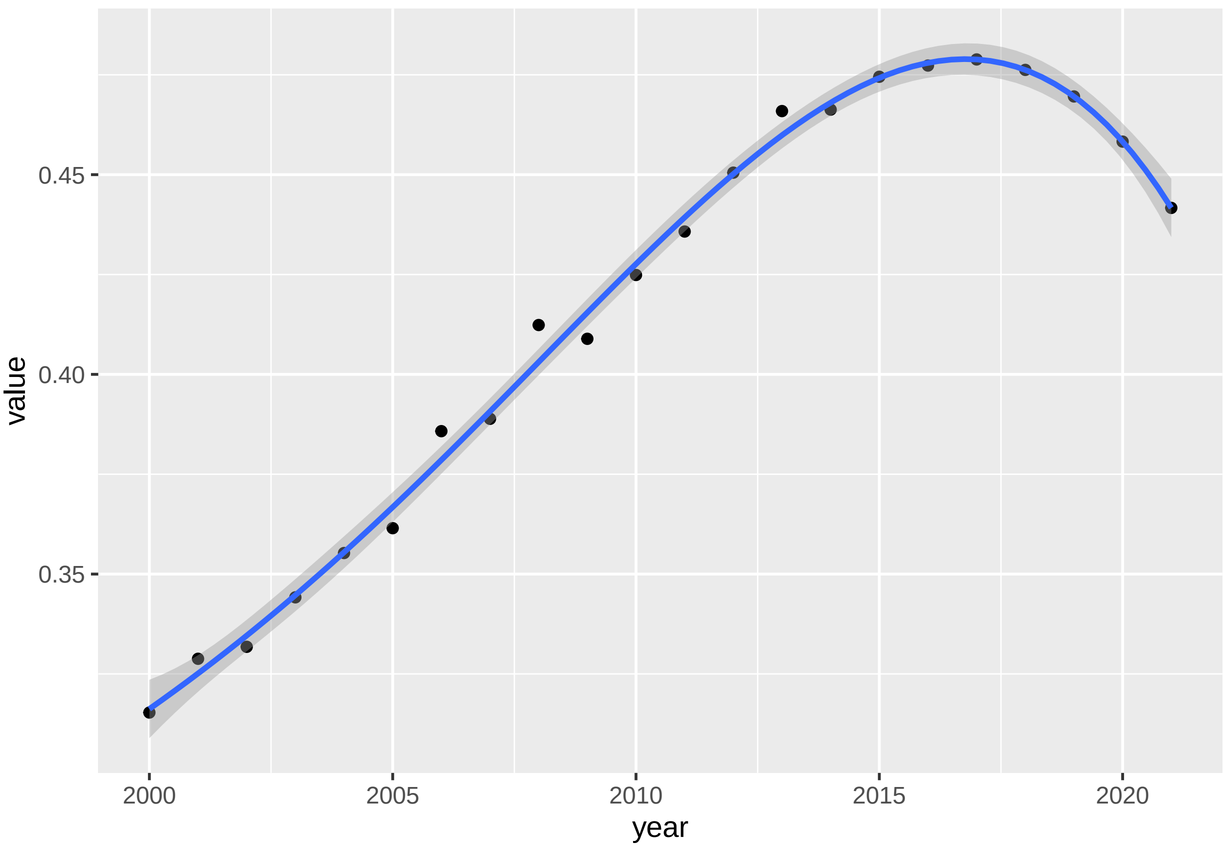

value
0.15
0.10
0.05
0.00
2000
2005
2010
2015
2020
year

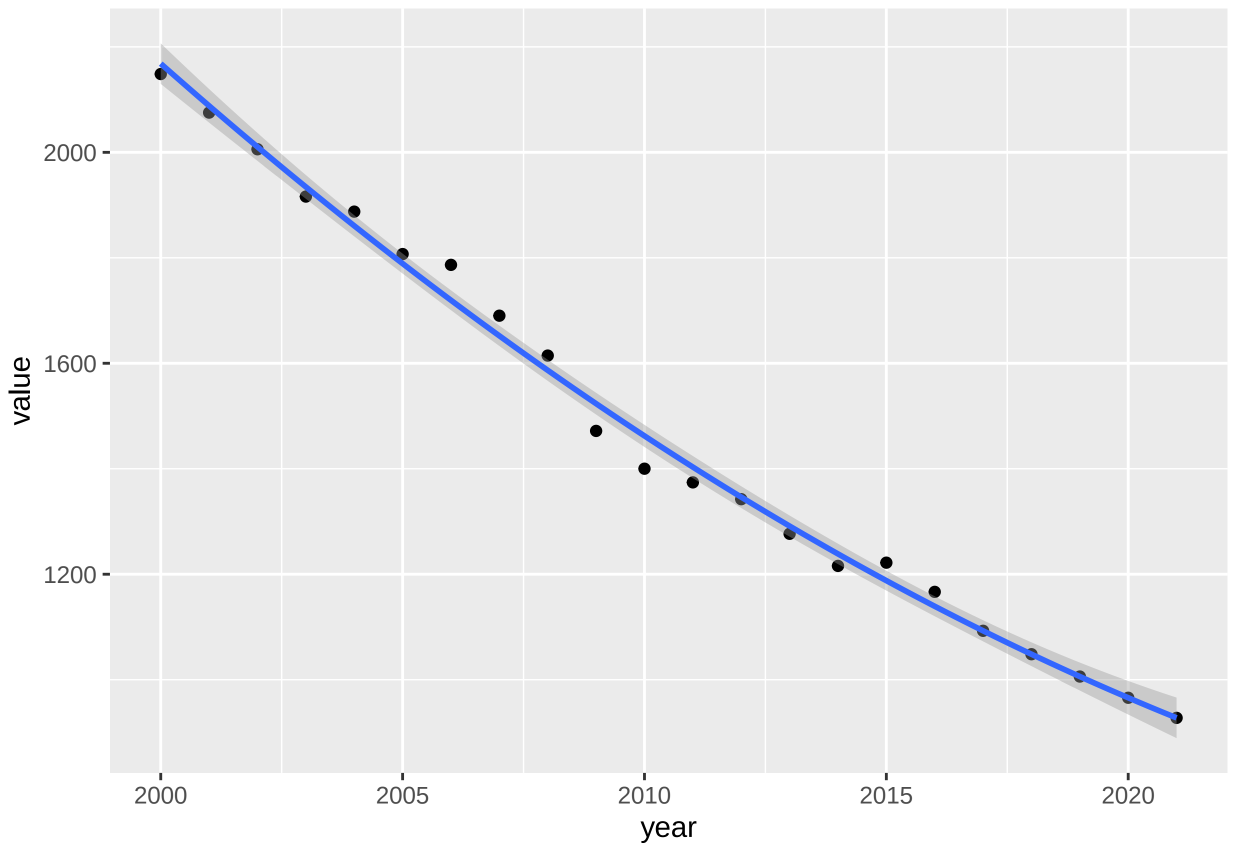

40
30
20
10
0
−10
value
2000
2005
2010
2015
2020
year


0.100
0.075
0.050
0.025
0.000
value
2000
2005
2010
2015
2020
year


80
60
40
20
value
2000
2005
2010
2015
2020
year


value
75
50
25
0
2000
2005
2010
2015
2020
year


value
0.45
0.40
0.35
2000
2005
2010
2015
2020
year



Error dtaframe generation

Once models are done, we have to measure their error and compare them with each other.

```
#Generating dataframe of all MSEs
Error<-cbind(arimaErrors,nnErrors,hybErrors,mlpErrors,expErrors,smaErrors,polErrors)%>%t()%>%data.frame
names(Error)<-Products%>%names()
Error
```

```
##              Chewing Tobacco  in  Pounds Cigarette Removals  in  Cigarettes
## arimaErrors                4.459685e-04                            31123.88
## nnErrors                   5.406670e-04                            38139.10
## hybErrors                  6.235240e-04                            37894.86
## mlpErrors                  5.599046e-04                            32172.75
## expErrors                  1.104295e-05                             4183.81
## smaErrors                  2.631846e-04                            20609.56
## polErrors                  2.608252e+03                           669331.08
##              Total Cigars  in  Cigars
## arimaErrors                26.683193
## nnErrors                    9.267640
## hybErrors                  19.958148
## mlpErrors                  25.624447
## expErrors                  47.865150
## smaErrors                   1.145378
## polErrors                 7.193974
##             Total Loose Tobacco  in  Cigarette Equivalents
## arimaErrors                                       283.0112
## nnErrors                                          254.8634
## hybErrors                                         270.9201
## mlpErrors                                         271.3176
## expErrors                                         152.6131
## smaErrors                                         193.2121
## polErrors                                         444.6121
##             Total Loose Tobacco  in  Pounds Small Cigars  in  Cigars
## arimaErrors                    1.168262e-03                2.3680145
## nnErrors                       1.051432e-03                1.2794276
## hybErrors                      1.118510e-03                2.1350519
## mlpErrors                      1.110140e-03                1.4467371
## expErrors                      6.299508e-04                0.4830845
## smaErrors                      7.975088e-04                0.6773153
## polErrors                      2.599955e+03             2502.1847513
##             Pipe Tobacco  in  Pounds
## arimaErrors             1.071333e-03
## nnErrors                6.213423e-04
