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)

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`)

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.

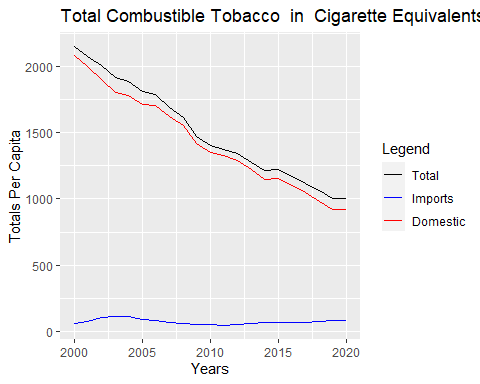
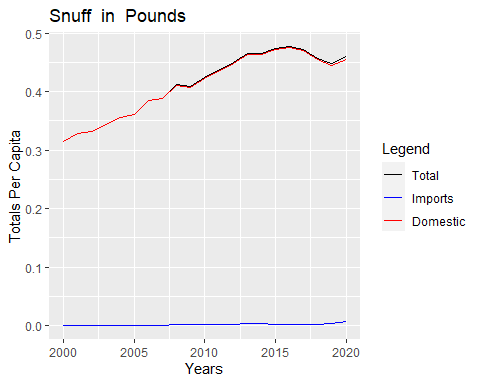
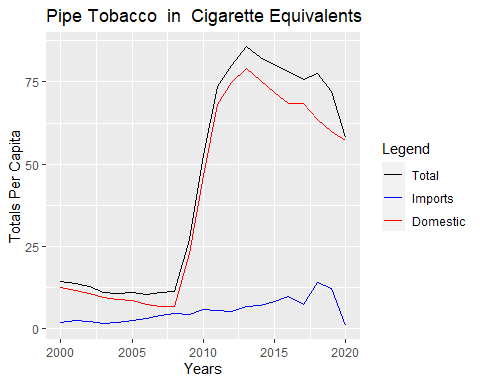
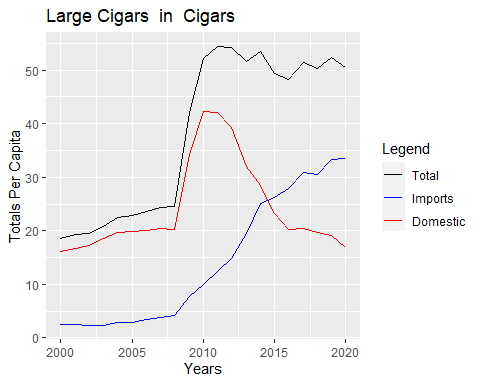
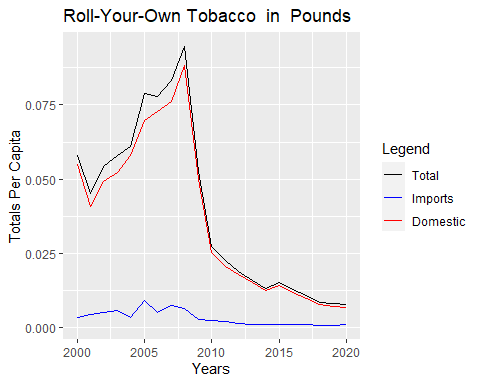
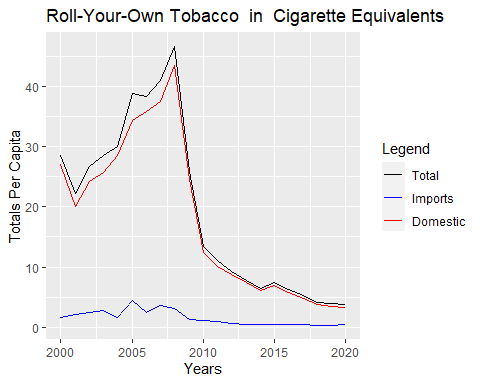
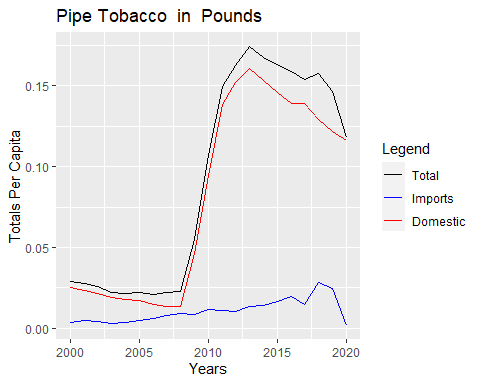
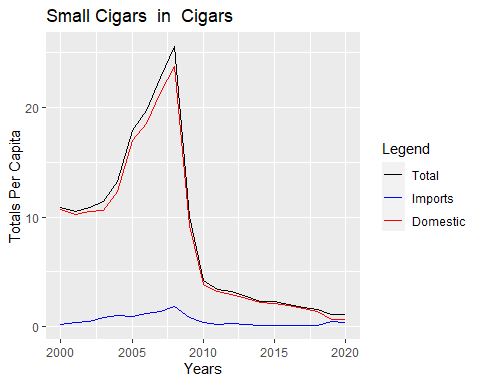
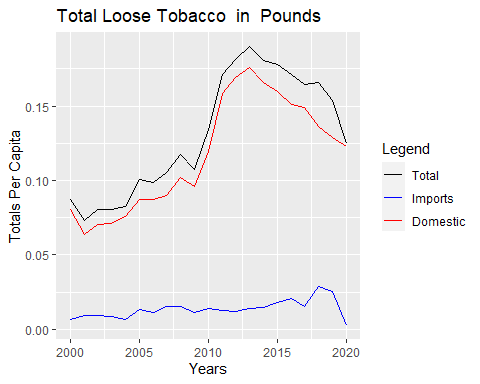
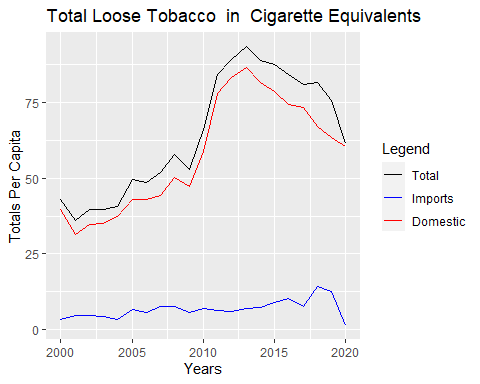
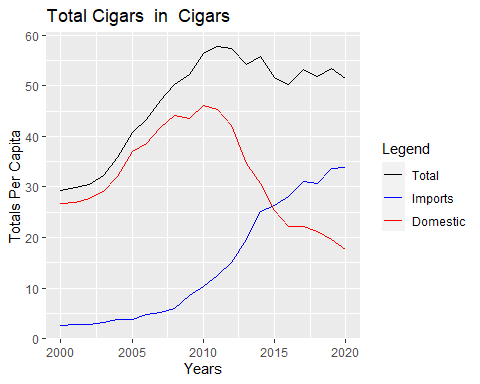
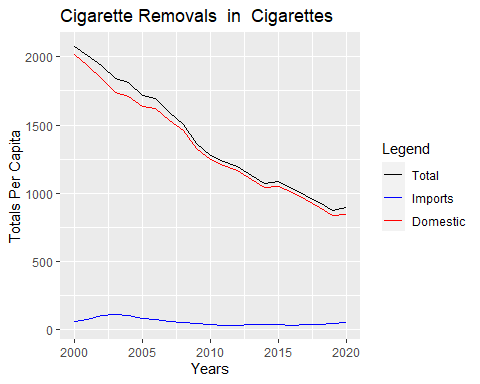
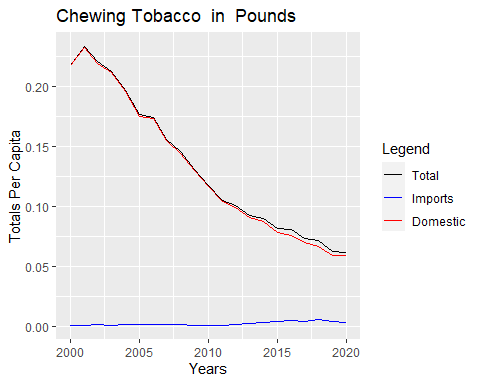
Products<-list()  
for (i in Tdata$item[1:13]){  
 Products<-c(Products,list(filter(Tdata, item==i)))  
}  
names(Products)<-Tdata$item[1:13]

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:

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')))  
}



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.

