ENV 790.30 - Time Series Analysis for Energy Data | Spring 2021

Assignment 7 - Due date 04/07/21

Student Name

Directions

You should open the .rmd file corresponding to this assignment on RStudio. The file is available on our class repository on Github. And to do so you will need to fork our repository and link it to your RStudio.

Once you have the project open the first thing you will do is change "Student Name" on line 3 with your name. Then you will start working through the assignment by **creating code and output** that answer each question. Be sure to use this assignment document. Your report should contain the answer to each question and any plots/tables you obtained (when applicable).

When you have completed the assignment, **Knit** the text and code into a single PDF file. Rename the pdf file such that it includes your first and last name (e.g., "LuanaLima_TSA_A07_Sp21.Rmd"). Submit this pdf using Sakai.

Set up

Some packages needed for this assignment: `forecast`,`tseries`,`smooth`. Do not forget to load them before running your script, since they are NOT default packages.

```
#Load/install required package here
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(tseries)
library(smooth)
```

```
## Loading required package: greybox
```

```
## Package "greybox", v0.6.8 loaded.
```

```
## This is package "smooth", v3.1.0
```

```
library(data.table)
library(gcookbook)
library(stats)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following object is masked from 'package:greybox':
##
##     hm

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(ggplot2)
library(kableExtra)
```

Importing and processing the data set

Consider the data from the file "inflowtimeseries.txt". The data corresponds to the monthly inflow in m^3/s for some hydro power plants in Brazil. You will only use the last column of the data set which represents one hydro plant in the Amazon river basin. The data span the period from January 1931 to August 2011 and is provided by the Brazilian ISO.

For all parts of the assignment prepare the data set such that the model consider only the data from January 2000 up to December 2009. Leave the year 2010 of data (January 2010 to December 2010) for the out-of-sample analysis. Do **NOT** use data fro 2010 and 2011 for model fitting. You will only use it to compute forecast accuracy of your model.

```
inflow<-read.csv("../Data/inflowtimeseriescleaned.csv", header=FALSE)
inflow_decade<-inflow[829:960,]

inflow_decade_date<-inflow_decade[,1]

colnames(inflow_decade)=c("Date","Inflow")
inflow_decade$`Inflow`<-as.numeric(inflow_decade$`Inflow`)
my_date <- paste(inflow_decade[,1])
my_date <- my(my_date)
my_date_decade<-my_date
my_date<-my_date[1:120]


inflow_decade <- cbind(my_date,inflow_decade[,2])
```

```
## Warning in cbind(my_date, inflow_decade[, 2]): number of rows of result is not a
## multiple of vector length (arg 1)
```

```
head(inflow_decade)
```

```
##      my_date
## [1,]   10957 22107
## [2,]   10988 29247
## [3,]   11017 36651
## [4,]   11048 34089
## [5,]   11078 25821
## [6,]   11109 18007
```

```
nvar <- ncol(inflow_decade) - 1
nobs <- nrow(inflow_decade)

ncolumns_decade <- ncol(inflow_decade)
nmonths_decade <- nrow(inflow_decade)

inflow_decade_ts<-ts(inflow_decade[,2], start=c(2000, 1), end=c(2010,12), frequency=12)
inflow_decade_ts<-cbind(my_date_decade,inflow_decade_ts)
```

```
inflow_09<-inflow_decade[1:120,]

head(inflow_09)
```

```
##      my_date
## [1,]   10957 22107
## [2,]   10988 29247
## [3,]   11017 36651
## [4,]   11048 34089
## [5,]   11078 25821
## [6,]   11109 18007
```

```
ncolumns <- ncol(inflow_09)
nmonths <- nrow(inflow_09)

inflow_09_ts<-ts(inflow_09, start=c(2000, 1), end=c(2009,12), frequency=12)
```

Part I: Preparing the data sets

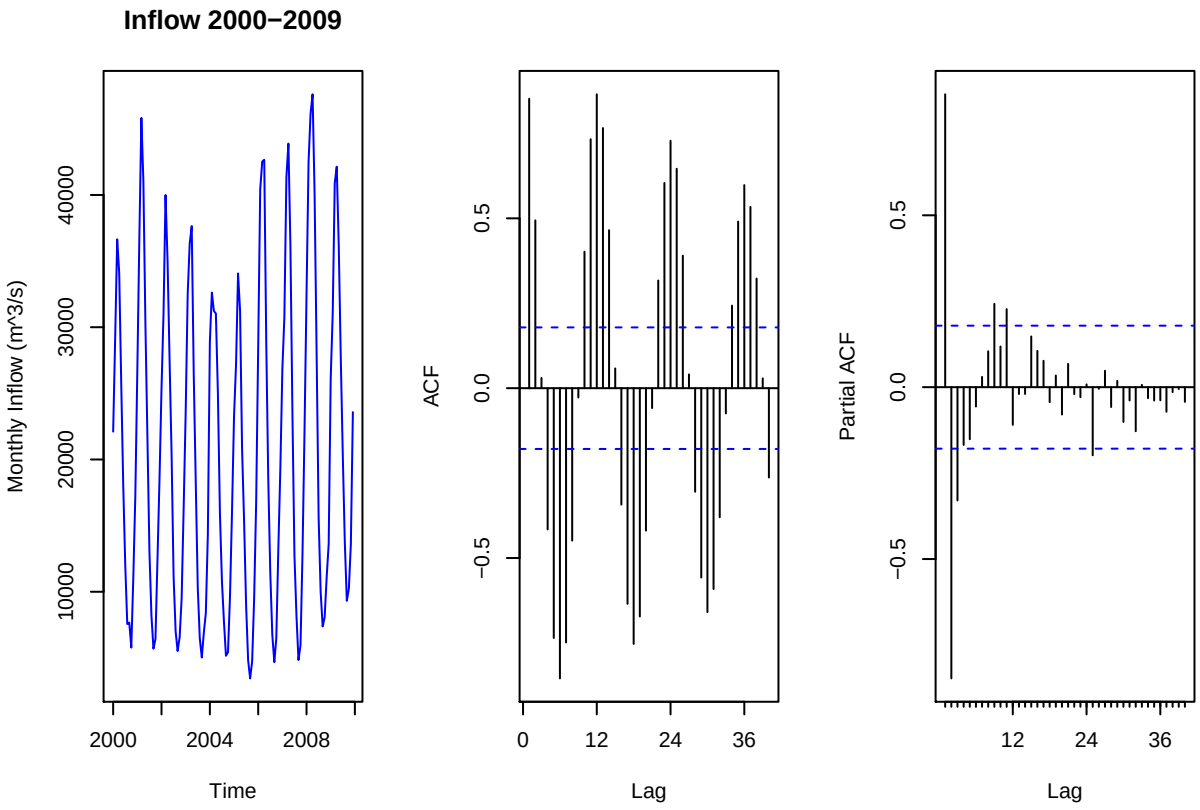
Q1

Read the file into a data frame. Prepare your time series data vector such that observations start in January 2000 and end in December 2009. Make you sure you specify the **start=** and **frequency=** arguments. Plot the time series over time, ACF and PACF.

```
par(mfrow=c(1,3))
plot(inflow_09_ts[,2], ylab="Monthly Inflow (m^3/s)",  col=c("blue"))+title("Inflow 2000-2009")
```

```
## integer(0)
```