## hybErrors               7.907405e-04
## mlpErrors               6.979961e-04
## expErrors               1.584815e-04
## smaErrors               5.684163e-04
## polErrors               2.602271e+03
##             Roll-Your-Own Tobacco  in  Cigarette Equivalents
## arimaErrors                                        21.300362
## nnErrors                                           11.065435
## hybErrors                                          19.031354
## mlpErrors                                          12.469192
## expErrors                                           4.605880
## smaErrors                                           6.544837
## polErrors                                         2238.941136
##             Roll-Your-Own Tobacco  in  Pounds Large Cigars  in  Cigars
## arimaErrors                      8.788480e-05                 8.625730
## nnErrors                         4.565589e-05                 1.416002
## hybErrors                        7.852407e-05                 7.812477
## mlpErrors                        4.579505e-05                 1.858485
## expErrors                        1.900375e-05                 9.049214
## smaErrors                        2.700384e-05                 2.915244
## polErrors                        2.613356e+03               148.642675
##             Pipe Tobacco  in  Cigarette Equivalents Snuff  in  Pounds
## arimaErrors                               259.65569      4.509091e-04
## nnErrors                                  107.63581      3.358415e-04
## hybErrors                                 209.33749      3.624968e-04
## mlpErrors                                 169.08301      2.720751e-04
## expErrors                                  38.41066      3.551449e-03
## smaErrors                                 137.76529      2.967051e-04
## polErrors                                 339.52965      2.566246e+03
##             Total Combustible Tobacco  in  Cigarette Equivalents
## arimaErrors                                            38702.251
## nnErrors                                               45264.742
## hybErrors                                              45394.039
## mlpErrors                                              39066.504
## expErrors                                                  1926.924
## smaErrors                                                 24101.664
## polErrors                                                956865.484
```

Getting best model for every type of tobacco

```
for(i in c(1:13)){
  print(rownames(Error)[which.min(Error[,i])])
}
```

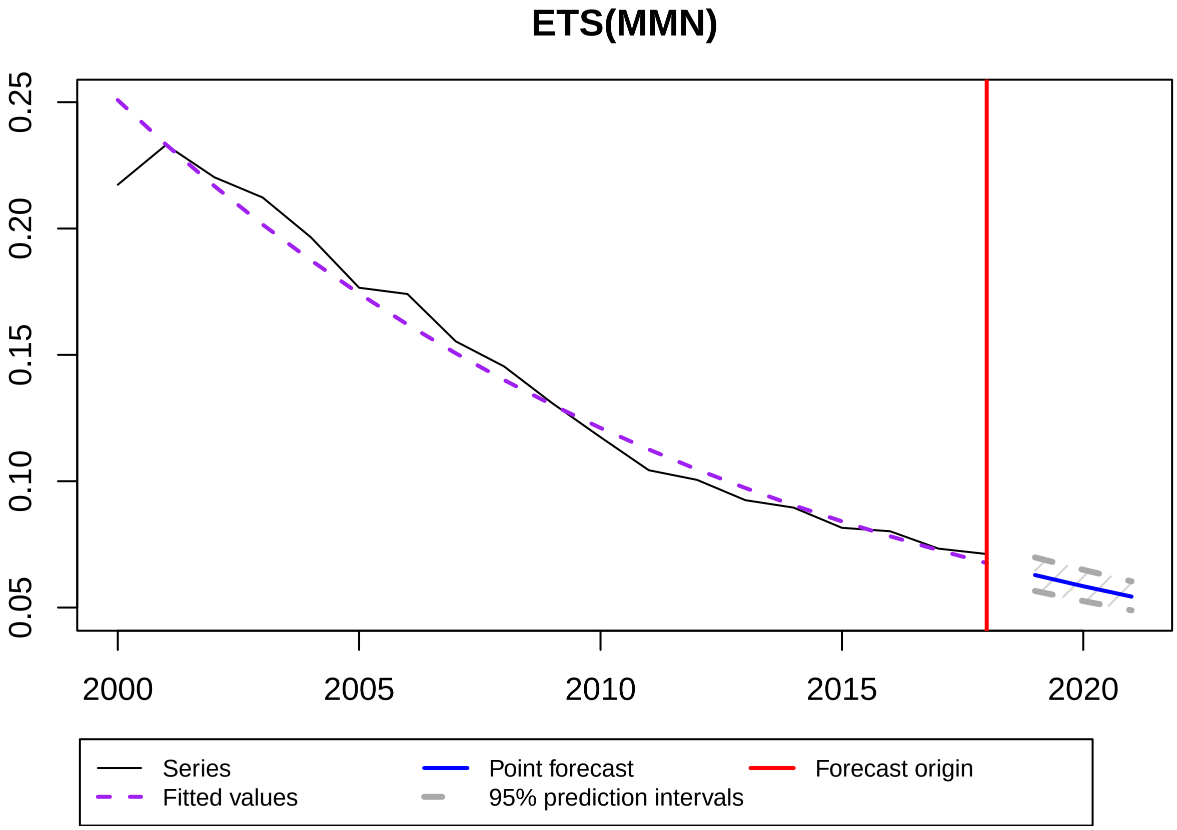
```
## [1] "expErrors"
## [1] "expErrors"
## [1] "smaErrors"
## [1] "expErrors"
## [1] "expErrors"
## [1] "expErrors"
## [1] "expErrors"
## [1] "expErrors"
## [1] "expErrors"
## [1] "nnErrors"
## [1] "expErrors"
## [1] "mlpErrors"
## [1] "expErrors"
```

