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

Assignment 7 - Due date 04/07/21

Xueying Feng

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(lubridate)
library(ggplot2)
library(forecast)
library(Kendall)
library(tseries)
#library(outliers)
library(tidyverse)
library(smooth)
```

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.

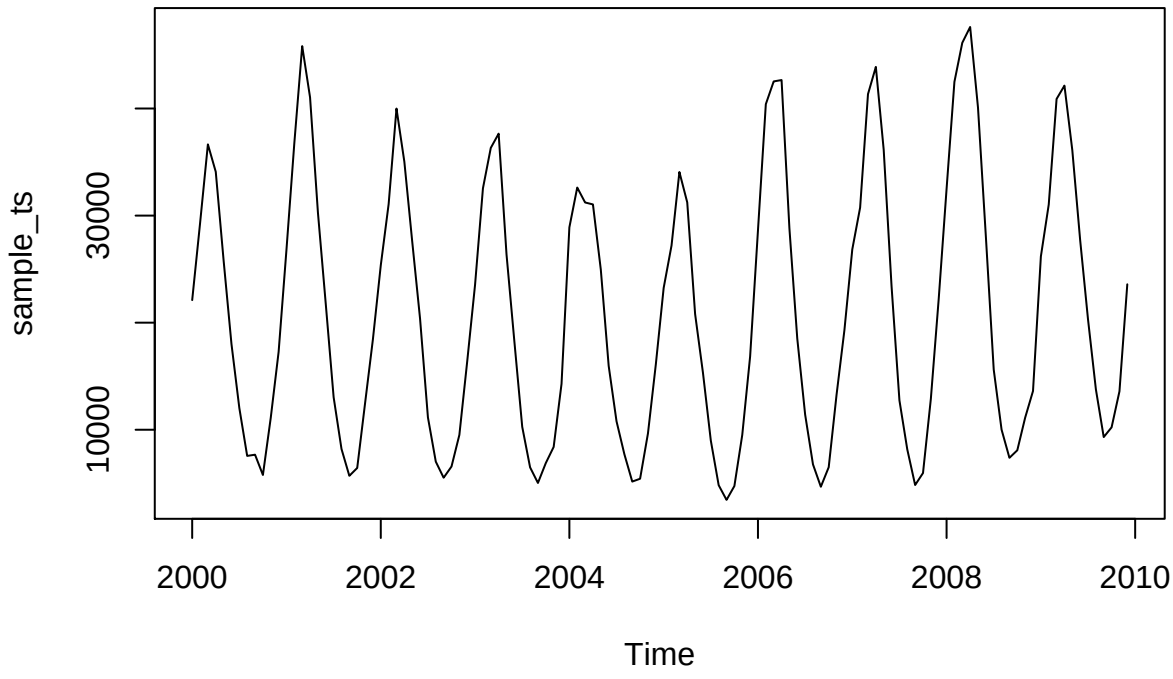