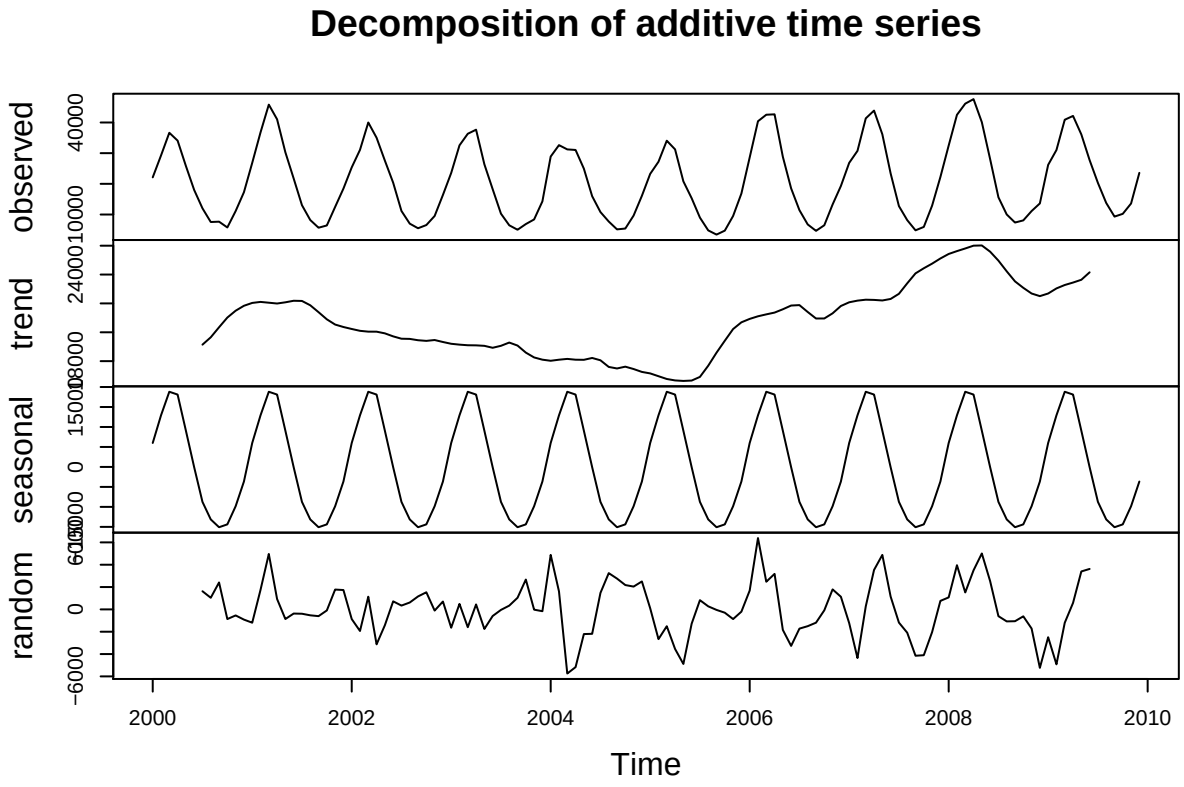
```
ACF_Plot <- Acf(inflow_09_ts[,2], lag = 40, plot = TRUE,main="")
PACF_Plot <- Pacf(inflow_09_ts[,2], lag = 40, plot = TRUE,main="")
```



Q2

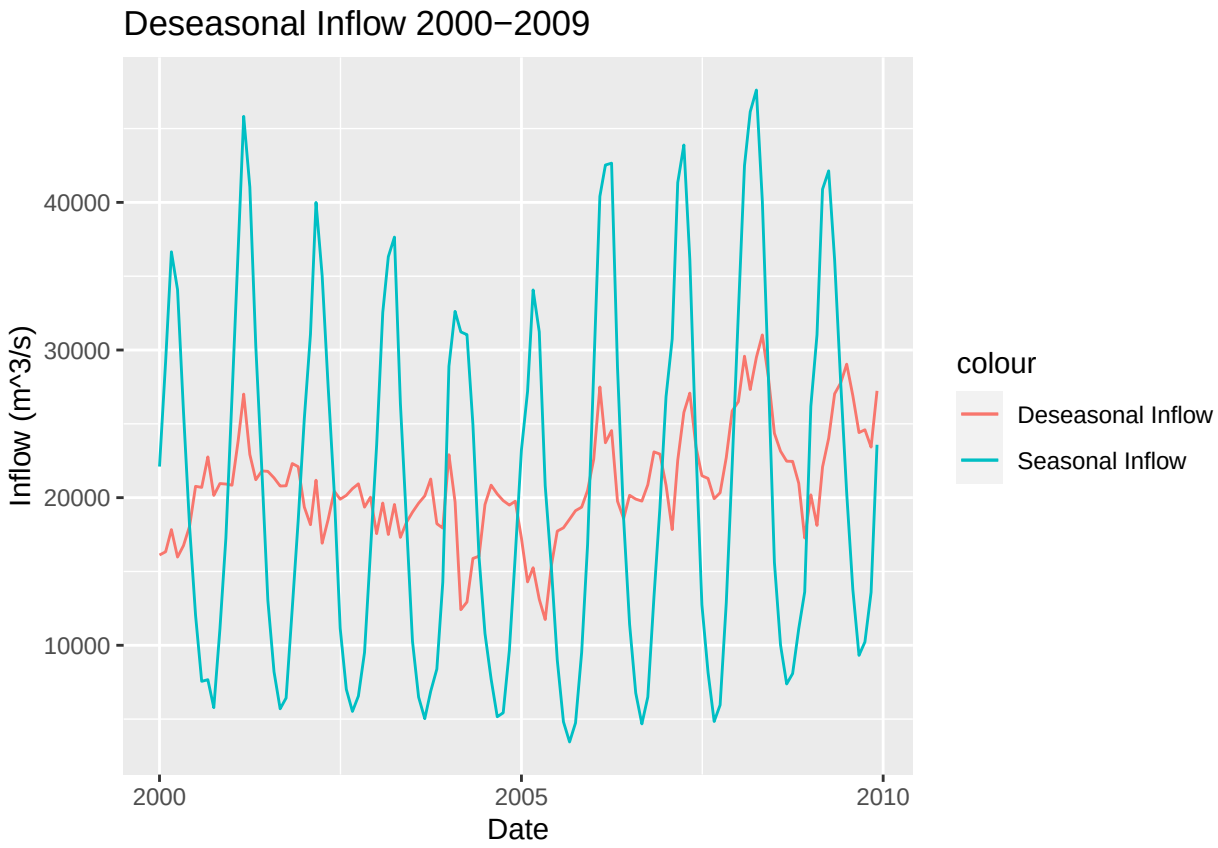
Using the *decompose*() or *stl*() and the *seasadj*() functions create a series without the seasonal component, i.e., a deseasonalized inflow series. Plot the deseasonalized series and original series together using ggplot, make sure your plot includes a legend. Plot ACF and PACF for the deaseasonalized series. Compare with the plots obtained in Q1.

```
decompose_inflow_09_ts <- decompose(inflow_09_ts[,2],"additive")
plot(decompose_inflow_09_ts)
```