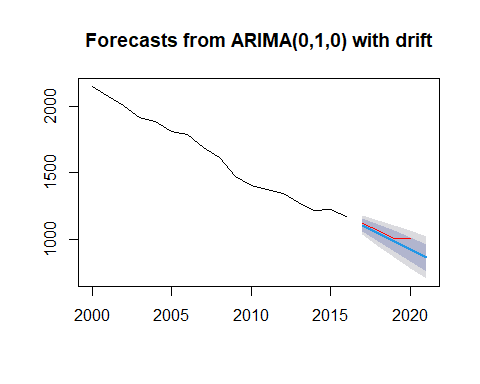
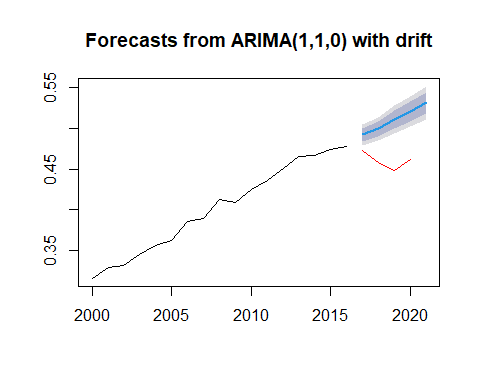
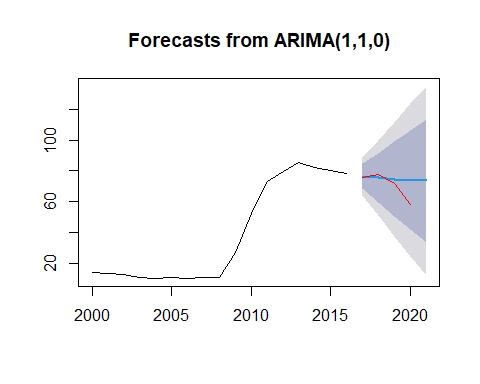
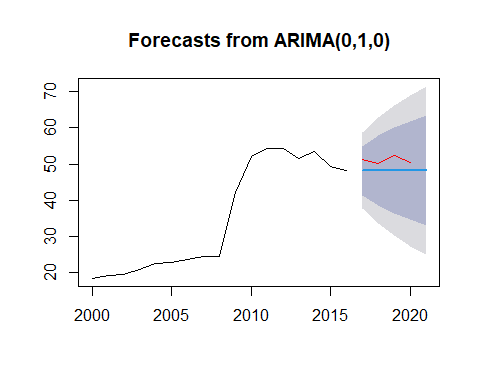
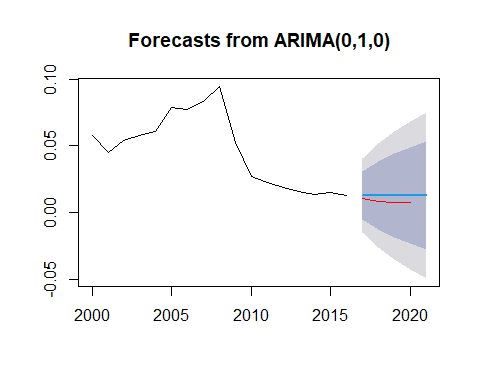
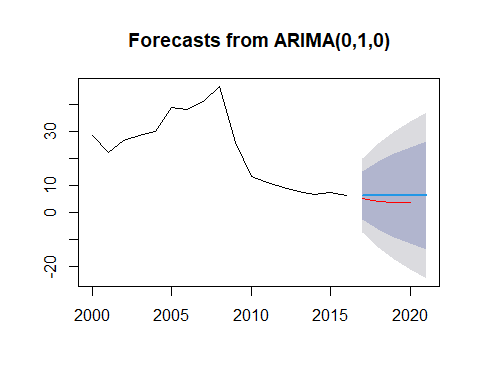
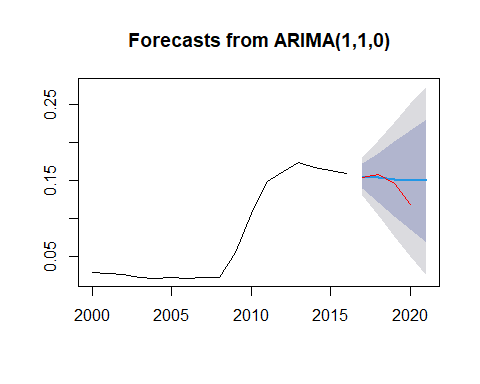
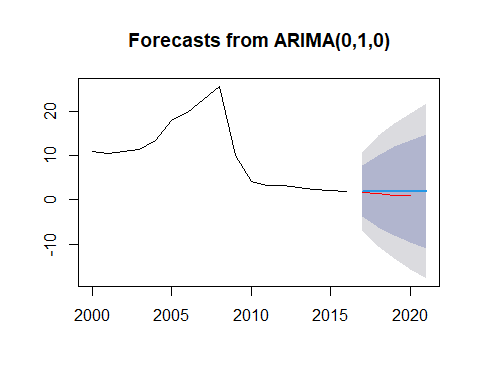
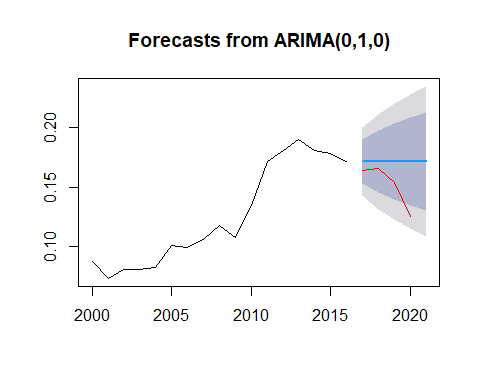
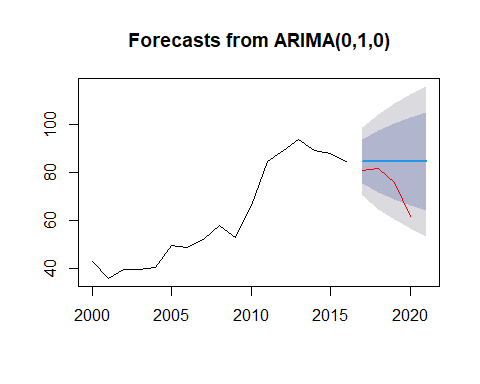
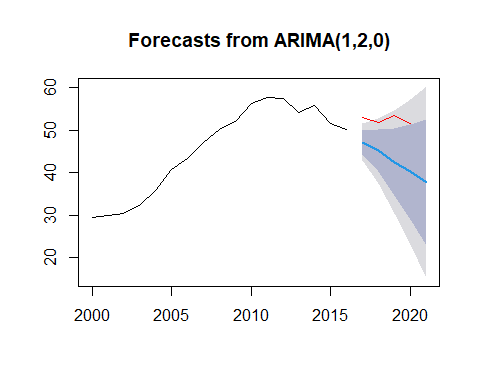
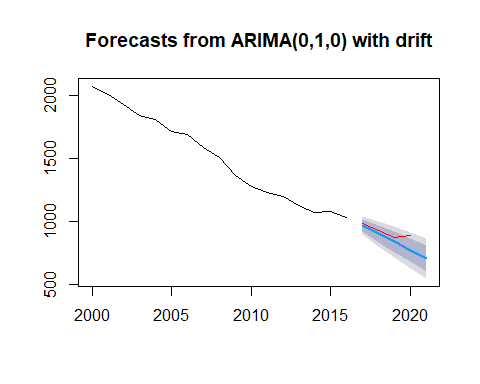
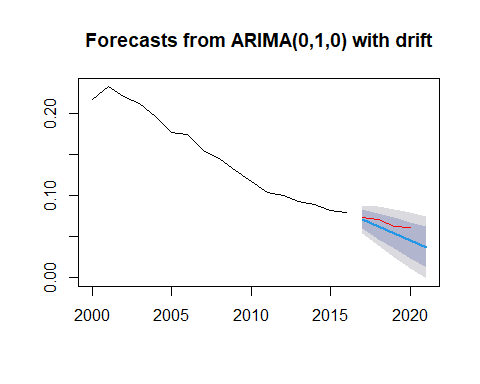
trainTotals<-list()  
testTotals<-list()  
for(i in c(1:13)){  
 trainTotals<-c(trainTotals, list(ts(head(totalsPerCapita[[i]],17),start=c(2000),end=c(2016),frequency = 1)))  
 testTotals<-c(testTotals, list(ts(tail(totalsPerCapita[[i]],4),start=c(2017),end=c(2020),frequency = 1)))  
}