For all parts of the assignment prepare the data set such that the model consider only the data from January 2000 up to December 2009. Leave the year 2010 of data (January 2010 to December 2010) for the out-of-sample analysis. Do **NOT** use data fro 2010 and 2011 for model fitting. You will only use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

Part I: Preparing the data sets

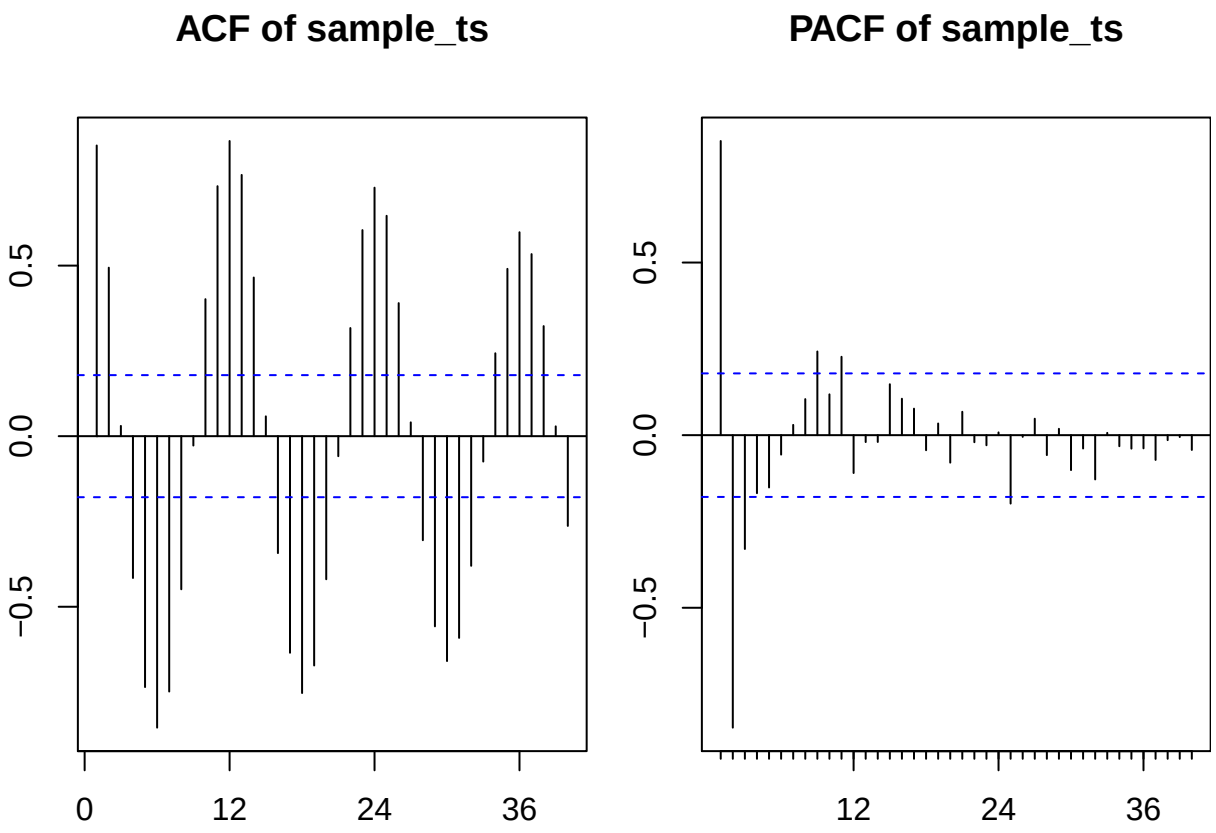
Q1

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.