2021 Final predictions per capita with 10 training set

```
#Generating new training set
trainTotals10<-list()
testTotals10<-list()
for(i in c(1:13)){
  trainTotals10<-c(trainTotals10, list(ts(head(totalsPerCapita[[i]],19),start=c(2000),end=c(2018),freque
  testTotals10<-c(testTotals10, list(ts(tail(totalsPerCapita[[i]],2),start=c(2019),end=c(2020),frequency
}

#Creating array to store the 2021 forecasts and errors
forecast2021<-c()
predictionMSE<-c()
```

Chewing Tobacco in Pounds forecast

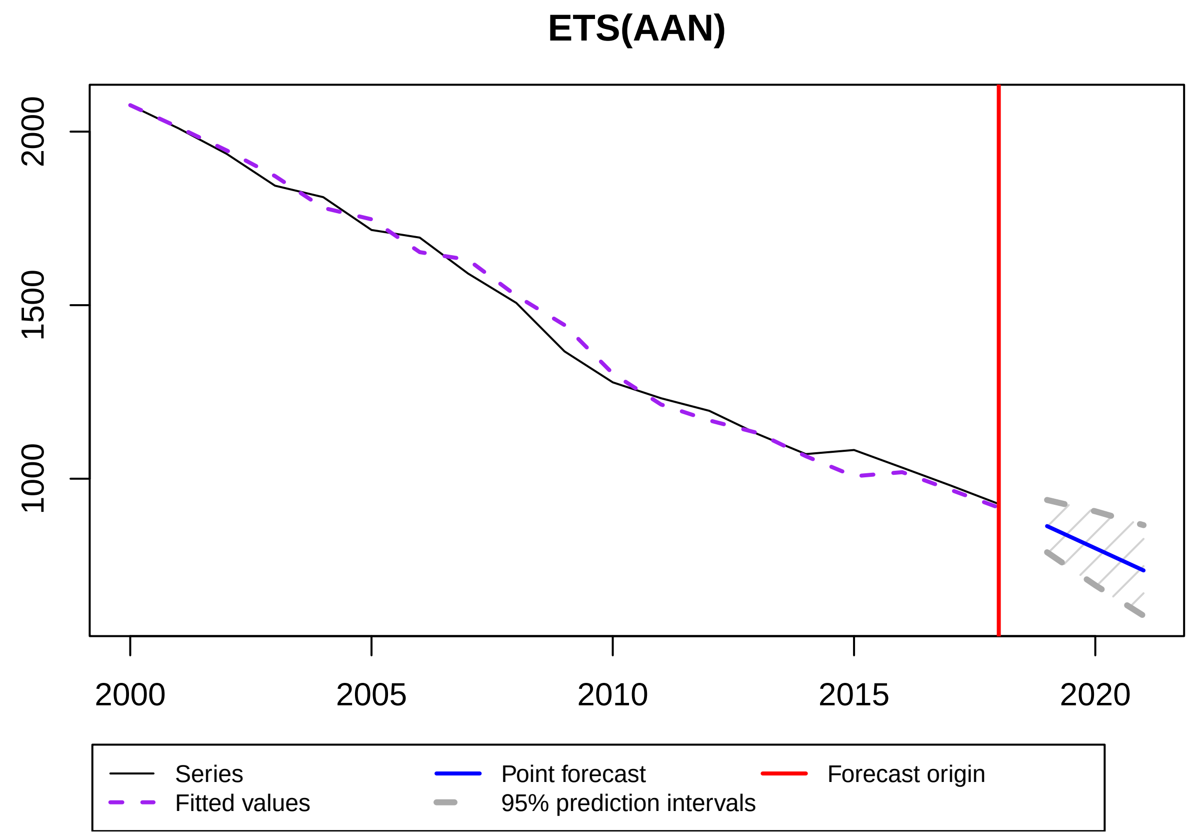
```
 #Generating adn plotting model
  exp_model<-es(trainTotals10[[1]], h=3, holdout=FALSE, interval=TRUE, silent='output')
```



```
 #Getting MSE (the head and tail are used to get from 2019-2020)
 prediction<-exp_model$forecast%>%as.numeric()%>%tail(3)%>%head(2)
 test<-testTotals10[[1]]%>%as.numeric()

 ##Saving MSE and forecast
 predictionMSE<-c(predictionMSE,MSE(prediction,test))
 forecast2021<-c(forecast2021,exp_model$forecast%>%as.numeric()%>%tail(1))
```