Function to get the Mean Squared Error from 2 vectors

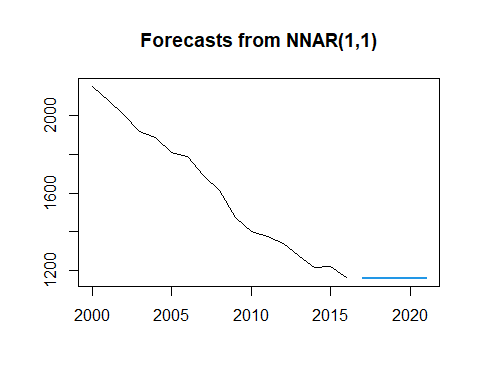
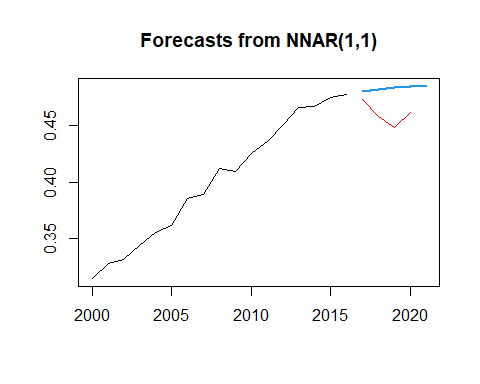
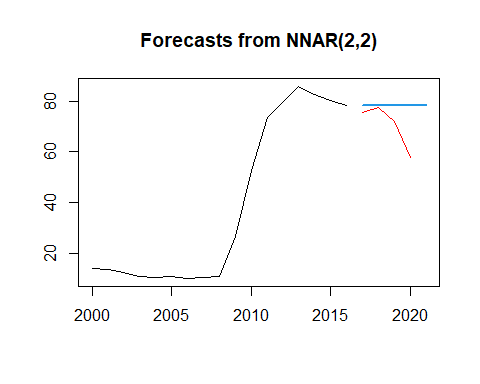
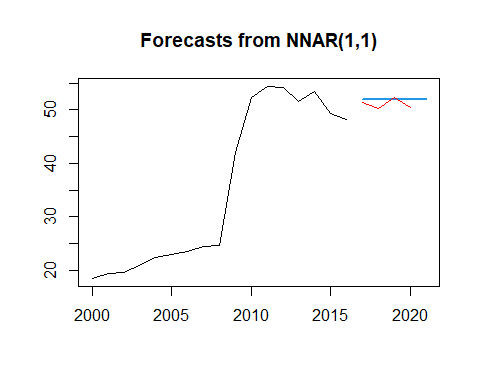
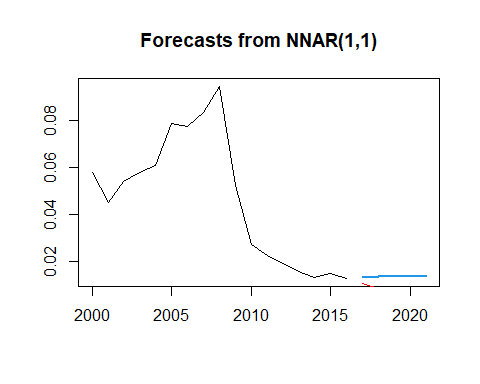
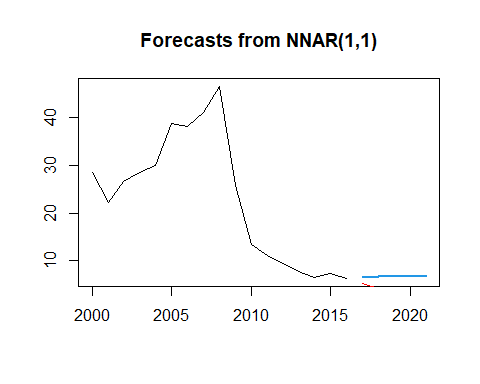
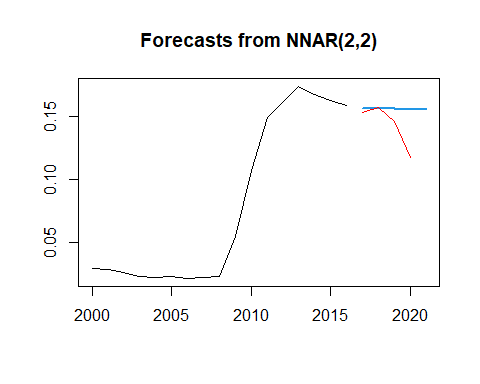
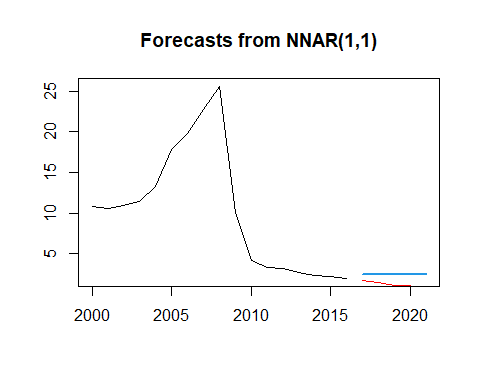
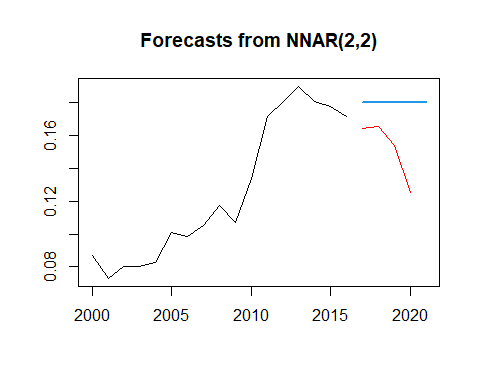
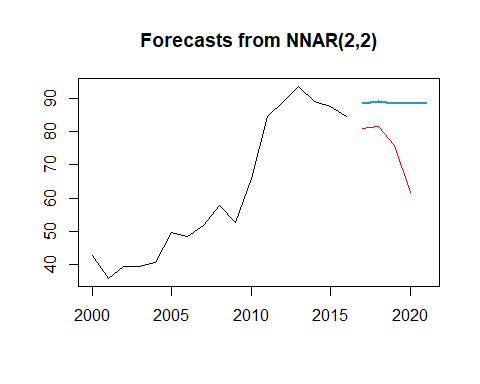
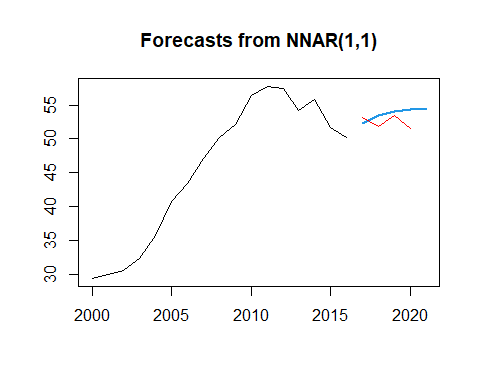
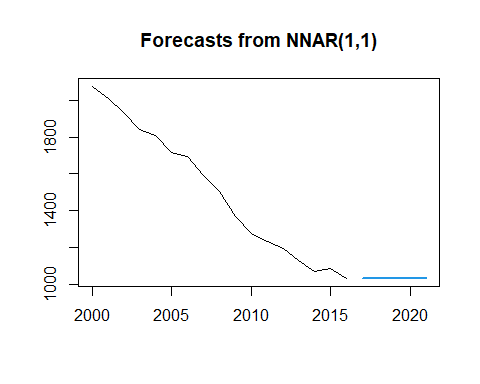
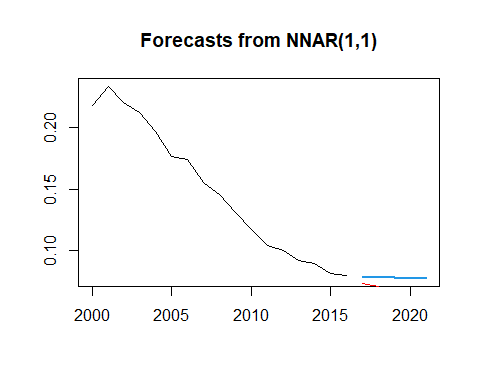
MSE<- function (v1,v2){  
 return(sum((v1-v2)^2)/length(v1))  
}

Now everything is ready to start with the models.

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))  
}



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))  
}



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.