```
#Read the file into a data frame
Data <- read.table(file="../Data/inflowtimeseries.txt",header=FALSE,skip=0)

#Select last column of the data set which represents one hydro plant in the Amazon river basin
inflow_data <- Data[,c(1:2,17)]

#Rename columns
colnames(inflow_data)=c("Month","Year", "MonthlyInflow")
#attach(inflow_data)

#Observations start in January 2000 and end in December 2009
sample<-inflow_data[which(inflow_data$Year>=2000 & inflow_data$Year<=2009),]
full<-inflow_data[which(inflow_data$Year>=2000 & inflow_data$Year<=2010),]

#Convert to time series
sample_ts<-ts(sample$MonthlyInflow, frequency = 12, start = c(2000,1))
full_ts<-ts(full$MonthlyInflow, frequency = 12, start = c(2000,1))


#Plot the time series
plot(sample_ts)
```



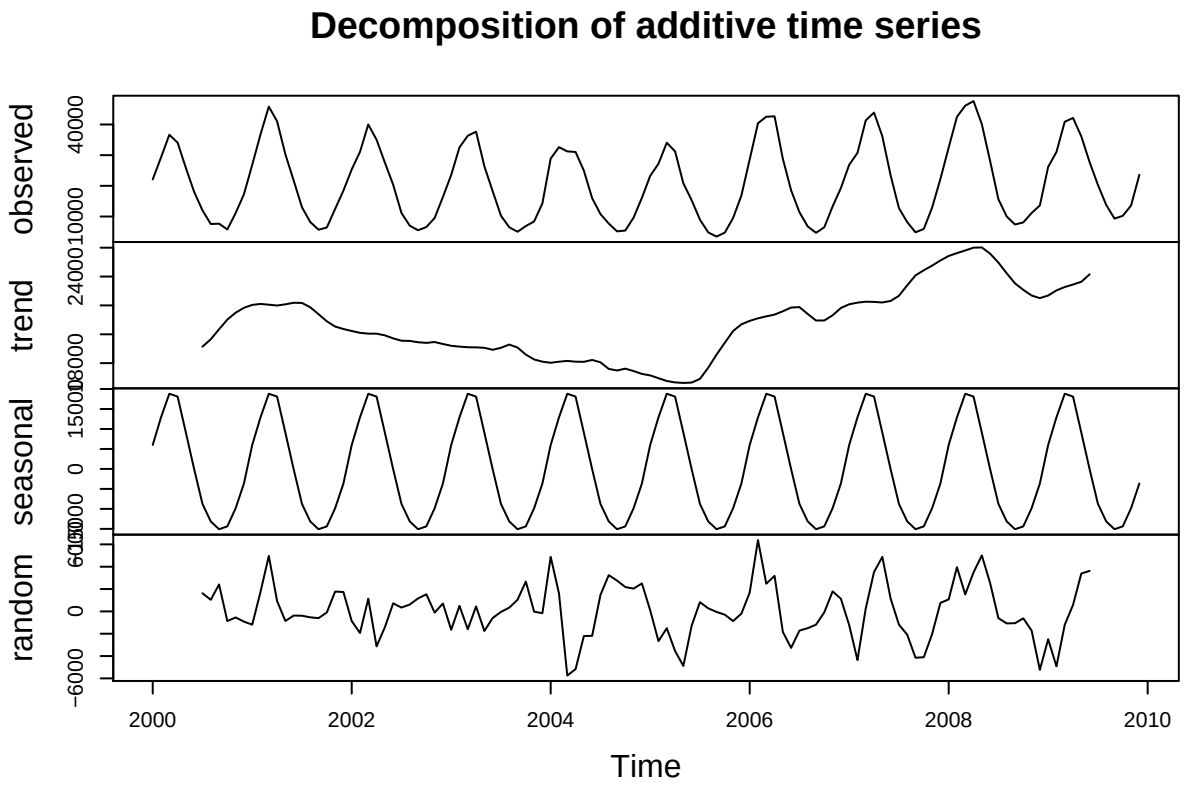