Cigarrates Removal in Cigarrates forecast

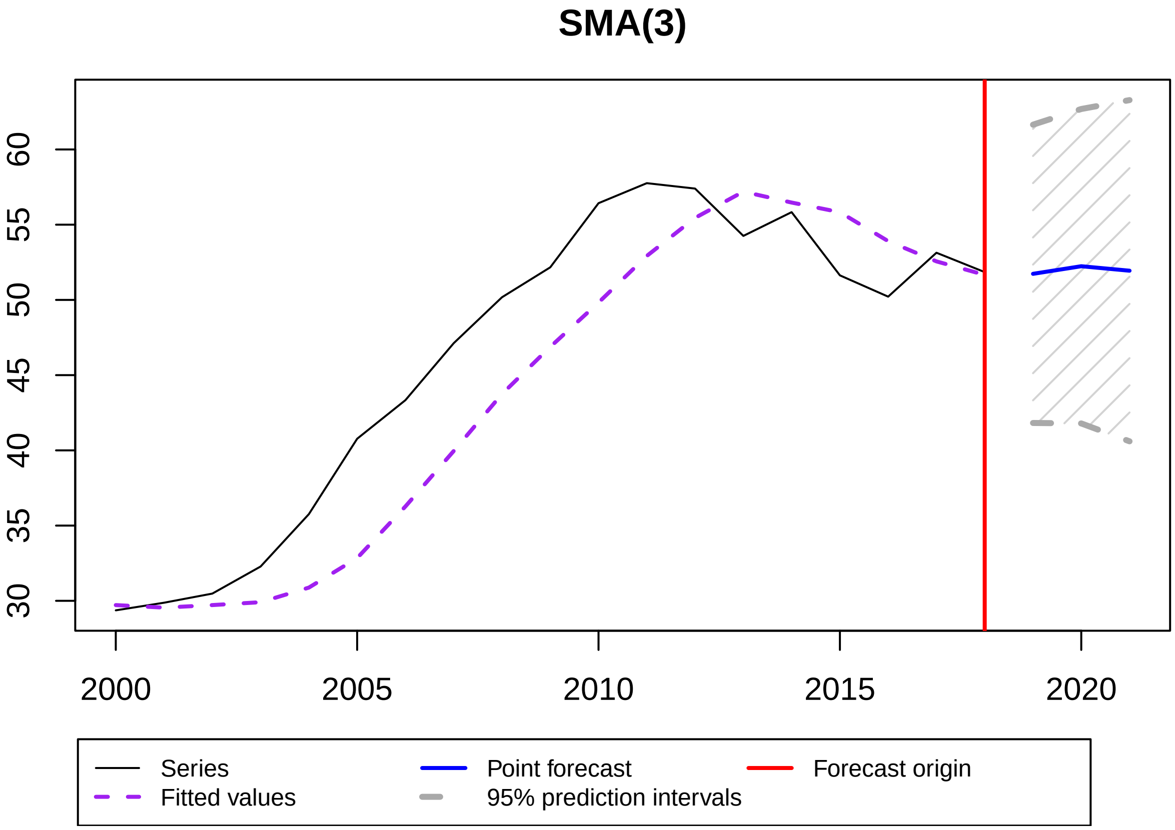
```
#Generating adn plotting model
 exp_model<-es(trainTotals10[[2]], h=3, holdout=FALSE, interval=TRUE, silent='output')
```



```
#Getting MSE (the head and tail are used to get from 2019-2020)
prediction<-exp_model$forecast%>%as.numeric()%>%tail(3)%>%head(2)
test<-testTotals10[[2]]%>%as.numeric()

##Saving MSE and forecast
predictionMSE<-c(predictionMSE,MSE(prediction,test))
forecast2021<-c(forecast2021,exp_model$forecast%>%as.numeric()%>%tail(1))
```

Total Cigars in Cigars Forecast

```
#Generating and plotting model
sma_model<-sma(trainTotals10[[3]], h=3, order=3, holdout=FALSE, interval=TRUE, silent='output')
```

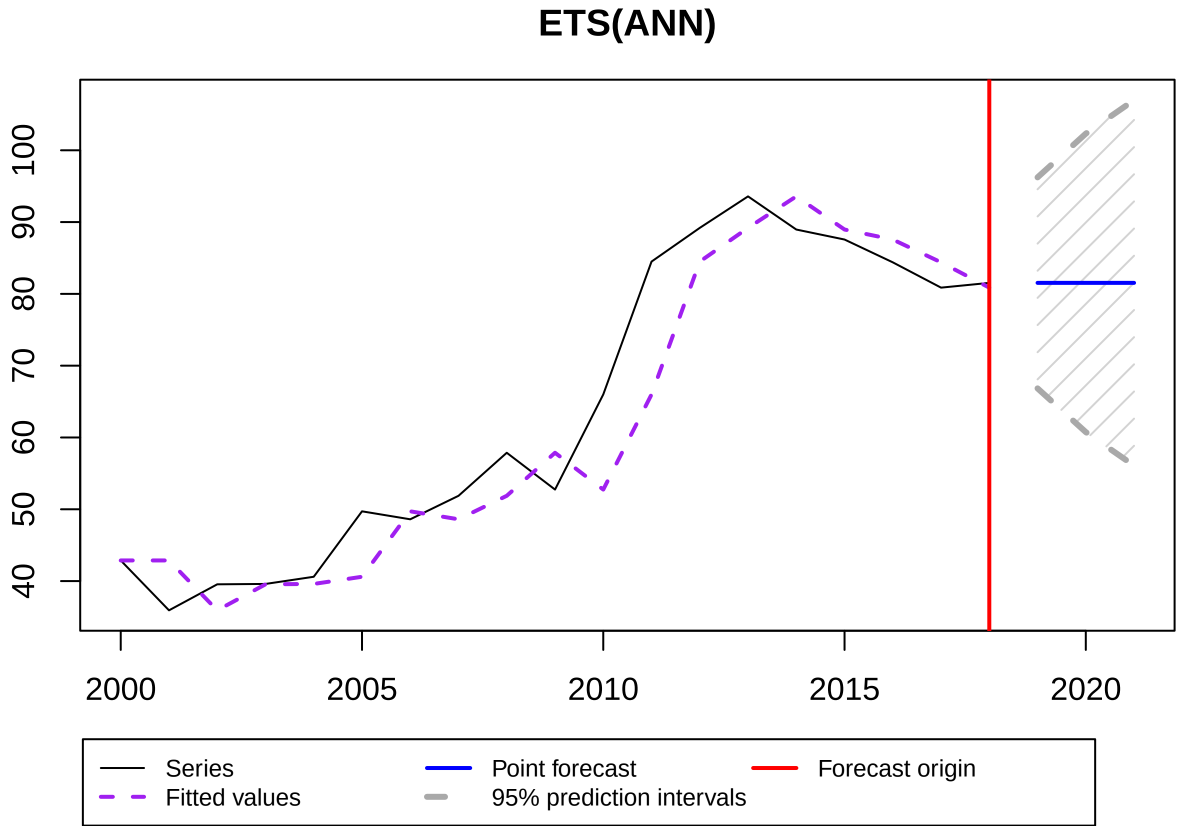


```
#Getting MSE (the head and tail are used to get from 2017-2020)
prediction<-sma_model$forecast%>%as.numeric()%>%tail(3)%>%head(1)
test<-testTotals10[[3]]%>%as.numeric()

#Saving MSE and forecast
predictionMSE<-c(predictionMSE,MSE(prediction,test))
forecast2021<-c(forecast2021,sma_model$forecast%>%as.numeric()%>%tail(1))
```

Total loose Tobacco in Cigarrates equivalent forecast

The data wont change because we move between units so if we do the same with Total loose Tobacco in Pounds it will show the same graph and prediction but scaled.