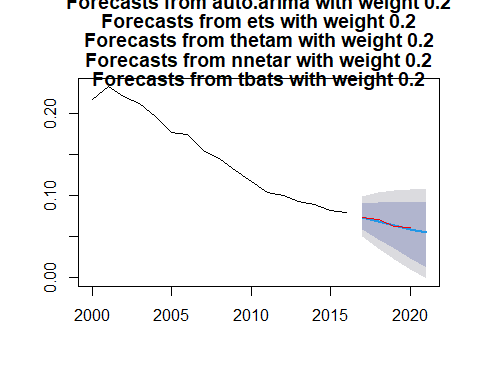
## Fitting the auto.arima model

## 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.

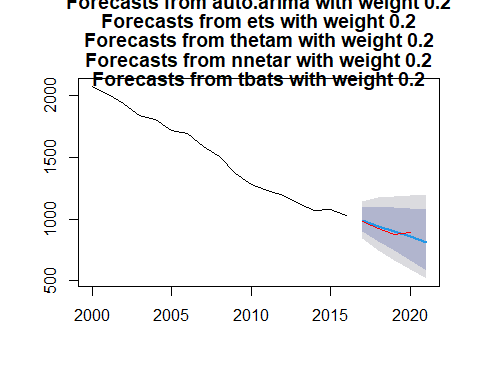
## Fitting the auto.arima model

## 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.

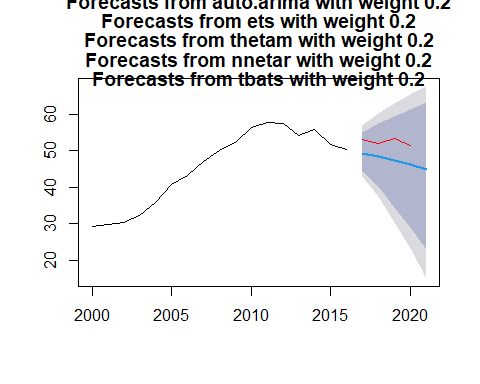
## Fitting the auto.arima model

## 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.

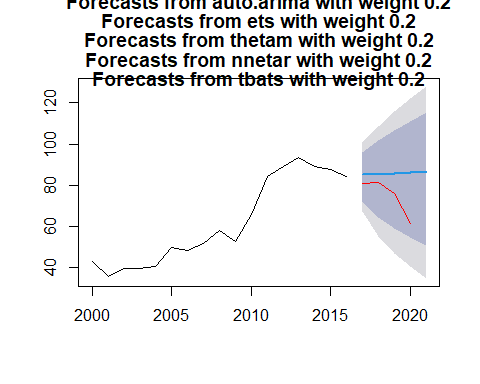
## Fitting the auto.arima model

## 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.

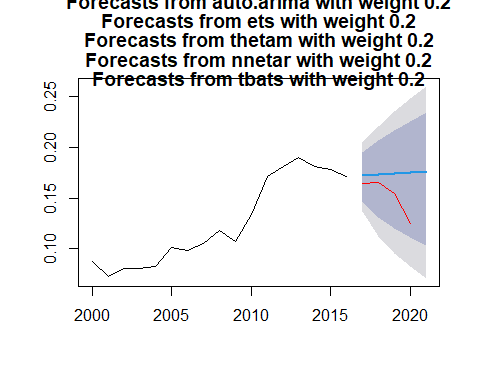
## Fitting the auto.arima model

## 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.

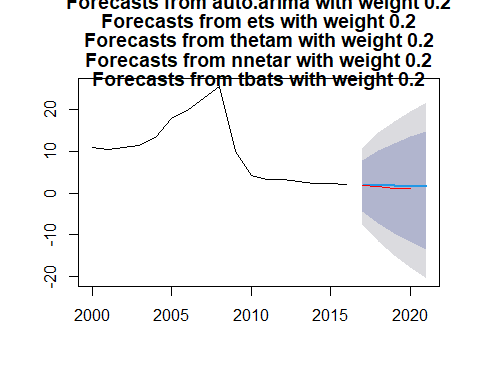
## Fitting the auto.arima model

## 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.

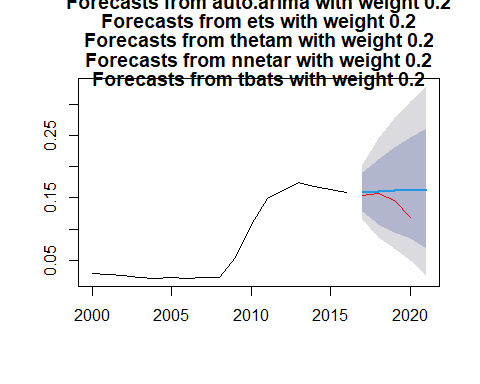
## Fitting the auto.arima model

## 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.

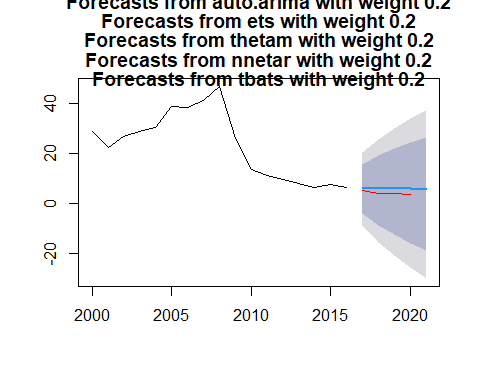
## Fitting the auto.arima model

## 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.

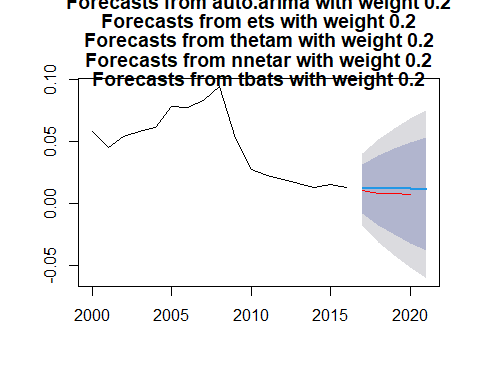
## Fitting the auto.arima model

## 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.

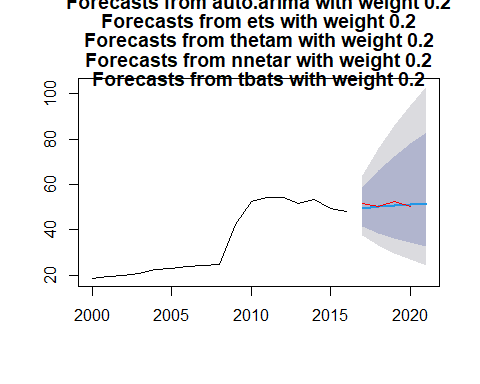
## Fitting the auto.arima model

## 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.

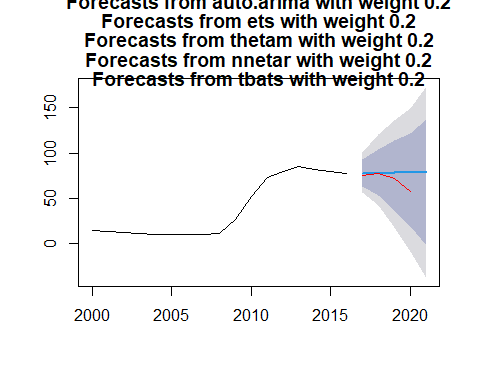
## Fitting the auto.arima model

## 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.

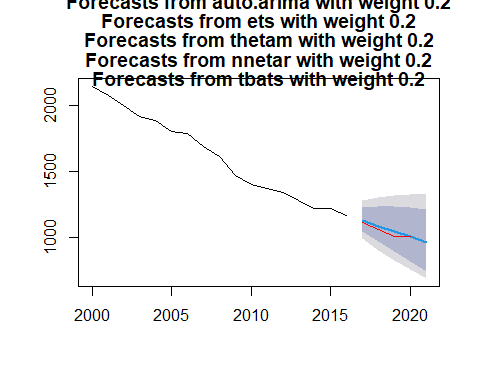
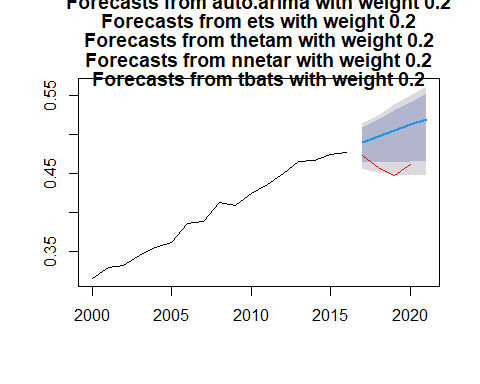
## Fitting the auto.arima model

## Fitting the ets model

## Fitting the thetam model

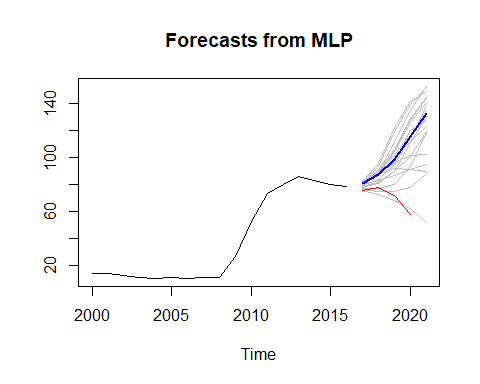
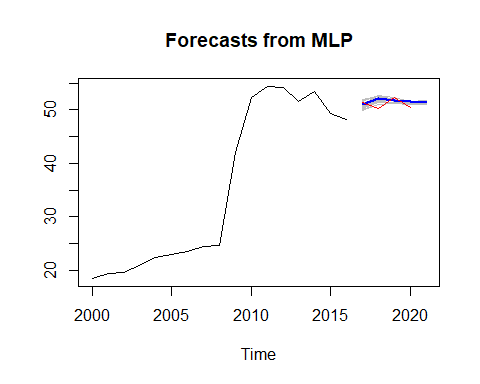
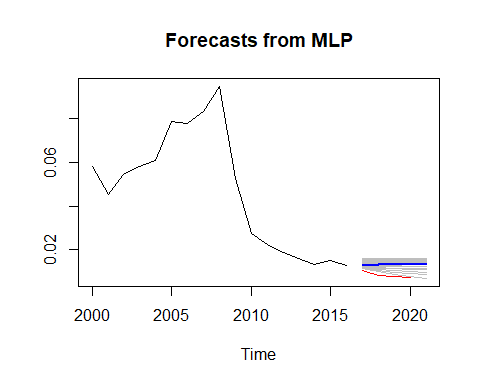
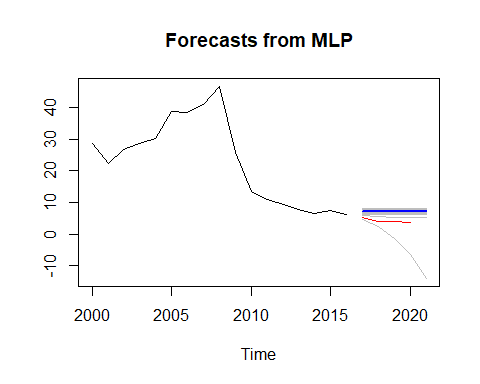
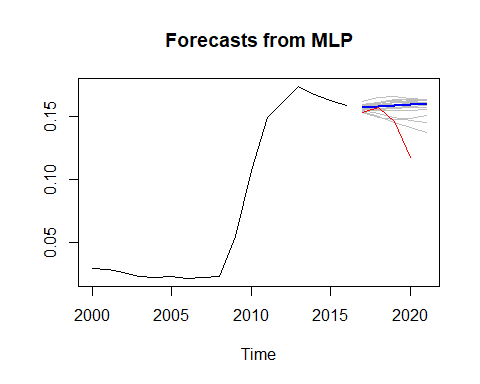
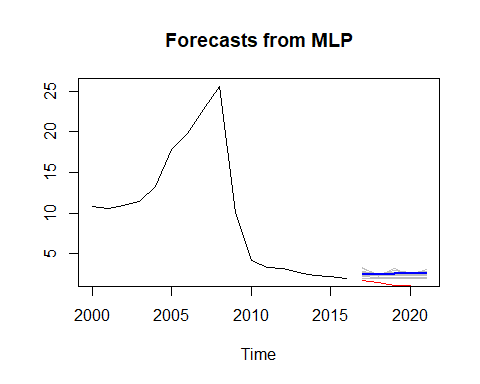
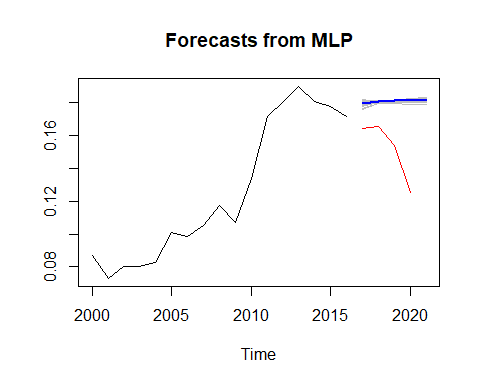
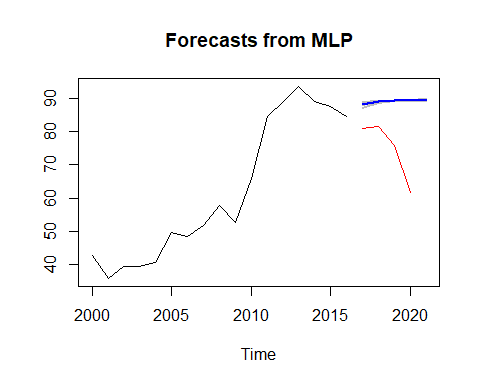
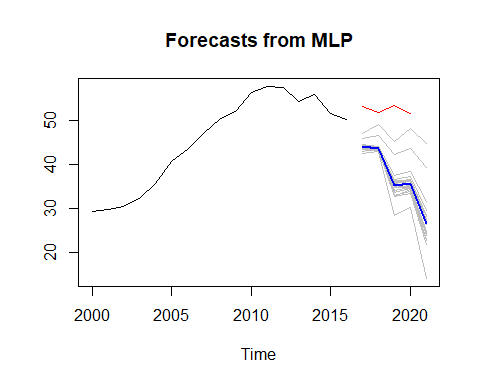
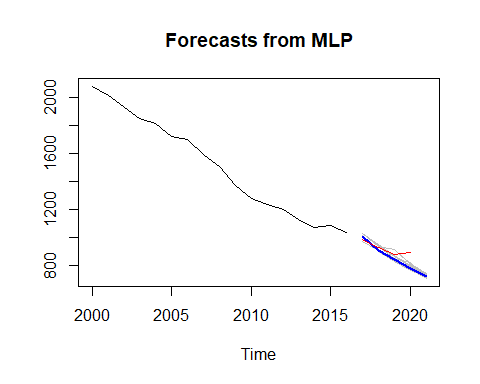
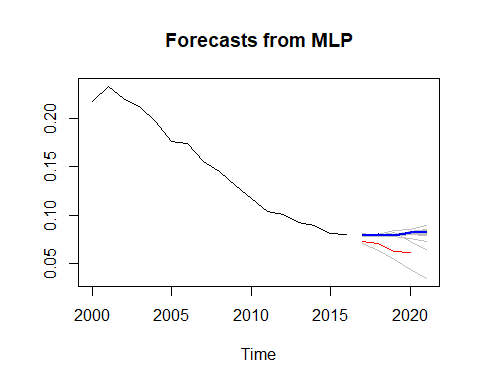
## Fitting the nnetar model