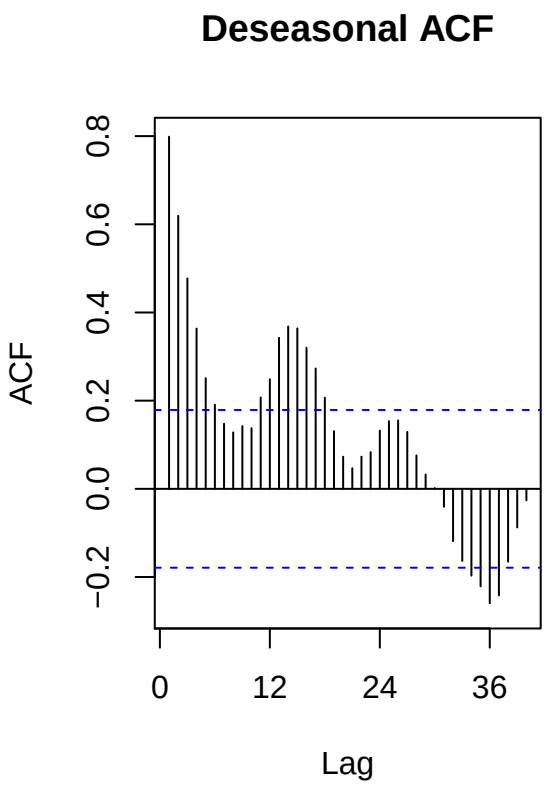


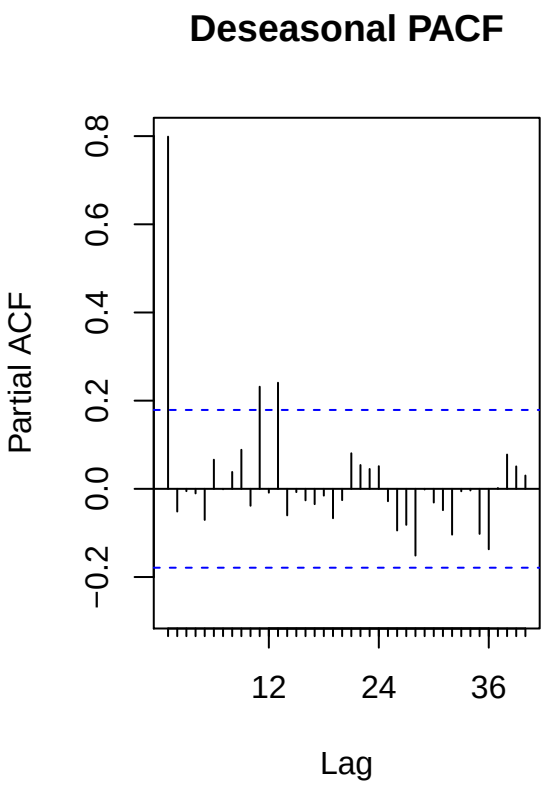

```
deseasonal_inflow_09_ts <- seasadj(decompose_inflow_09_ts)

#par(mfrow=c(1,3))


ggplot()+
  geom_line(aes(x = my_date, y = deseasonal_inflow_09_ts, color = "Deseasonal Inflow")) +
  geom_line(aes(x = my_date, y = inflow_09_ts[,2], color = "Seasonal Inflow"))+
  ylab("Inflow (m^3/s)") +
  xlab("Date") +
  ggtitle("Deseasonal Inflow 2000-2009")
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```

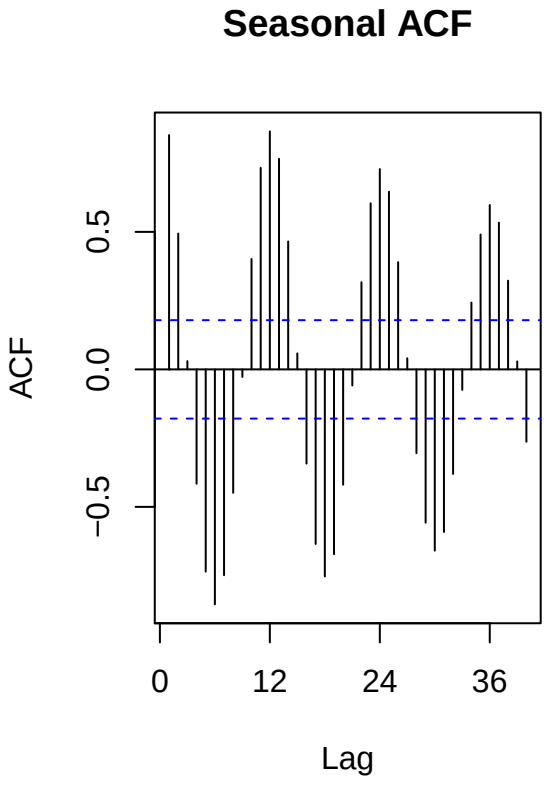


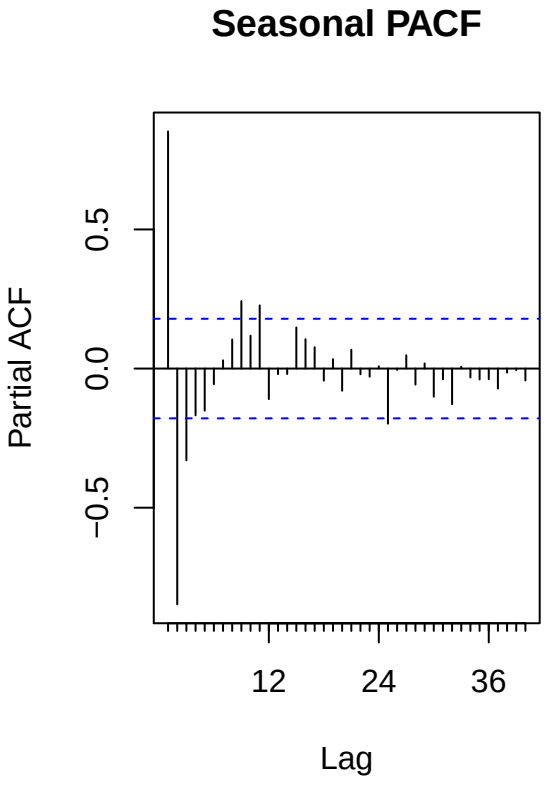

```
par(mfrow=c(1,2))
ACF_Plot_deseasonal <- Acf(deseasonal_inflow_09_ts, lag = 40, plot = TRUE,main="Deseasonal ACF")
PACF_Plot_deseasonal <- Pacf(deseasonal_inflow_09_ts, lag = 40, plot = TRUE,main="Deseasonal PACF")
```