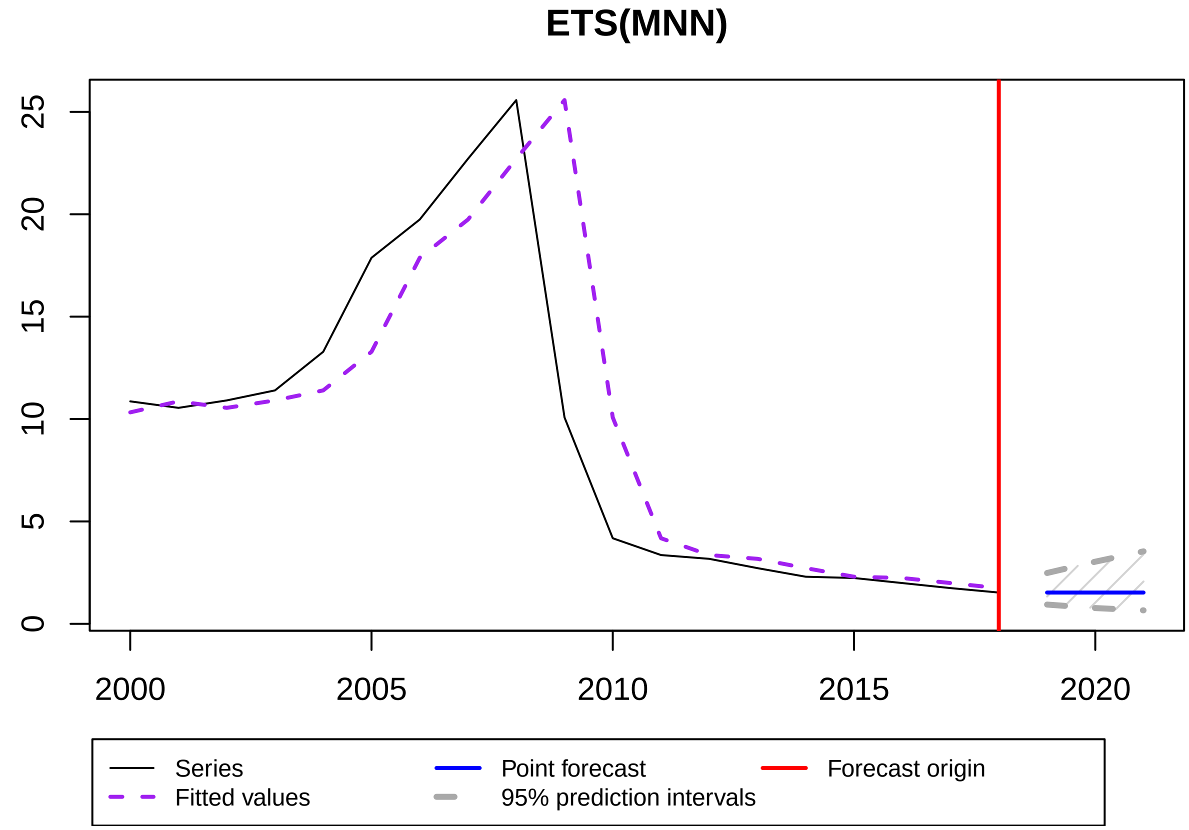
```
#Generating adn plotting model
exp_model<-es(trainTotals10[[4]], h=3, holdout=FALSE, interval=TRUE, silent='output')
```



```
#Getting MSE (the head and tail are used to get from 2019-2020)
prediction<-exp_model$forecast%>%as.numeric()%>%tail(3)%>%head(2)
test<-testTotals10[[4]]%>%as.numeric()

##Saving MSE and forecast
predictionMSE<-c(predictionMSE,MSE(prediction,test))
forecast2021<-c(forecast2021,exp_model$forecast%>%as.numeric()%>%tail(1))
```

Small Cigars in Cigars forecast

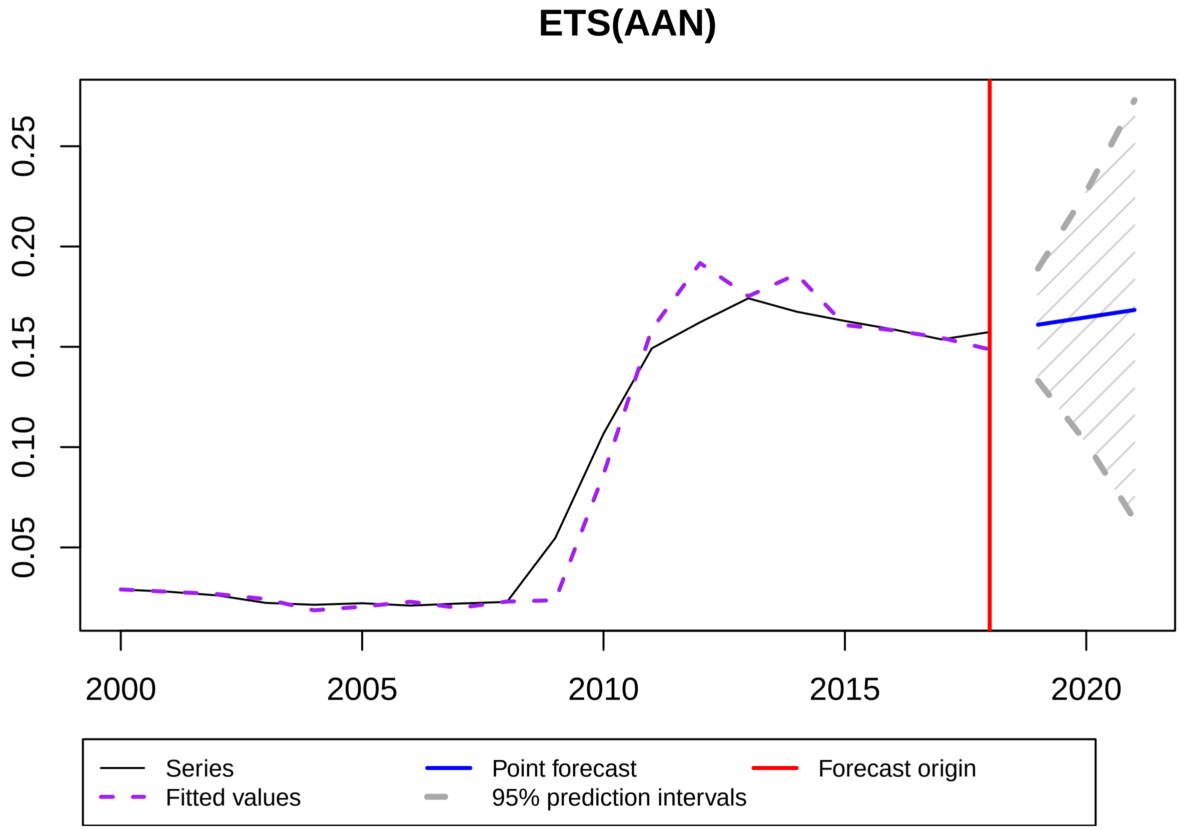
```
#Generating adn plotting model
exp_model<-es(trainTotals10[[6]], h=3, holdout=FALSE, interval=TRUE, silent='output')
```



```
#Getting MSE (the head and tail are used to get from 2019-2020)
prediction<-exp_model$forecast%>%as.numeric()%>%tail(3)%>%head(2)
test<-testTotals10[[6]]%>%as.numeric()

##Saving MSE and forecast
predictionMSE<-c(predictionMSE,MSE(prediction,test))
forecast2021<-c(forecast2021,exp_model$forecast%>%as.numeric()%>%tail(1))
```