## Fitting the tbats model

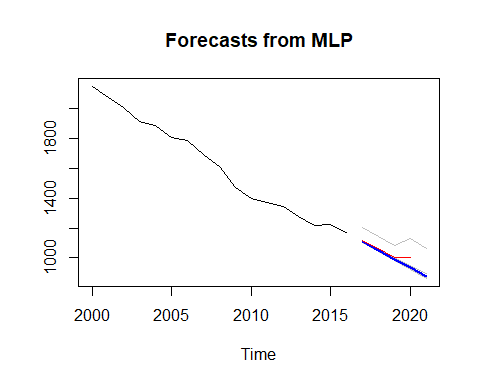
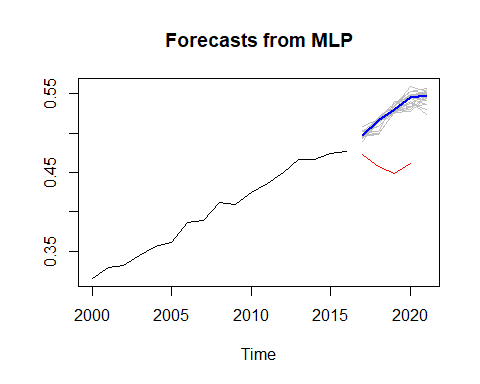


mlpErrors<-c()  
for(i in c(1:13)){  
 #Training  
 mlp\_fit<-mlp(trainTotals[[i]])  
 mlp\_model<-forecast::forecast(mlp\_fit,5)  
   
 #Plotting  
 plot(mlp\_model)  
 lines(testTotals[[i]],col='red')  
   
 #Getting MSE (the head and tail are used to get from 2017-2020)  
 prediction<-mlp\_model$fitted%>%as.numeric()%>%tail(5)%>%head(4)  
 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).



## 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).



library(smooth)

## Loading required package: greybox

## Package "greybox", v1.0.4 loaded.

##   
## Attaching package: 'greybox'

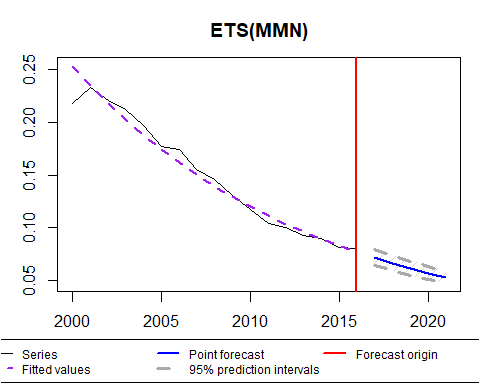
## The following object is masked \_by\_ '.GlobalEnv':  
##   
## MSE

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

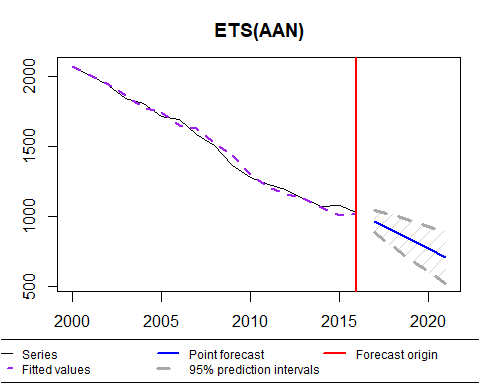
## This is package "smooth", v3.1.5

library(greybox)  
  
expErrors<-c()  
for (i in c(1:13)){  
 #Generating adn plotting model  
 exp\_model<-es(trainTotals[[i]], h=5, holdout=FALSE, interval=TRUE, silent=FALSE)  
   
 #Getting MSE (the head and tail are used to get from 2017-2020)  
 prediction<-exp\_model$forecast%>%as.numeric()%>%tail(5)%>%head(4)  
 test<-testTotals[[i]]%>%as.numeric()  
   
 ##Saving MSE in expError vector  
 expErrors<-c(expErrors,MSE(prediction,test))  
}

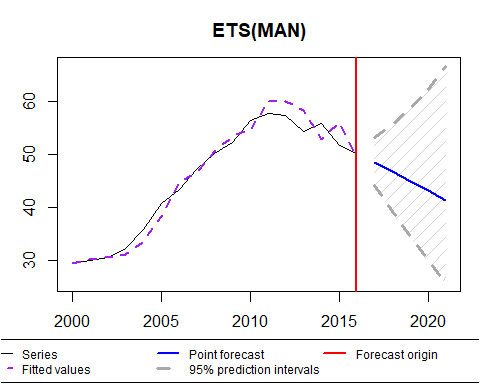
## Forming the pool of models based on... ANN, AAN, Estimation progress: 33%44%56%67%78%89%100%... Done!



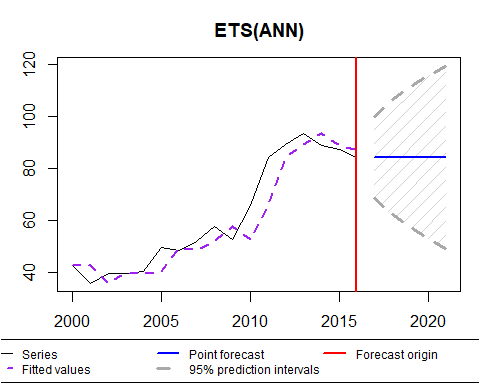
## Forming the pool of models based on... ANN, AAN, Estimation progress: 33%44%56%67%78%89%100%... Done!



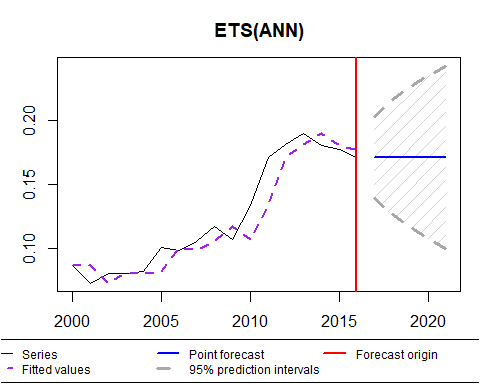
## Forming the pool of models based on... ANN, AAN, Estimation progress: 33%44%56%67%78%89%100%... Done!



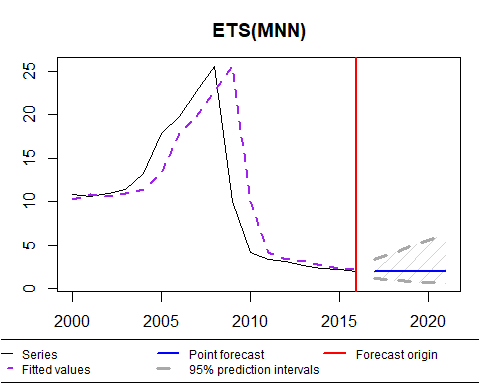
## Forming the pool of models based on... ANN, AAN, Estimation progress: 100%... Done!



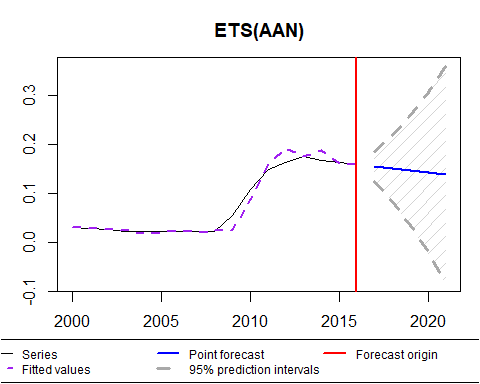
## Forming the pool of models based on... ANN, AAN, Estimation progress: 100%... Done!



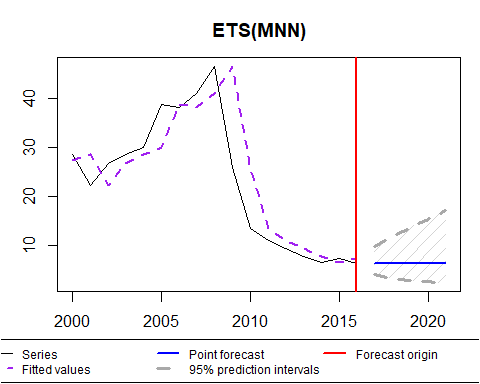
## Forming the pool of models based on... ANN, AAN, Estimation progress: 100%... Done!



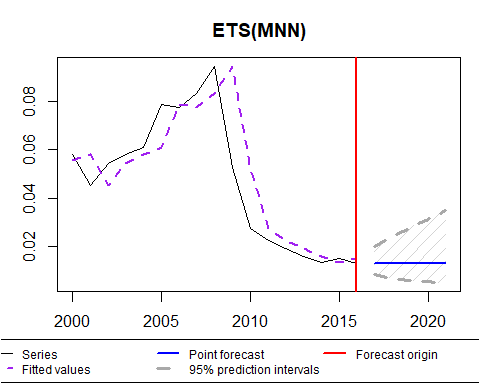
## Forming the pool of models based on... ANN, AAN, Estimation progress: 33%44%56%67%78%89%100%... Done!



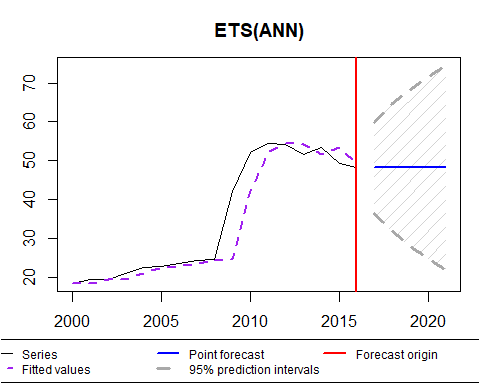
## Forming the pool of models based on... ANN, AAN, Estimation progress: 100%... Done!



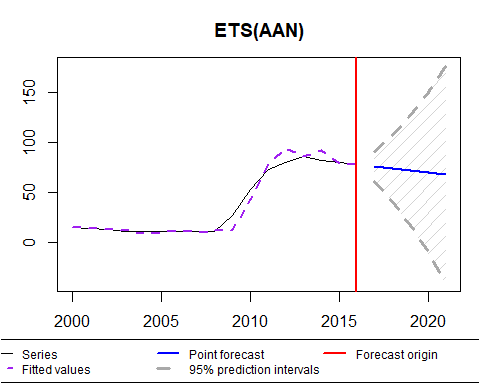
## Forming the pool of models based on... ANN, AAN, Estimation progress: 100%... Done!