```
par(mfrow=c(1,2))
ACF_Plot <- Acf(inflow_09_ts[,2], lag = 40, plot = TRUE,main="Seasonal ACF")
PACF_Plot <- Pacf(inflow_09_ts[,2], lag = 40, plot = TRUE,main="Seasonal PACF")
```





After

deseasoning the data and based on the plot, it seems that the function worked and we are just left with the trend and the random components. However, the ACF of the deseasonal data still seems to show a significant seasonal autocorrelation. There certainly is less seasonality when compared to the seasonal plots but it has not been eliminated entirely. Just looking at the decomposed plot, it seems as though the random element the decompose() function identified still has some seasonality in it.

Part II: Forecasting with ARIMA models and its variations

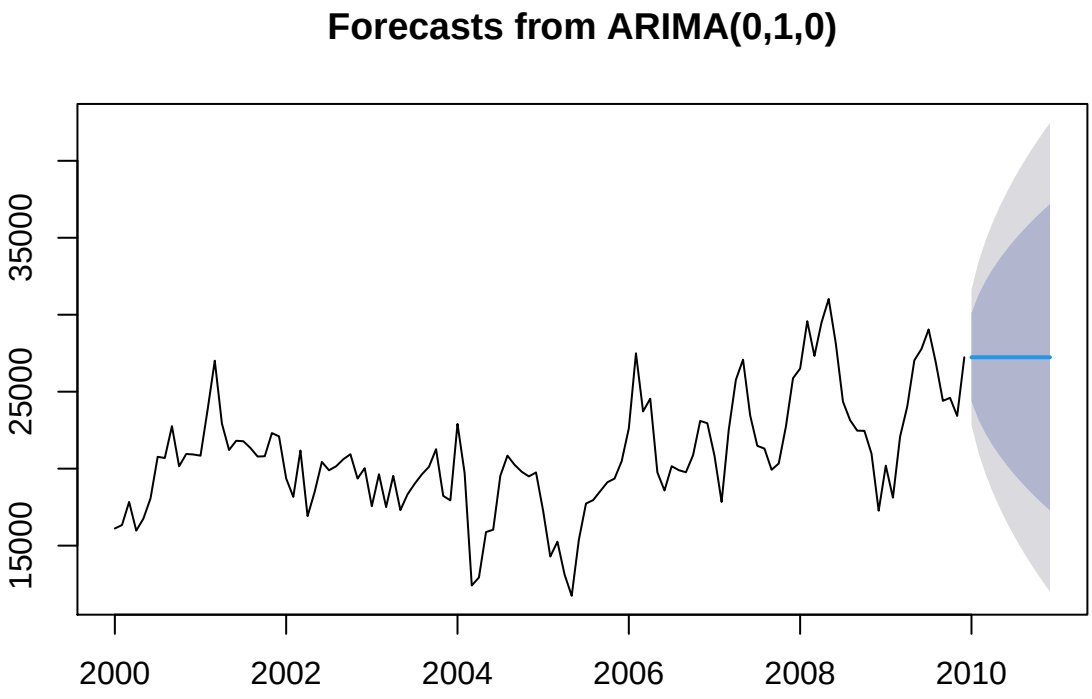
Q3

Fit a non-seasonal ARIMA(p, d, q) model usign the auto.arima() function to the non-seasonal data. Forecast 12 months ahead of time using the $forecast()$ function. Plot your forecasting results and further include on the plot the last year of non-seasonal data to compare with forecasted values (similar to the plot on the lesson file for M10).

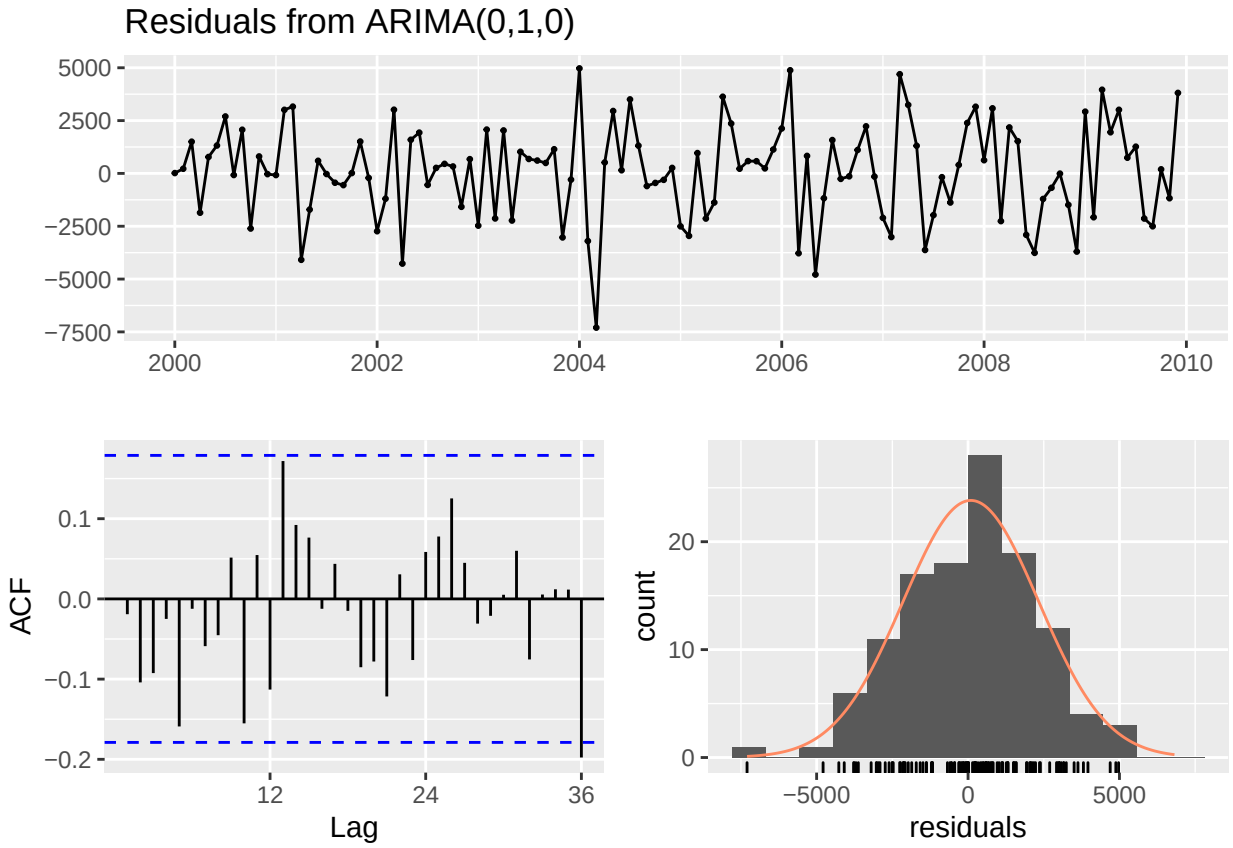
```
nonseasonal_ARIMA_fit <- auto.arima(deseasonal_inflow_09_ts)
print(nonseasonal_ARIMA_fit)
```

```
## Series: deseasonal_inflow_09_ts
## ARIMA(0,1,0)
##
## sigma^2 estimated as 5031380:  log likelihood=-1087.01
## AIC=2176.02   AICc=2176.05   BIC=2178.8
```

```
ARIMA_forecast <- forecast(object = nonseasonal_ARIMA_fit, h = 12)
plot(ARIMA_forecast)
```




```
checkresiduals(ARIMA_forecast)
```

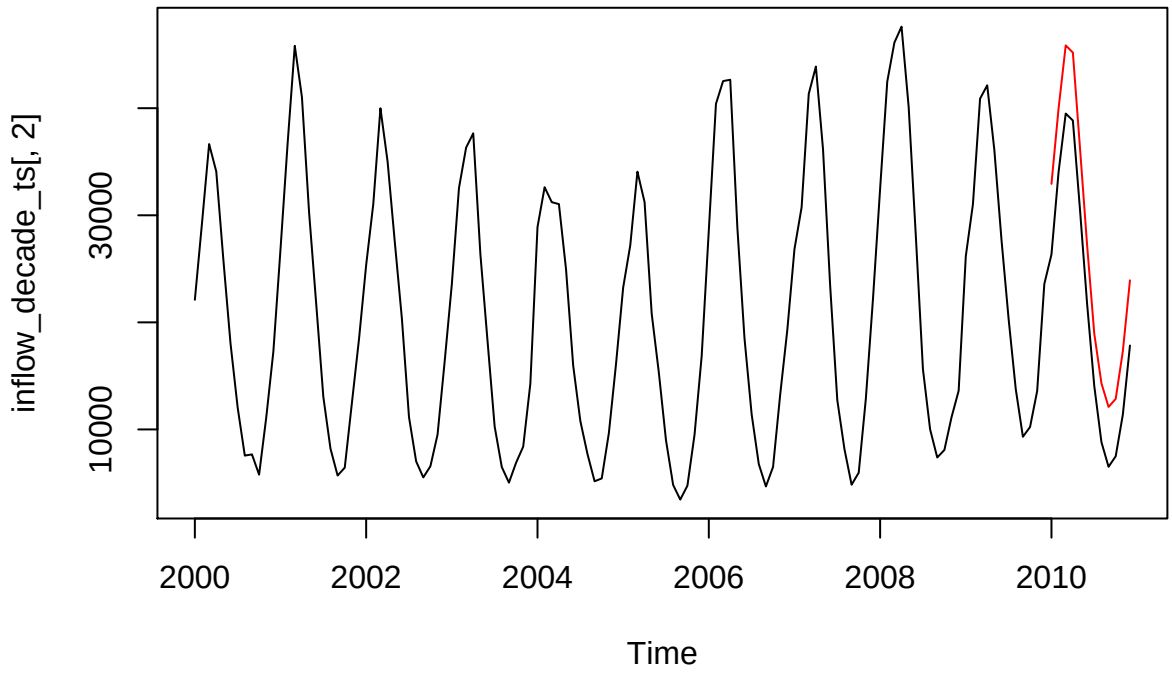



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0)
## Q* = 24.208, df = 24, p-value = 0.4498
##
## Model df: 0.   Total lags used: 24
```

Q4

Put the seasonality back on your forecasted values and compare with the original seasonal data values. *Hint* : One way to do it is by summing the last year of the seasonal component from your decompose object to the forecasted series.

```
decompose_inflow_decade_ts <- decompose(inflow_decade_ts[,2],"additive")
SNAIVE_Inflow <- window(decompose_inflow_decade_ts$seasonal, start = 2010) + window(ARIMA_forecast$mean

plot(inflow_decade_ts[,2])
lines(SNAIVE_Inflow,col="red")
```