```
#ACF and PACF plots
par(mar=c(3,3,3,0));par(mfrow=c(1,2))
ACF_Plot <- Acf(sample_ts, lag = 40, plot = TRUE,main="ACF of sample_ts")
PACF_Plot <- Pacf(sample_ts, lag = 40, plot = TRUE,main="PACF of sample_ts")
```



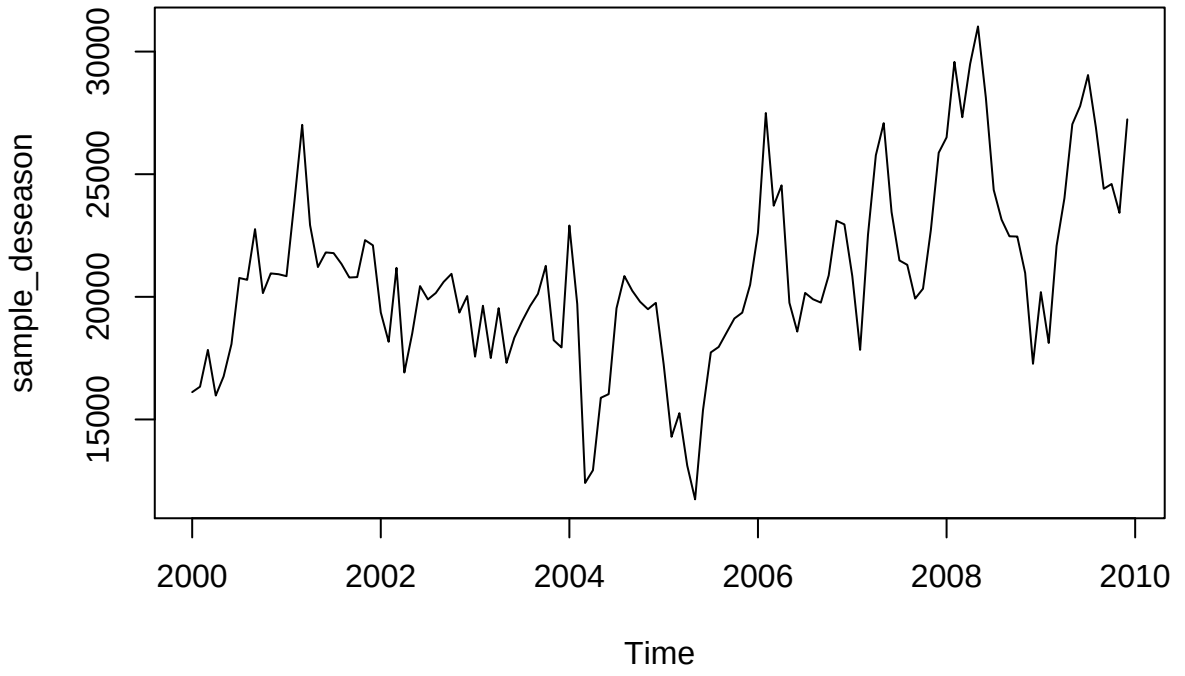
Q2

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.

```
# Using R decompose function
decompose_sample_ts <- decompose(sample_ts)
plot(decompose_sample_ts)
```

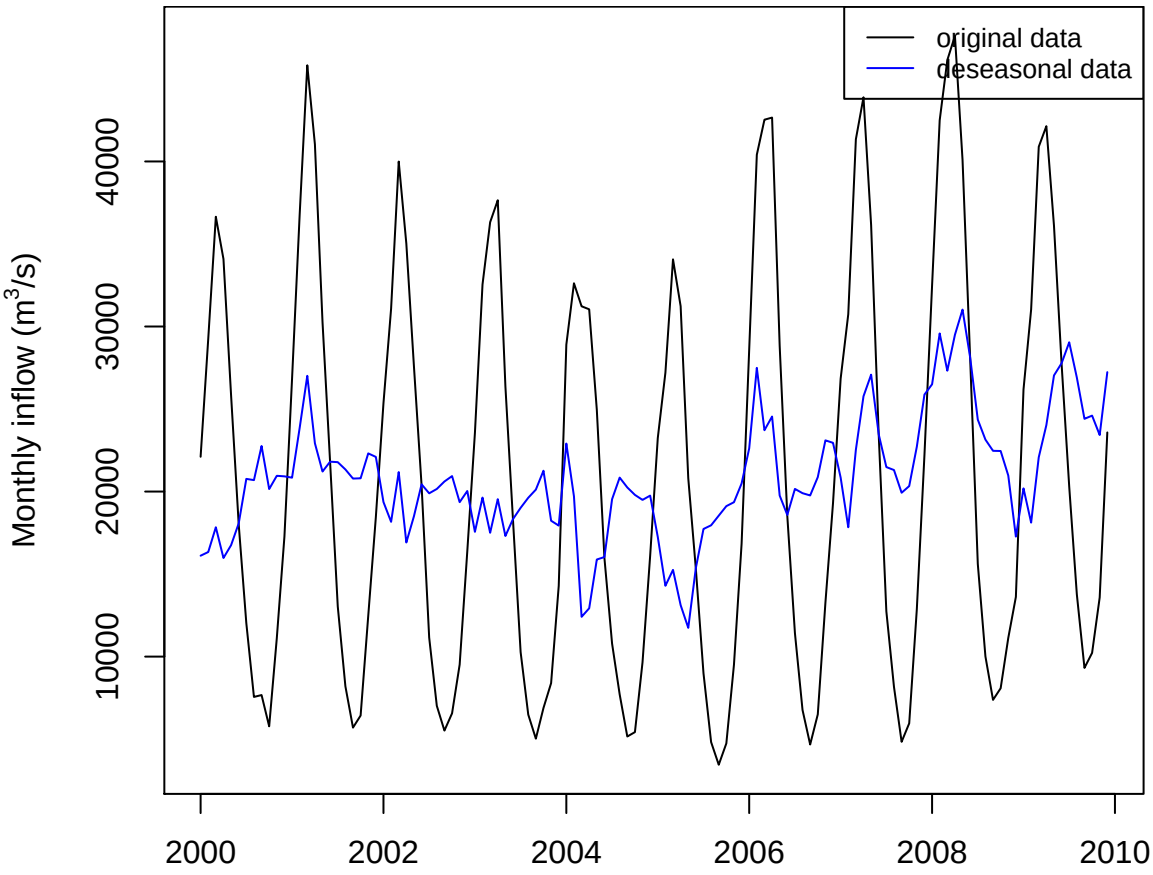