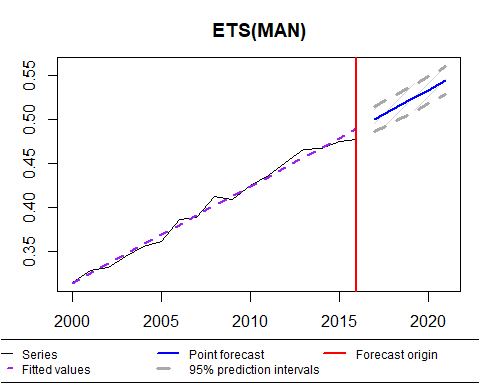
## Forming the pool of models based on... ANN, AAN, Estimation progress: 100%... Done!



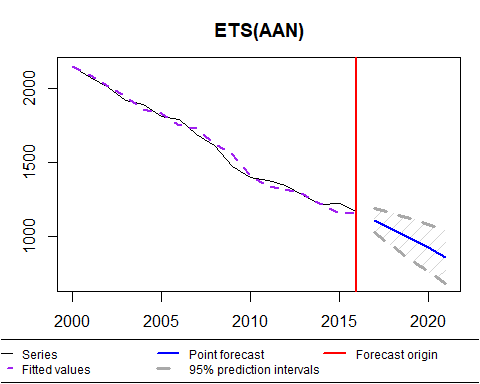
## Forming the pool of models based on... ANN, AAN, Estimation progress: 33%44%56%67%78%89%100%... Done!



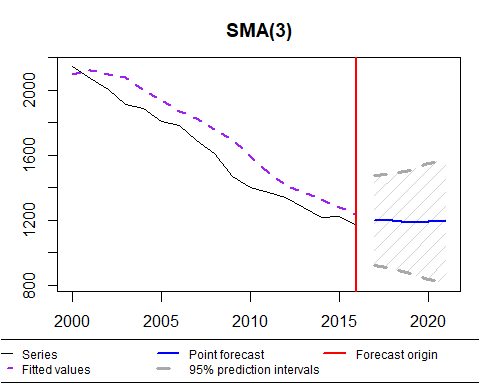
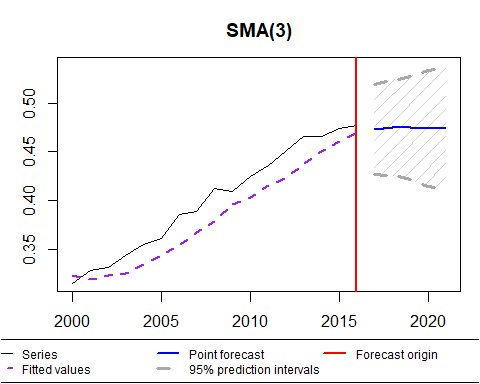
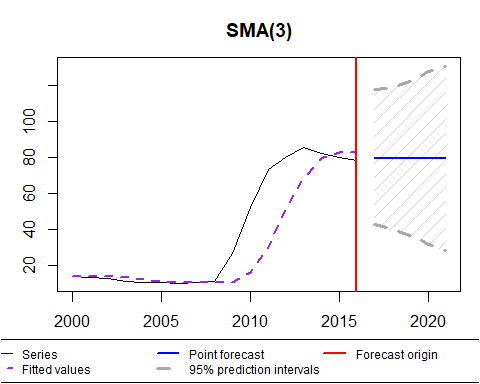
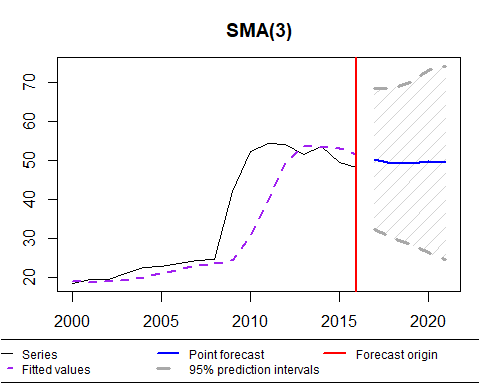
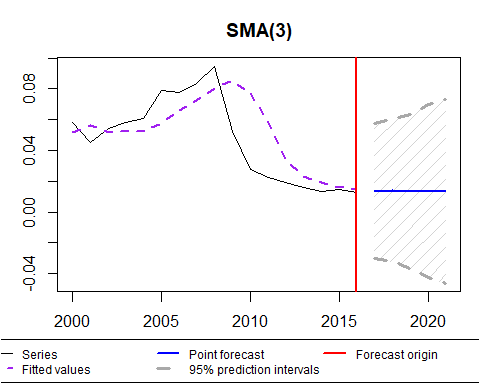
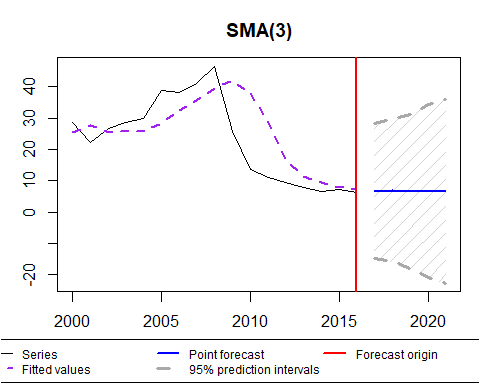
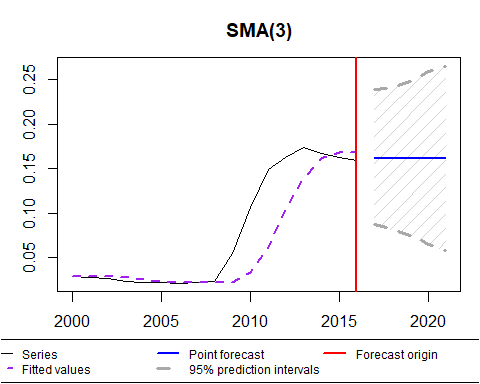
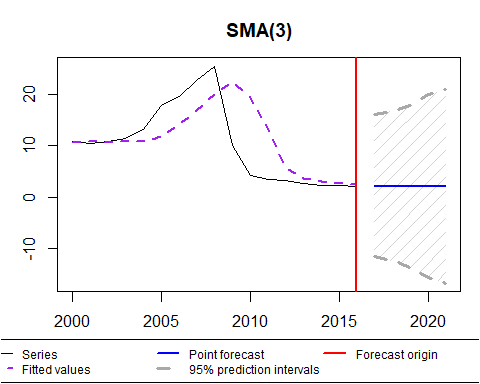
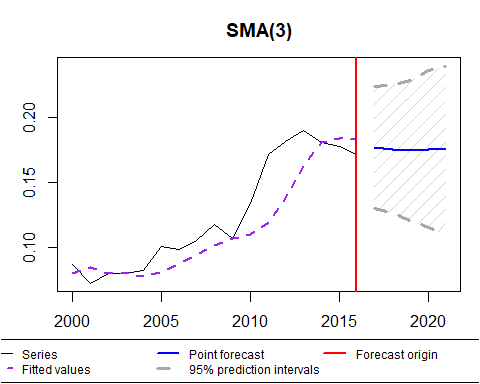
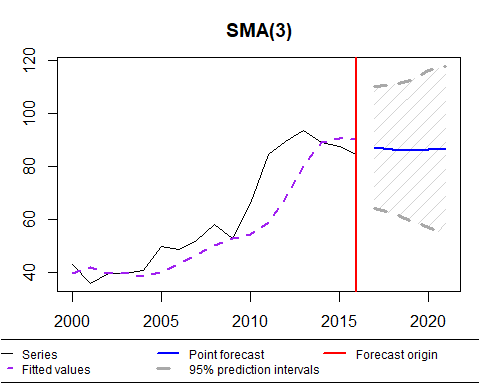
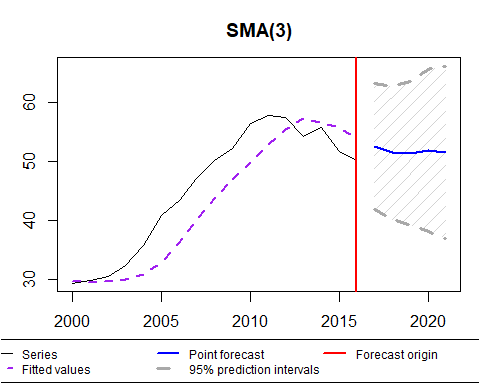
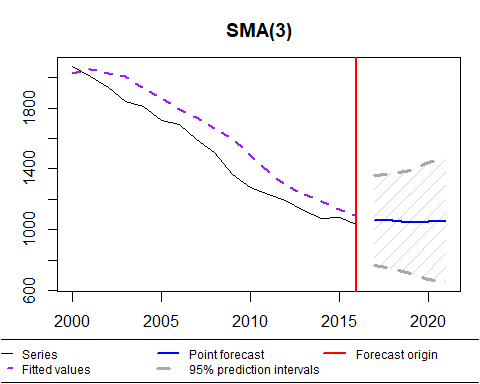
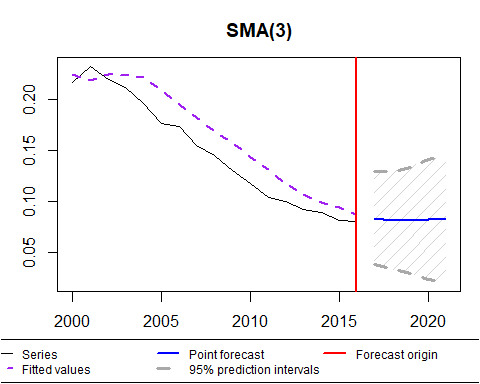
## Forming the pool of models based on... ANN, AAN, Estimation progress: 33%44%56%67%78%89%100%... Done!