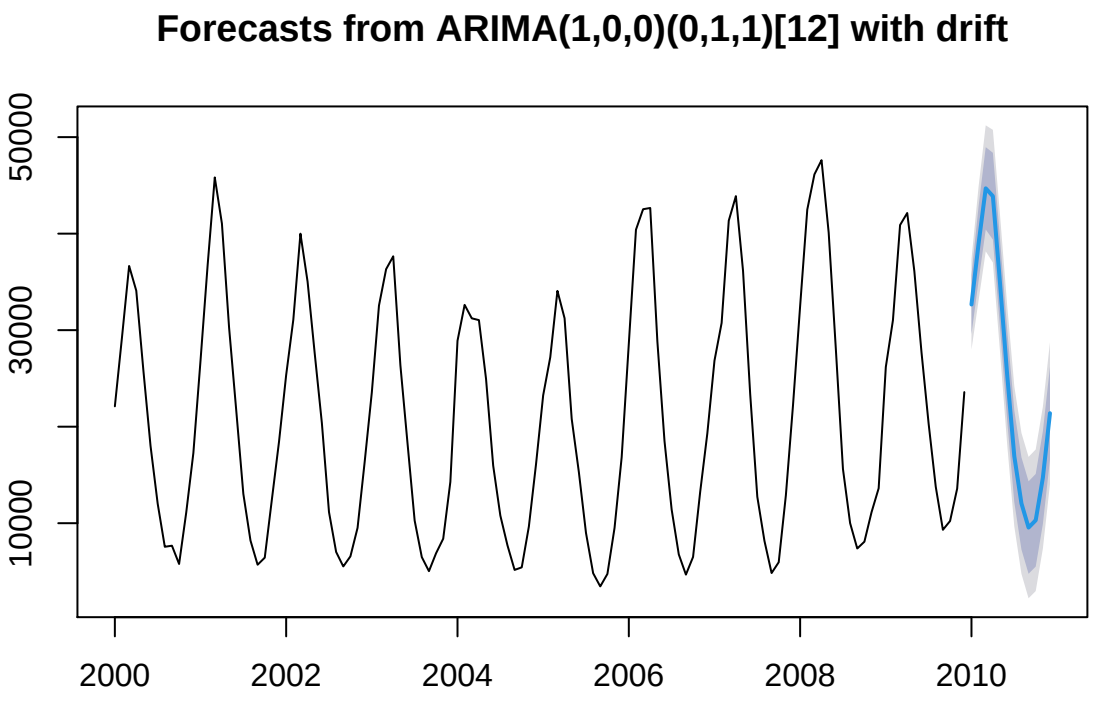
Q5

Repeat Q3 for the original data, but now fit a seasonal ARIMA$(p,d,q)x(P,D,Q)_{12}$ also using the auto.arima().

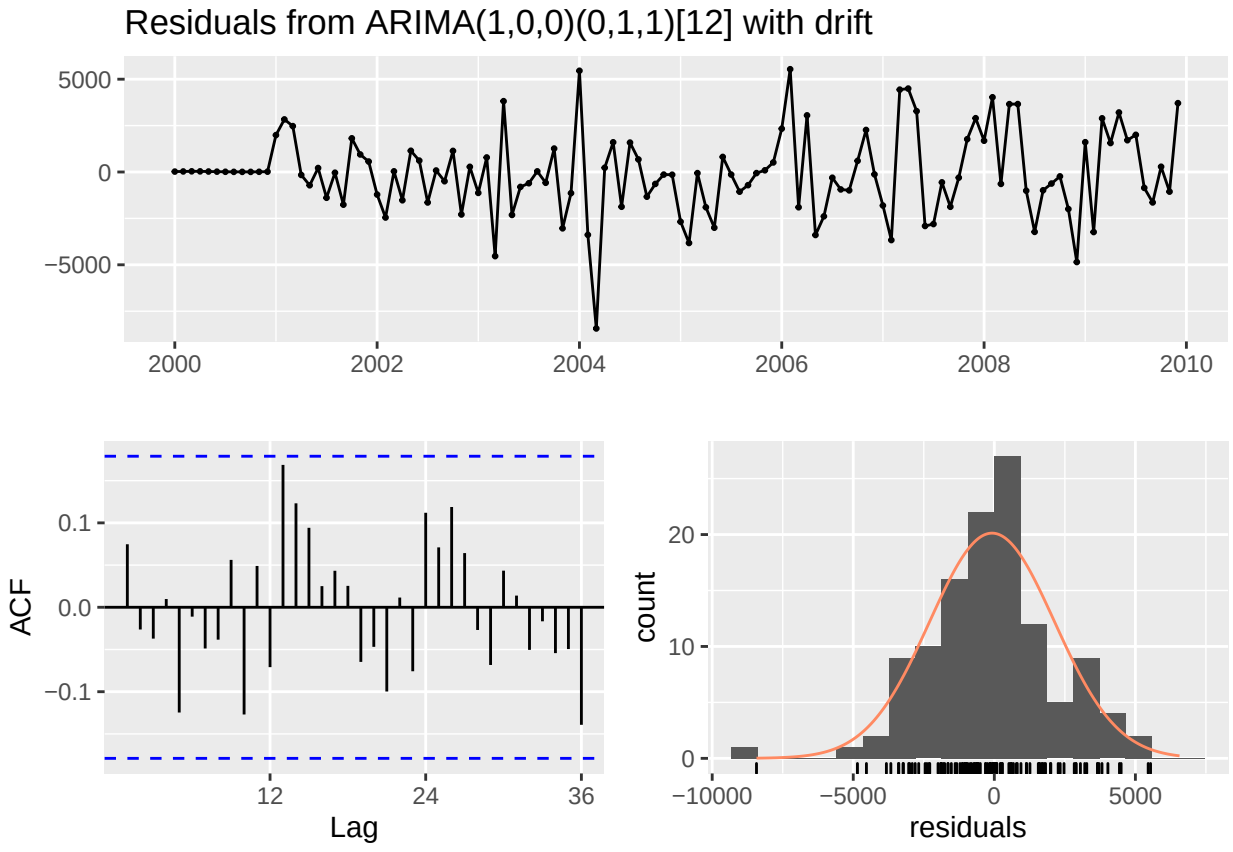
```
SARIMA_fit <- auto.arima(inflow_09_ts[,2])
print(SARIMA_fit)
```

```
## Series: inflow_09_ts[, 2]
## ARIMA(1,0,0)(0,1,1)[12] with drift
##
## Coefficients:
##          ar1     sma1    drift
##       0.7739  -0.8724  54.7963
## s.e.  0.0614   0.1767  25.8402
##
## sigma^2 estimated as 5566545:  log likelihood=-999.07
## AIC=2006.14   AICc=2006.52   BIC=2016.86
```

```
SARIMA_forecast <- forecast(object = SARIMA_fit, h = 12)
plot(SARIMA_forecast)
```




```
checkresiduals(SARIMA_forecast)
```




```
##
##  Ljung-Box test
```