Pippe Tobacco in Pounds forecast

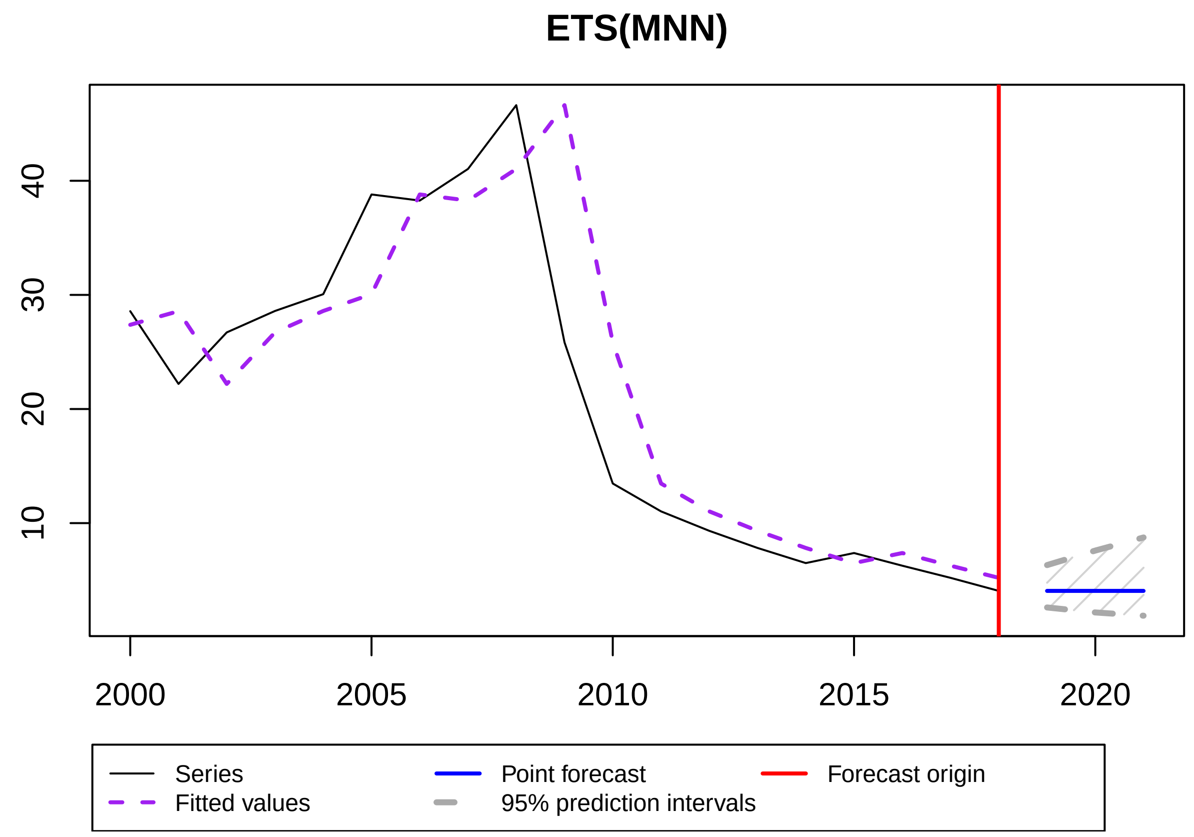
```
#Generating adn plotting model
exp_model<-es(trainTotals10[[7]], h=3, holdout=FALSE, interval=TRUE, silent='output')
```



```
#Getting MSE (the head and tail are used to get from 2019-2020)
prediction<-exp_model$forecast%>%as.numeric()%>%tail(3)%>%head(2)
test<-testTotals10[[7]]%>%as.numeric()

##Saving MSE and forecast
predictionMSE<-c(predictionMSE,MSE(prediction,test))
forecast2021<-c(forecast2021,exp_model$forecast%>%as.numeric()%>%tail(1))
```

Roll-your-own Tobacco in cigarrate equivalents forecast

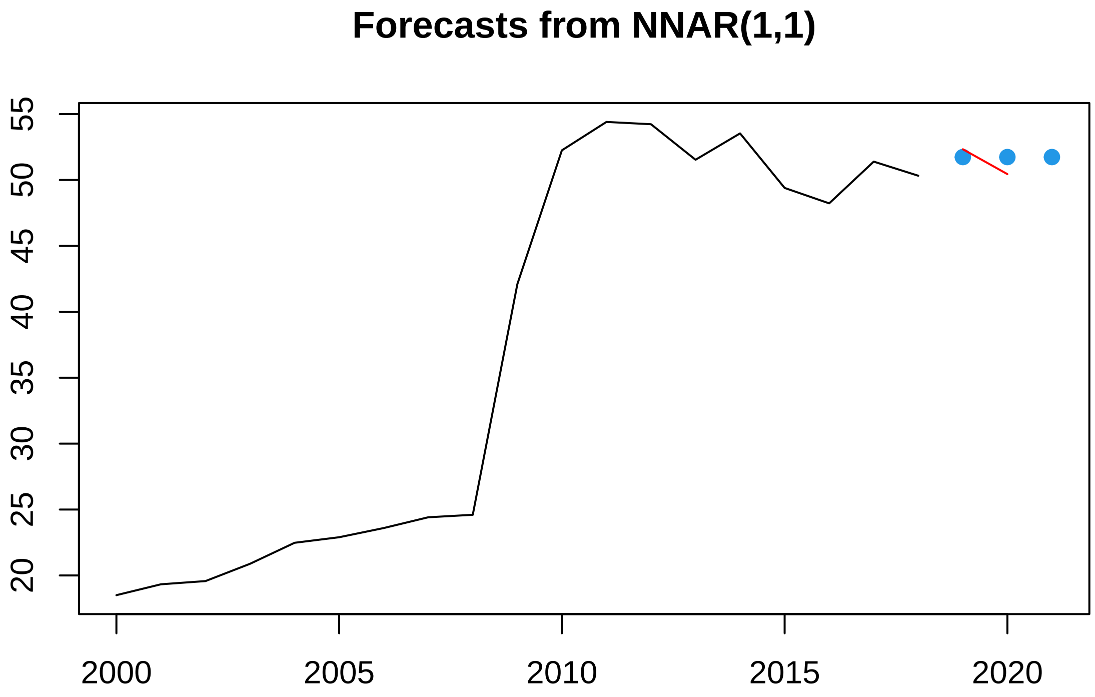
```
#Generating adn plotting model
exp_model<-es(trainTotals10[[8]], h=3, holdout=FALSE, interval=TRUE, silent='output')
```



```
#Getting MSE (the head and tail are used to get from 2019-2020)
prediction<-exp_model$forecast%>%as.numeric()%>%tail(3)%>%head(2)
test<-testTotals10[[8]]%>%as.numeric()

##Saving MSE and forecast
predictionMSE<-c(predictionMSE,MSE(prediction,test))
forecast2021<-c(forecast2021,exp_model$forecast%>%as.numeric()%>%tail(1))
```