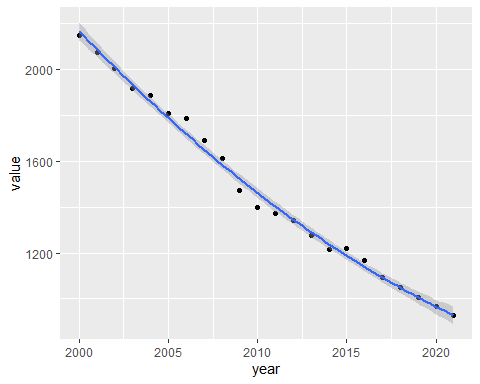
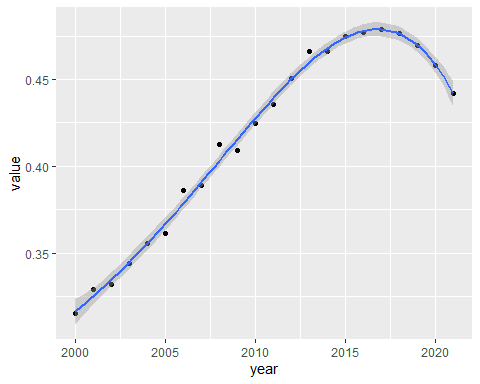
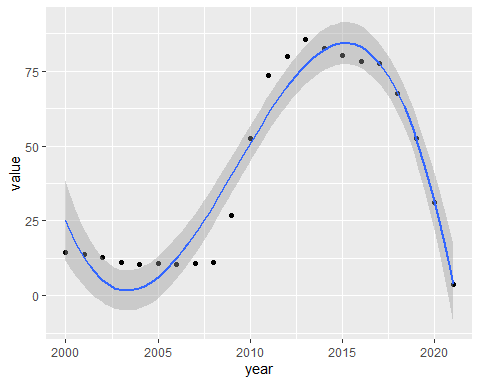
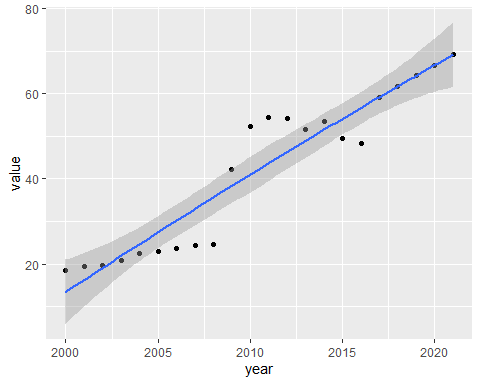
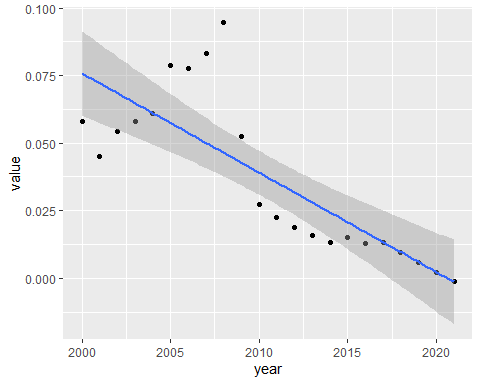
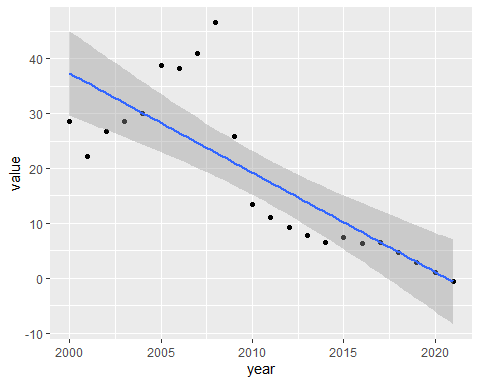
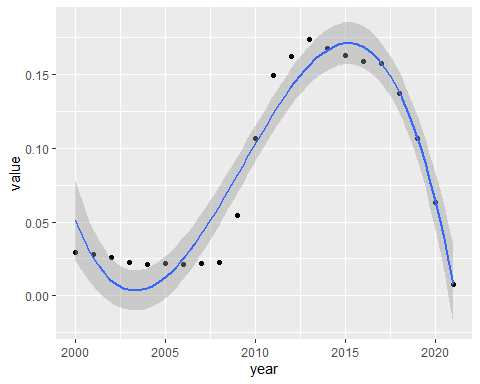
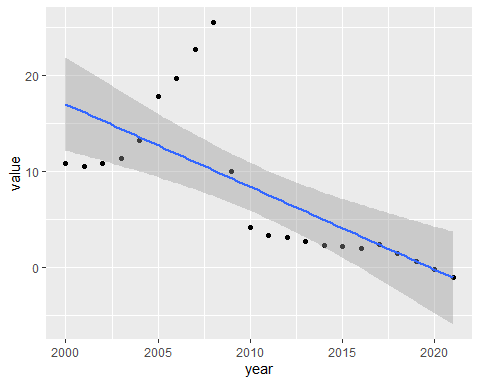
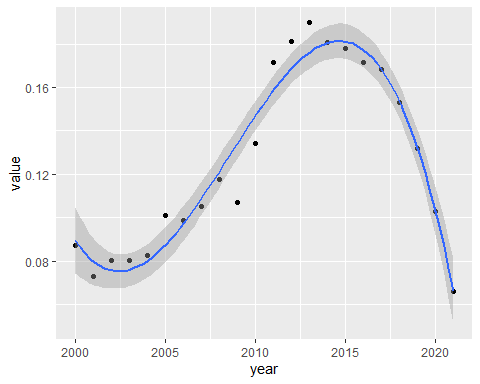
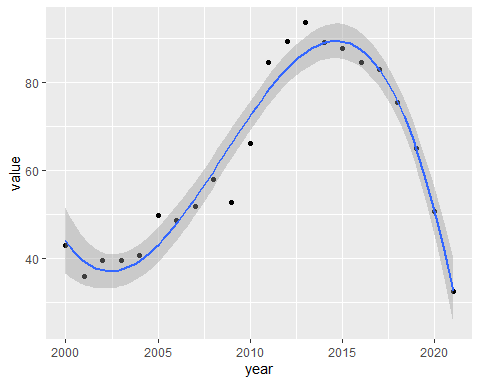
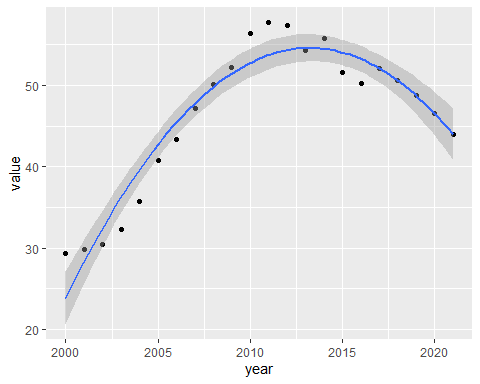
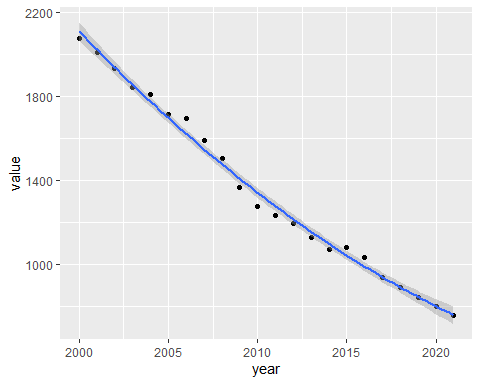
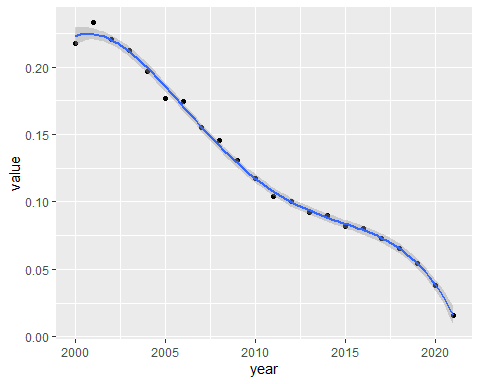
## Forming the pool of models based on... ANN, AAN, Estimation progress: 33%44%56%67%78%89%100%... Done!



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=FALSE)  
   
 #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))  
}



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()))  
}



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 5.409447e-04 38140.60  
## hybErrors 6.235166e-04 37895.13  
## mlpErrors 5.631840e-04 35716.19  
## 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 9.265726  
## hybErrors 19.952000  
## mlpErrors 25.697775  
## expErrors 47.865150  
## smaErrors 1.145378  
## polErrors 7.193974  
## Total Loose Tobacco in Cigarette Equivalents  
## arimaErrors 283.0112  
## nnErrors 253.0774  
## hybErrors 270.9224  
## mlpErrors 269.4958  
## 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 1.048570e-03 1.2790001  
## hybErrors 1.116583e-03 2.1412306  
## mlpErrors 1.105259e-03 1.5254397  
## 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 4.254766e-04  
## hybErrors 7.900426e-04  
## mlpErrors 6.899923e-04  
## expErrors 1.584815e-04  
## smaErrors 5.684163e-04  
## polErrors 2.602271e+03  
## Roll-Your-Own Tobacco in Cigarette Equivalents  
## arimaErrors 21.300362  
## nnErrors 11.065468  
## hybErrors 19.031402  
## mlpErrors 11.929274  
## 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 4.565609e-05 1.416057  
## hybErrors 7.852417e-05 7.791212  
## mlpErrors 5.358204e-05 1.971813  
## 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 121.54749 3.362417e-04  
## hybErrors 208.80058 3.624791e-04  
## mlpErrors 169.88838 2.725109e-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 45260.168  
## hybErrors 45393.960  
## mlpErrors 39383.663  
## 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])])  
}

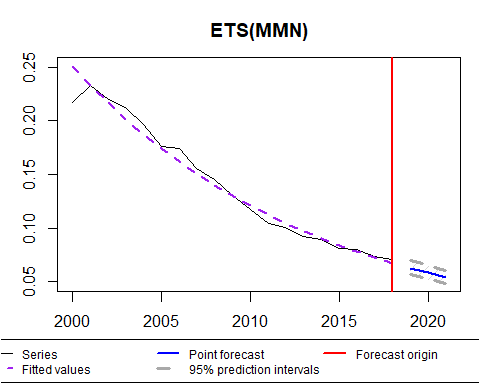
## [1] "expErrors"  
## [1] "expErrors"  
## [1] "smaErrors"  
## [1] "expErrors"  
## [1] "expErrors"  
## [1] "expErrors"  
## [1] "expErrors"  
## [1] "expErrors"  
## [1] "expErrors"  
## [1] "nnErrors"  
## [1] "expErrors"  
## [1] "mlpErrors"  
## [1] "expErrors"

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),frequency = 1)))  
 testTotals10<-c(testTotals10, list(ts(tail(totalsPerCapita[[i]],2),start=c(2019),end=c(2020),frequency = 1)))  
}  
  
#Creating array to store the 2021 forecasts and errors  
forecast2021<-c()  
predictionMSE<-c()

#Generating adn plotting model  
 exp\_model<-es(trainTotals10[[1]], h=3, holdout=FALSE, interval=TRUE, silent=FALSE)

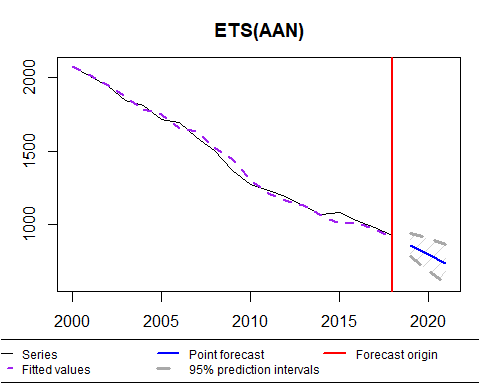
## Forming the pool of models based on... ANN, AAN, Estimation progress: 33%44%56%67%78%89%100%... Done!