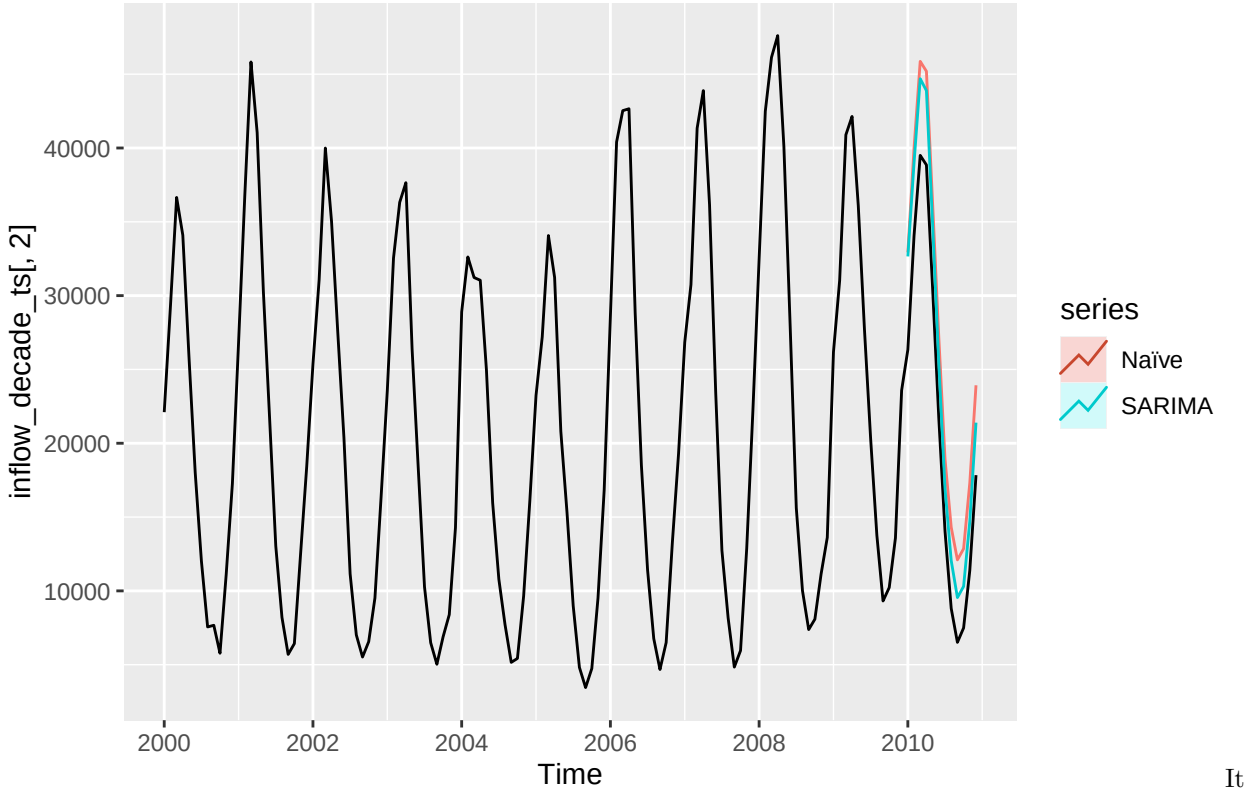
```
##
## data:  Residuals from ARIMA(1,0,0)(0,1,1)[12] with drift
## Q* = 19.881, df = 21, p-value = 0.5288
##
## Model df: 3.   Total lags used: 24
```

Q6

Compare the plots from Q4 and Q5 using the autoplot() function.

```
autoplot(inflow_decade_ts[,2]) +
  autolayer(SNAIVE_Inflow, PI=FALSE, series="Naïve") +
  autolayer(SARIMA_forecast,PI=FALSE, series="SARIMA")
```

```
## Warning: Ignoring unknown parameters: PI
```



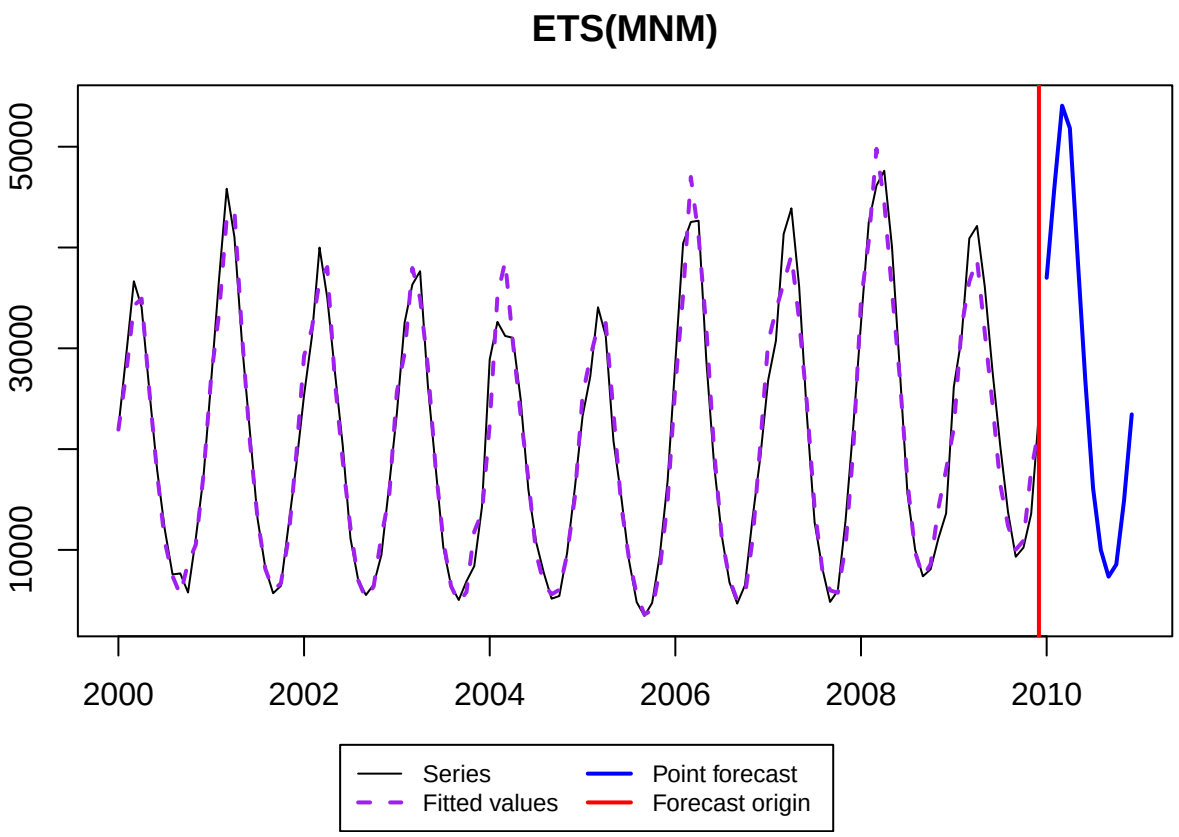
It seems as though the SARIMA model works slightly better as a predictor of our 2010 data than the naive model. The seasonality is more accurate in the above graph.