Large Cigars in Cigars forecast

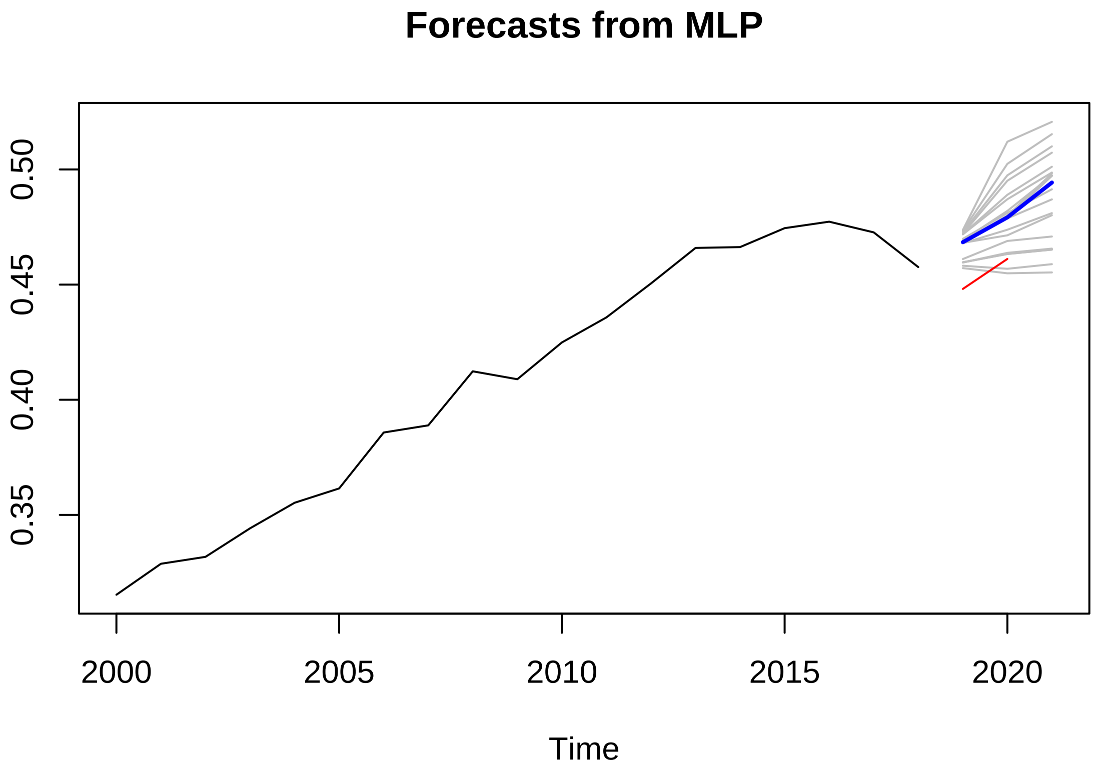
```
library(forecast)
library(forecastHybrid)
library(fpp2)
library(nnfor)
#Training model
fit<-nnetar(trainTotals10[[10]],lambda='auto')
nn_model<-forecast::forecast(fit,h=3)

#Plotting prediction and testing data (red for testing data)
plot(nn_model)
lines(testTotals10[[10]],col='red')
```



```
#Getting MSE (the head and tail are used to get from 2017-2020)
prediction<-nn_model$fitted%>%as.numeric()%>%tail(3)%>%head(2)
test<-testTotals10[[10]]%>%as.numeric()

#Saving MSE and forecast
predictionMSE<-c(predictionMSE,MSE(prediction,test))
forecast2021<-c(forecast2021,nn_model$fittedt%>%as.numeric()%>%tail(1))
```