#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))

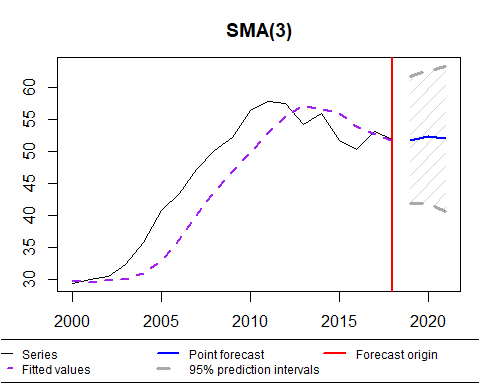
#Generating adn plotting model  
 exp\_model<-es(trainTotals10[[2]], h=3, holdout=FALSE, interval=TRUE, silent=FALSE)

## Forming the pool of models based on... ANN, AAN, Estimation progress: 33%44%56%67%78%89%100%... Done!



#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))

#Generating and plotting model  
 sma\_model<-sma(trainTotals10[[3]], h=3, order=3, holdout=FALSE, interval=TRUE, silent=FALSE)

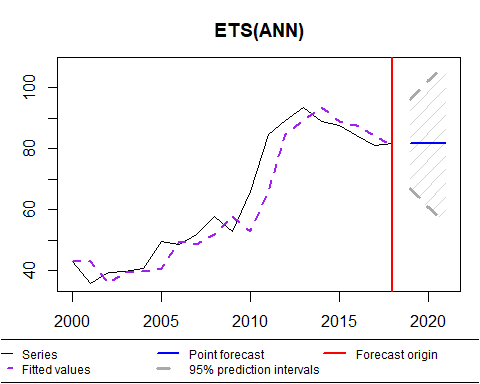


#Getting MSE (the head and tail are used to get from 2017-2020)  
 prediction<-sma\_model$forecast%>%as.numeric()%>%tail(3)%>%head(1)  
 test<-testTotals10[[3]]%>%as.numeric()  
   
 #Saving MSE and forecast  
 predictionMSE<-c(predictionMSE,MSE(prediction,test))  
 forecast2021<-c(forecast2021,sma\_model$forecast%>%as.numeric()%>%tail(1))

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=FALSE)

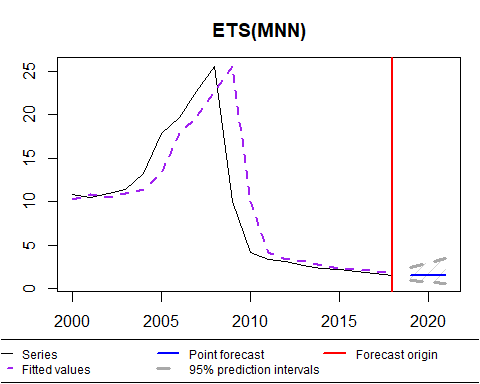
## Forming the pool of models based on... ANN, AAN, Estimation progress: 100%... Done!



#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))

#Generating adn plotting model  
 exp\_model<-es(trainTotals10[[6]], h=3, holdout=FALSE, interval=TRUE, silent=FALSE)

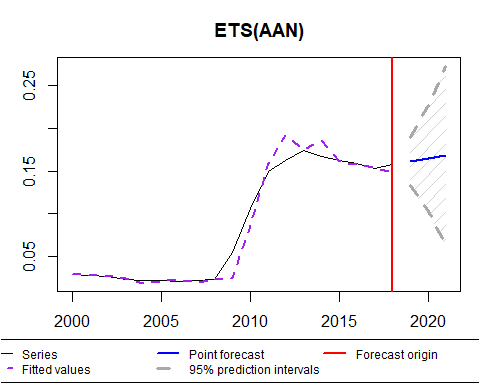
## Forming the pool of models based on... ANN, AAN, Estimation progress: 100%... Done!



#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))

#Generating adn plotting model  
 exp\_model<-es(trainTotals10[[7]], h=3, holdout=FALSE, interval=TRUE, silent=FALSE)

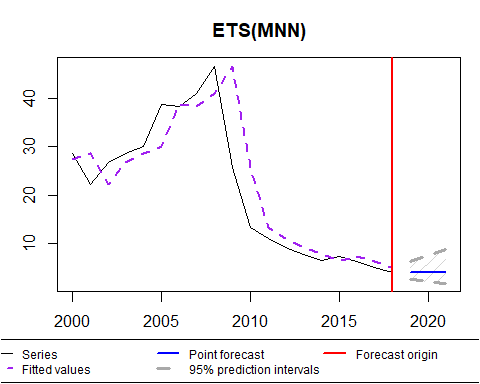
## Forming the pool of models based on... ANN, AAN, Estimation progress: 33%44%56%67%78%89%100%... Done!



#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))

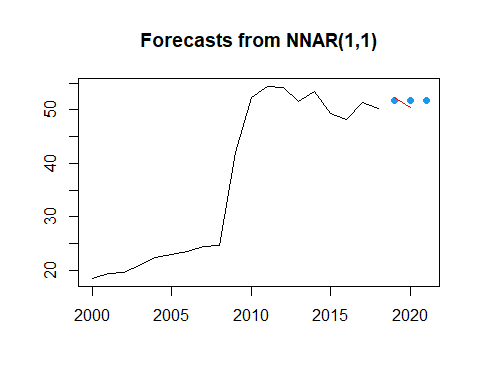
#Generating adn plotting model  
 exp\_model<-es(trainTotals10[[8]], h=3, holdout=FALSE, interval=TRUE, silent=FALSE)

## Forming the pool of models based on... ANN, AAN, Estimation progress: 100%... Done!



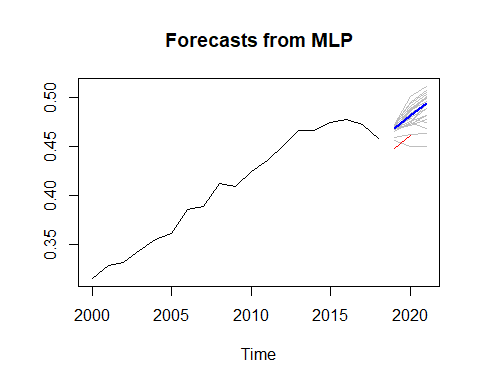
#Getting MSE (the head and tail are used to get from 2019-2020)  
 prediction<-exp\_model$forecast%>%as.numeric()%>%tail(3)%>%head(2)  
 test<-testTotals10[[8]]%>%as.numeric()  
   
 ##Saving MSE and forecast  
 predictionMSE<-c(predictionMSE,MSE(prediction,test))  
 forecast2021<-c(forecast2021,exp\_model$forecast%>%as.numeric()%>%tail(1))

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))

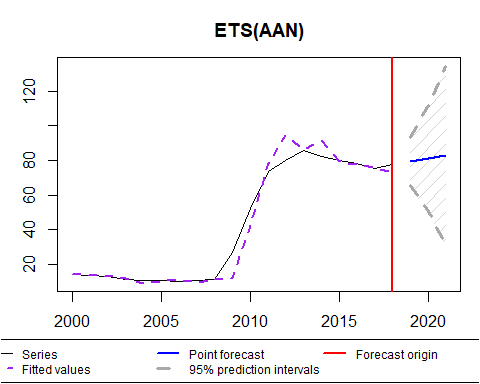
#Training  
 mlp\_fit<-mlp(trainTotals10[[12]])  
 mlp\_model<-forecast::forecast(mlp\_fit,3)  
   
 #Plotting  
 plot(mlp\_model)  
 lines(testTotals10[[12]],col='red')



#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))

library(smooth)  
 #Generating adn plotting model  
 exp\_model<-es(trainTotals10[[11]], h=3, holdout=FALSE, interval=TRUE, silent=FALSE)

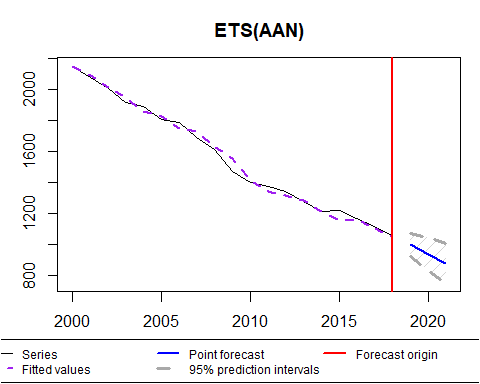
## Forming the pool of models based on... ANN, AAN, Estimation progress: 33%44%56%67%78%89%100%... Done!



#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))

#Generating adn plotting model  
 exp\_model<-es(trainTotals10[[13]], h=3, holdout=FALSE, interval=TRUE, silent=FALSE)

## Forming the pool of models based on... ANN, AAN, Estimation progress: 33%44%56%67%78%89%100%... Done!



#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))