Part III: Forecasting with Other Models

Q7

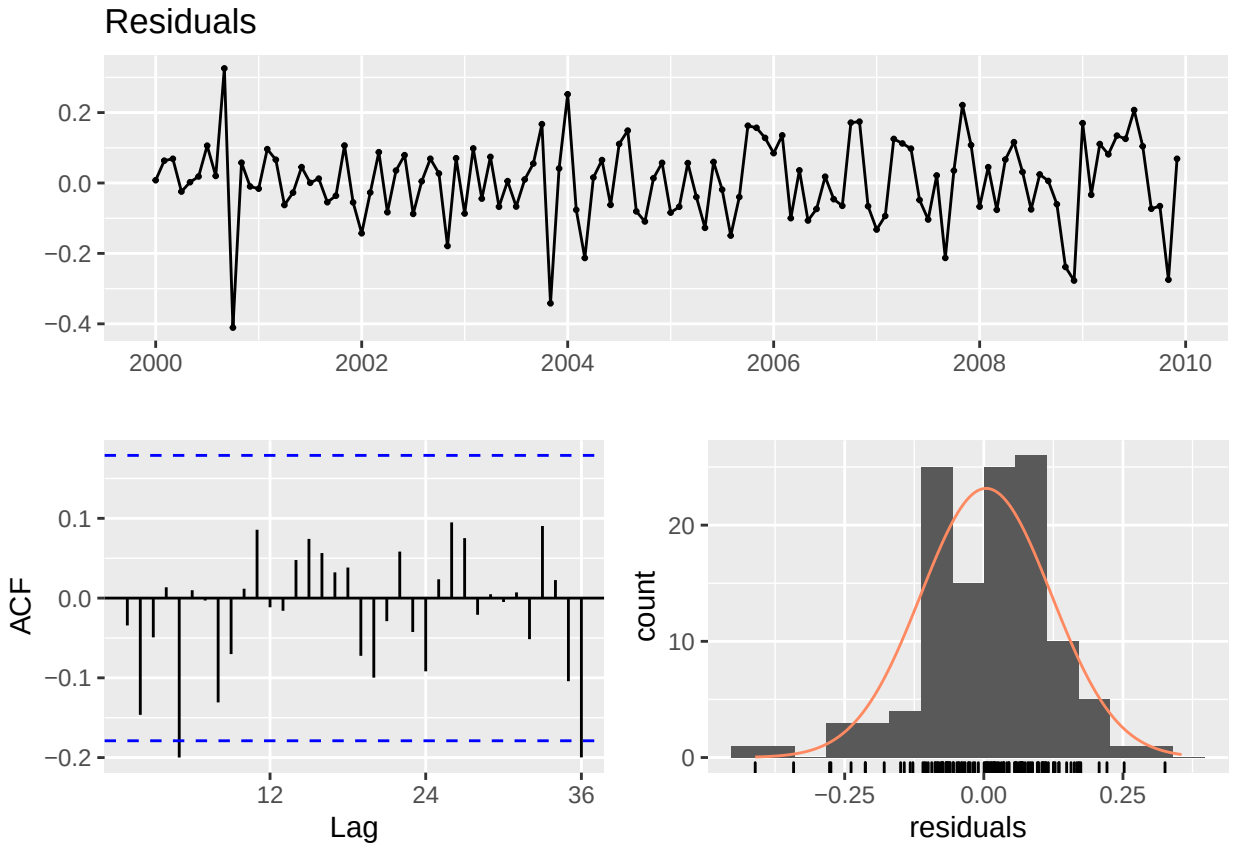
Fit an exponential smooth model to the original time series using the function *es*() from package `smooth`. Note that this function automatically does the forecast. Do not forget to set the arguments: silent=FALSE and holdout=FALSE, so that the plot is produced and the forecast is for the year of 2010.

```
InflowES_seas<-es(inflow_09_ts[,2], model="ZZZ", h=12, silent="output", holdout=FALSE)
```




```
checkresiduals(InflowES_seas)
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```

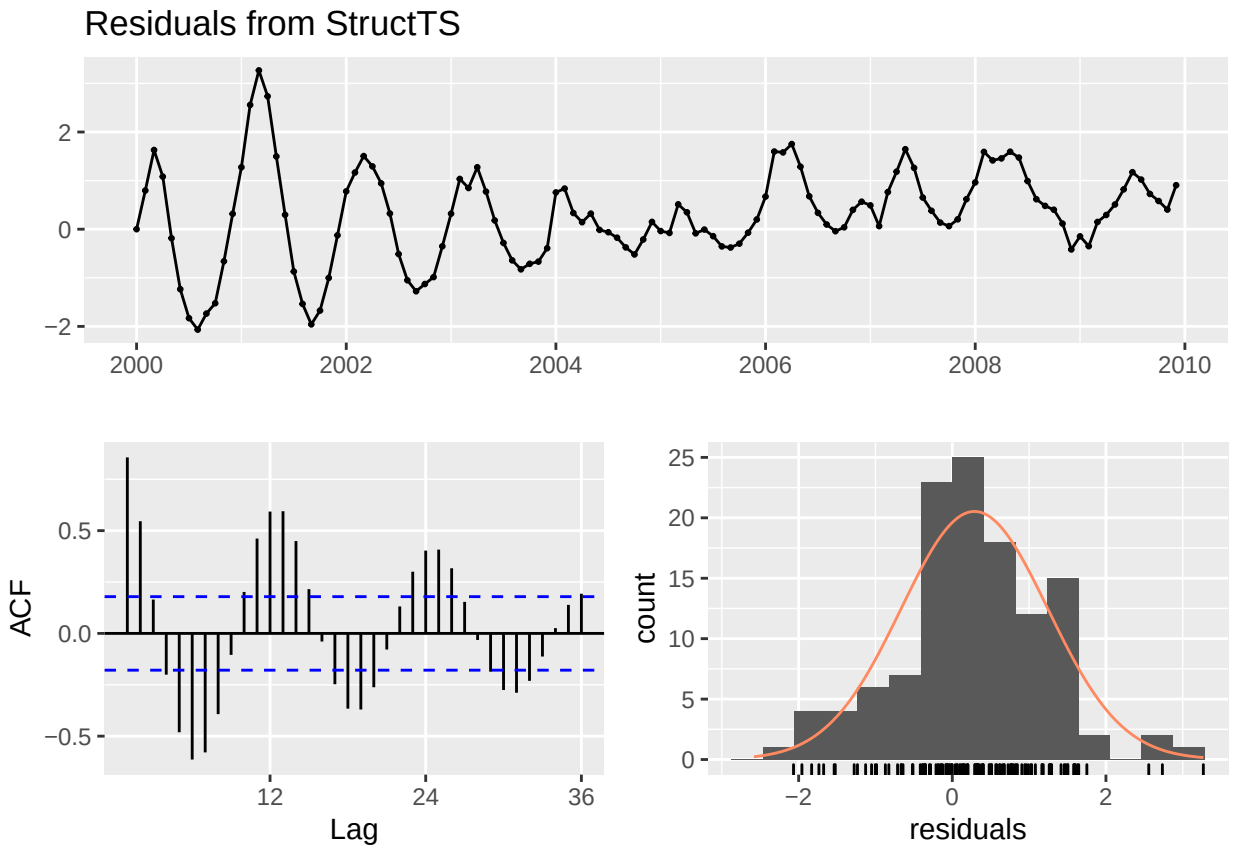


Q8

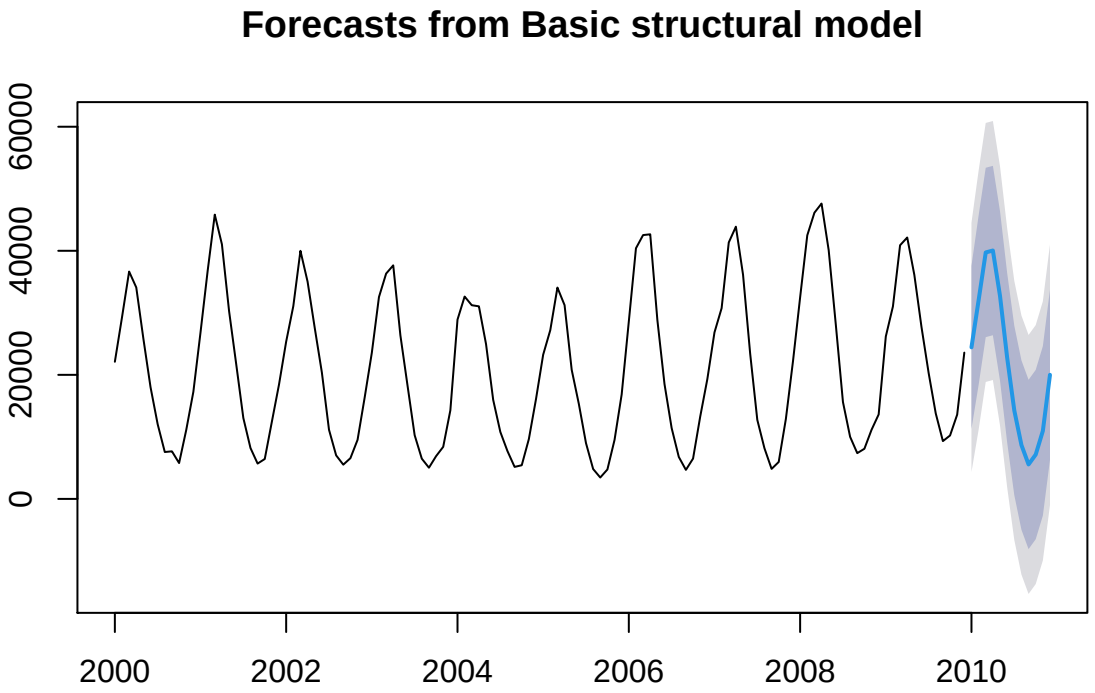
Fit a state space model to the original time series using the function $StructTS()$ from package `stats`. Which one of the tree model we learned should you try: "local", "trend", or "BSM". Why? Play with argument `fixed` a bit to try to understand how the different variances can affect the model. If you can't seem to find a variance that leads to a good fit here is a hint: try $fixed = c(0.1, 0.001, NA, NA)$. Since $StructTS()$ fits a state space model to the data, you need to use $forecast()$ to generate the forecasts. Like you do for the ARIMA fit.