Snuff in Pounds forecast

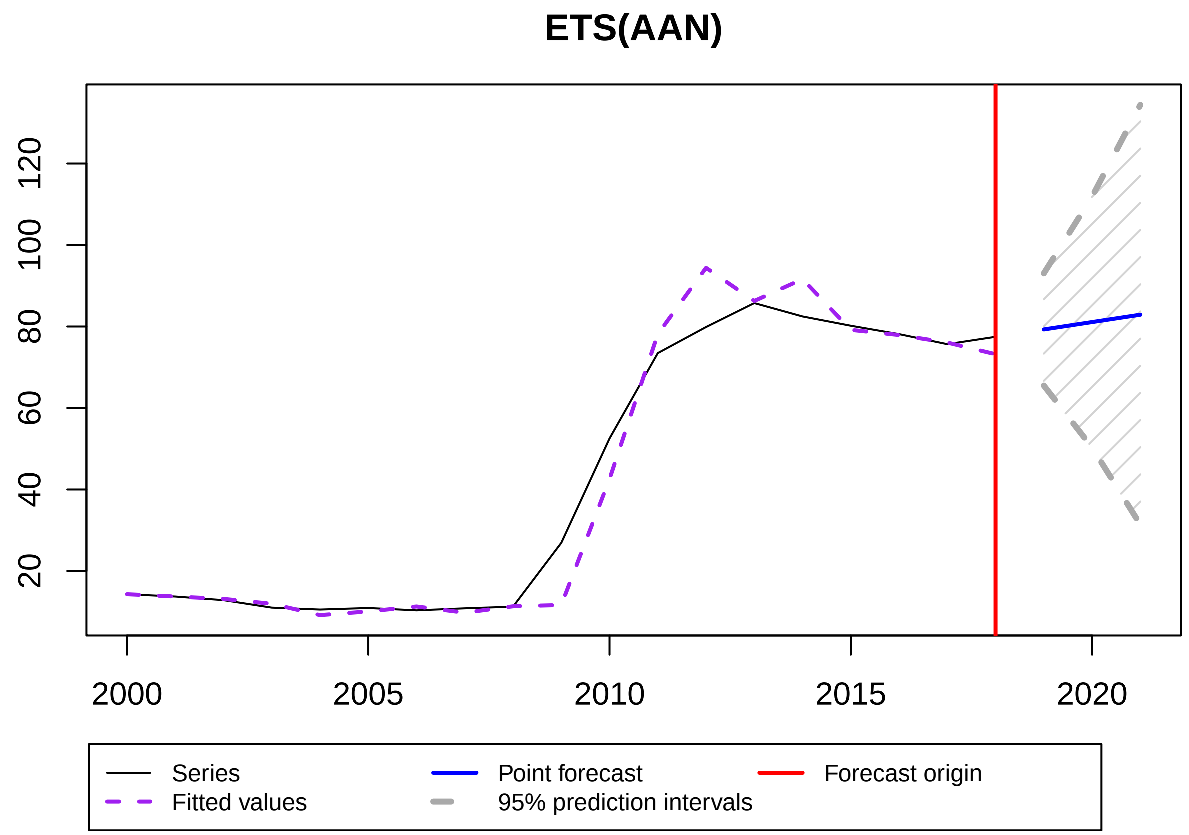
```
#Training
mlp_fit<-mlp(trainTotals10[[12]])
mlp_model<-forecast::forecast(mlp_fit,3)

#Plotting
plot(mlp_model)
lines(testTotals10[[12]],col='red')
```



```
#Getting MSE (the head and tail are used to get from 2017-2020)
prediction<-mlp_model$fitted%>%as.numeric()%>%tail(3)%>%head(2)
test<-testTotals10[[12]]%>%as.numeric()

#Saving MSE and forecast
predictionMSE<-c(predictionMSE,MSE(prediction,test))
forecast2021<-c(forecast2021,exp_model$forecast%>%as.numeric()%>%tail(1))
```

Pipe Tobacco in cigarratte equivalents forecast

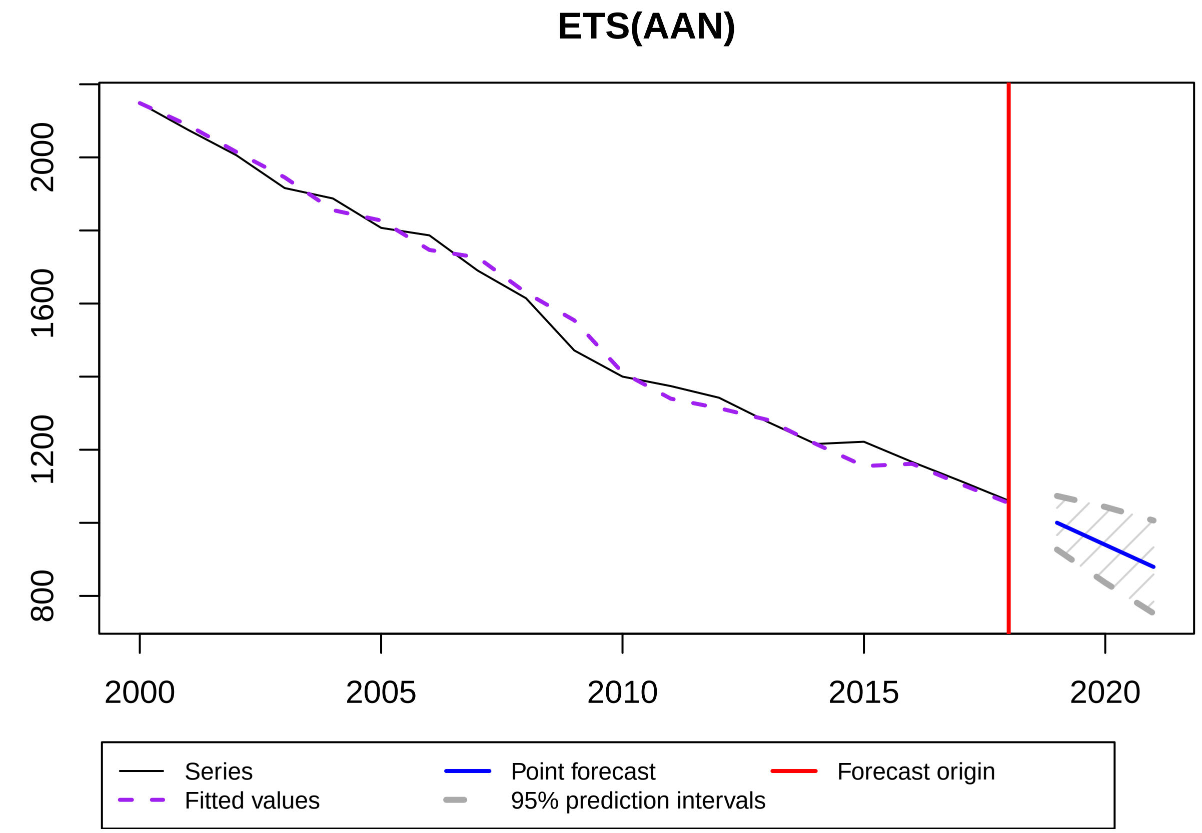
```
library(smooth)
#Generating adn plotting model
exp_model<-es(trainTotals10[[11]], h=3, holdout=FALSE, interval=TRUE, silent='output')
```



```
#Getting MSE (the head and tail are used to get from 2019-2020)
prediction<-exp_model$forecast%>%as.numeric()%>%tail(3)%>%head(2)
test<-testTotals10[[11]]%>%as.numeric()

##Saving MSE and forecast
predictionMSE<-c(predictionMSE,MSE(prediction,test))
forecast2021<-c(forecast2021,exp_model$forecast%>%as.numeric()%>%tail(1))
```

Totals by model

```
#Generating adn plotting model
exp_model<-es(trainTotals10[[13]], h=3, holdout=FALSE, interval=TRUE, silent='output')
```



```
#Getting MSE (the head and tail are used to get from 2019-2020)
prediction<-exp_model$forecast%>%as.numeric()%>%tail(3)%>%head(2)
test<-testTotals10[[13]]%>%as.numeric()

##Saving MSE and forecast
predictionMSE<-c(predictionMSE,MSE(prediction,test))
forecast2021<-c(forecast2021,exp_model$forecast%>%as.numeric()%>%tail(1))
```