```
InflowSP_seas <- StructTS(inflow_09_ts[,2],
                    type="BSM",fixed=c(0,0.001,NA,NA))    #this function has convergence issues
checkresiduals(InflowSP_seas)
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```




```
#Generating forecasts
# StructTS() does not call the forecast() internally so we need one more step
InflowSP_seas_for <- forecast(InflowSP_seas,h=12)
plot(InflowSP_seas_for)
```



We should

use the BSM tree model because this is a seasonal time series model and a basic structural model fits this series by maximum likelihood so it's pretty accurate.

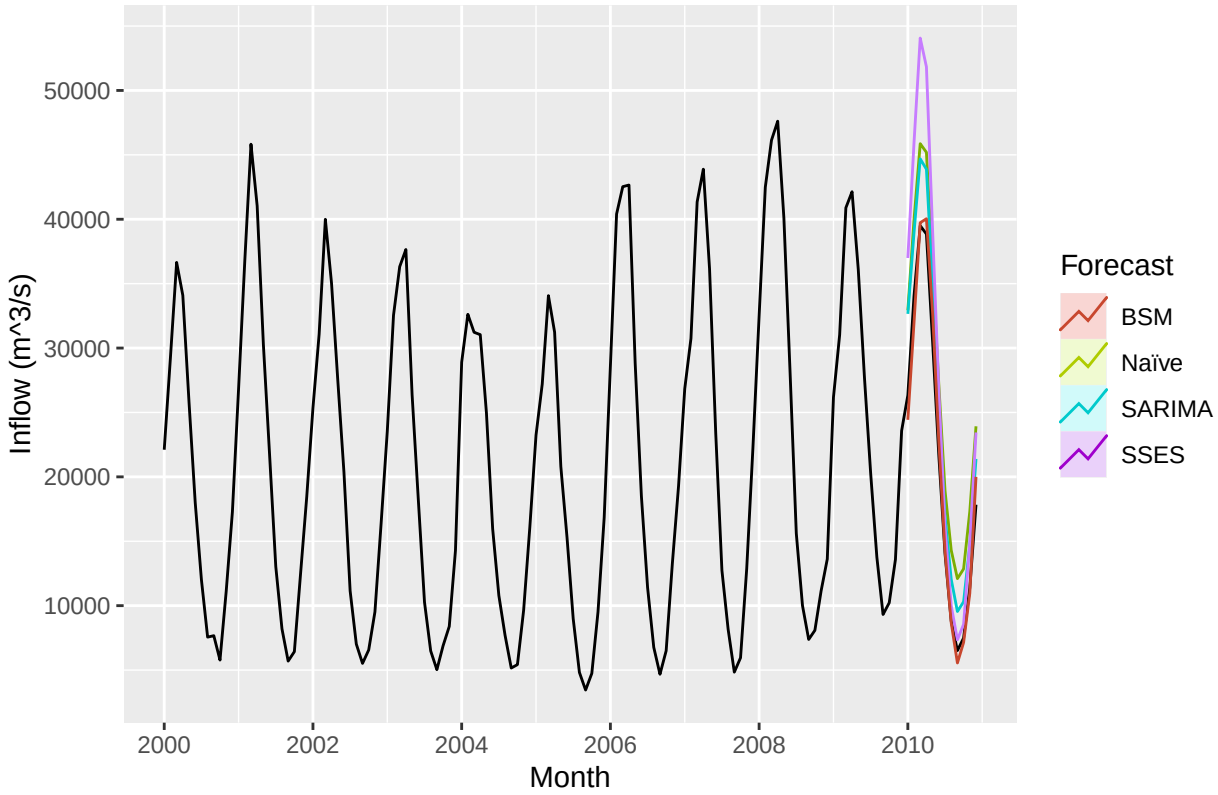
Part IV: Checking Forecast Accuracy

Q9

Make one plot with the complete original seasonal historical data (Jan 2000 to Dec 2010). Now add the forecasts from each of the developed models in parts Q4, Q5, Q7 and Q8. You can do it using the autoplot() combined with autolayer(). If everything is correct in terms of time line, the forecasted lines should appear only in the final year. If you decide to use ggplot() you will need to create a data frame with all the series will need to plot. Remember to use a different color for each model and add a legend in the end to tell which forecast lines corresponds to each model.

```
autoplot(inflow_decade_ts[,2]) +
  #autolayer(ARIMA_forecast, PI=FALSE, series="ARIMA") +
  autolayer(SNAIVE_Inflow, PI=FALSE, series="Naïve") +
  autolayer(SARIMA_forecast,PI=FALSE, series="SARIMA") +
  autolayer(InflowES_seas$forecast, series="SSES") +
  autolayer(InflowSP_seas_for,PI=FALSE,series="BSM") +
  xlab("Month") + ylab("Inflow (m^3/s)") +
  guides(colour=guide_legend(title="Forecast"))
```

```
## Warning: Ignoring unknown parameters: PI
```



Q10

From the plot in Q9 which model or model(s) are leading to the better forecasts? Explain your answer. Hint: Think about which models are doing a better job forecasting the high and low inflow months for example.

Based on the plot in Q9, it appears that the BSM model is consistently leading to better forecasts. The BSM seems to properly follow both the highs and lows of the seasonality of 2010 data. ### Q11

Now compute the following forecast metrics we learned in class: RMSE and MAPE, for all the models you plotted in part Q9. You can do this by hand since your have forecasted and observed values for the year of 2010. Or you can use R function *accuracy*() from package "forecast" to do it. Build and a table with the results and highlight the model with the lowest MAPE. Does the lowest MAPE corresponds match your answer for part Q10?

```
last_obs <- inflow_decade[(nobs-12+1):nobs]

#Model 1: Arithmetic mean
ARIMA_scores <- accuracy(ARIMA_forecast$mean,last_obs)  #store the performance metrics

#Model 2: Seasonal naive
SNAIVE_scores <- accuracy(SNAIVE_Inflow,last_obs)

# Model 3:  SARIMA
SARIMA_scores <- accuracy(SARIMA_forecast$mean,last_obs)

# Model 4:  SSES
SSES_scores <- accuracy(InflowES_seas$forecast,last_obs)

# Model 5:  BSM
SS_scores <- accuracy(InflowSP_seas_for$mean,last_obs)

#Create Data Frame
scores <- as.data.frame(rbind(ARIMA_scores, SNAIVE_scores, SARIMA_scores,SSES_scores,SS_scores))
row.names(scores) <- c("ARIMA", "SNAIVE","SARIMA","SSES","BSM")

#choose model with lowest RMSE
best_model_index <- which.min(scores[,"RMSE"])
cat("The best model by RMSE is:", row.names(scores[best_model_index,]))
```

```
## The best model by RMSE is: BSM
```

```
kbl(scores,
      caption = "Forecast Accuracy for Seasonal Data",
      digits = array(5,ncol(scores))) %>%
  kable_styling(full_width = FALSE, position = "center") %>%
  #highlight model with lowest RMSE
  kable_styling(latex_options="striped", stripe_index = which.min(scores[,"RMSE"]))
```

My answer to part 10, that the BSM model was the best predictor for the 2010 data is confirmed by the table above given its RMSE and its MAE which are both lower than any other model. Based on my own observation and the accuracy scores in the above table, the BSM model is the best predictor of our seasonal data.

Table 1: Forecast Accuracy for Seasonal Data

	ME	RMSE	MAE	MPE	MAPE
ARIMA	-16111.12	16111.46	16111.12	-144.84962	144.8496
SNAIVE	-16111.12	20136.06	16111.12	-145.64572	145.6457
SARIMA	-14324.48	19120.31	14754.58	-129.64316	133.4792
SSES	-16795.37	23552.24	18078.04	-152.10015	163.5456
BSM	-10399.05	15896.54	12486.25	-94.23755	112.8632