```
# Using R seasadj function
sample_deseason <- seasadj(decompose_sample_ts)
plot(sample_deseason)
```




```
# Make a basic graph
par(mar=c(2,5,0,2))
plot(sample_ts, type="l", xlab="Year",ylab=expression(paste("Monthly inflow (", m^3, "/","s)", sep="")))
lines(sample_deseason, col="blue")
```

```r
# Add a legend
legend("topright",
  legend = c("original data", "deseasonal data"),
  col = c("black", "blue"),
  lty=1:1,
  cex=0.8)
```

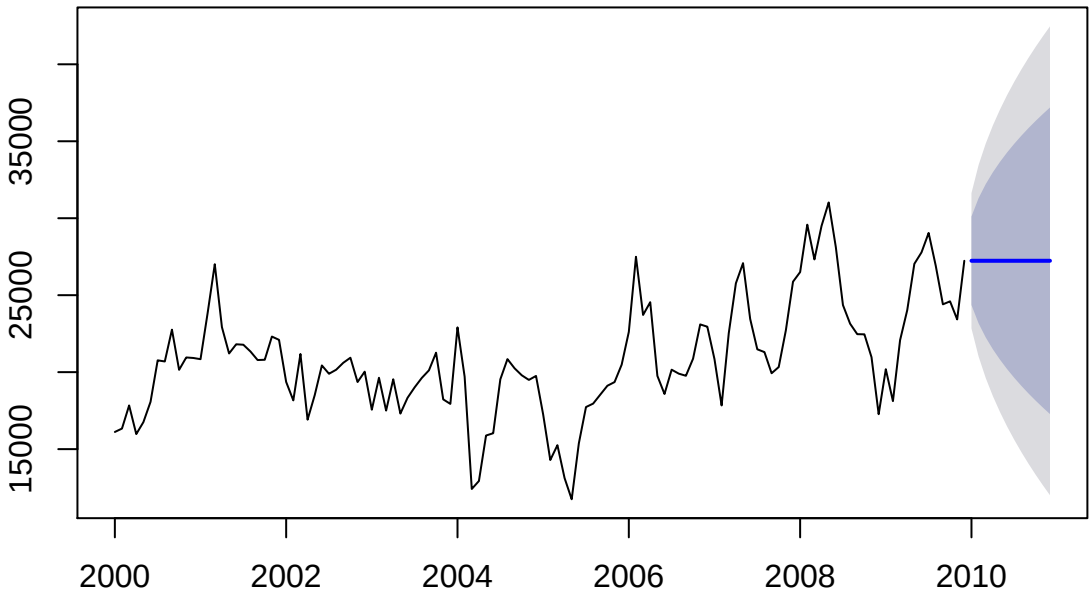


Part II: Forecasting with ARIMA models and its variations

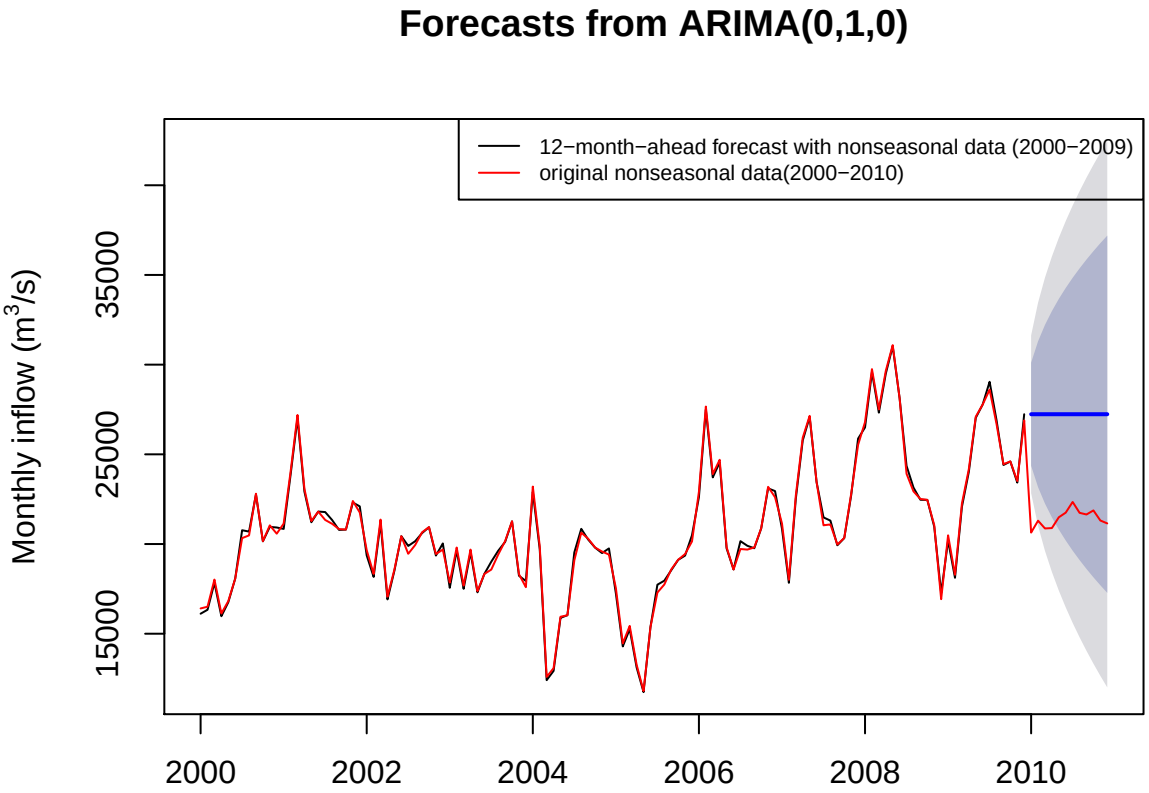
Q3

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).

```r
fit_sample_deseason <- auto.arima(sample_deseason)
#checkresiduals(fit_sample_deseason)

nonseas_ARIMA_forecast <- forecast(fit_sample_deseason, h=12)
plot(nonseas_ARIMA_forecast)
```




```
# Using R seasadj function on Full data 2000-2010
full_deseason <- seasadj(decompose(full_ts))

#Make new plot
par(mar=c(2,5,5,2))
plot(nonseas_ARIMA_forecast, type="l", xlab="Year",ylab=expression(paste("Monthly inflow (", m^3, "/","
lines(full_deseason, col="red")
legend("topright",
  legend=c("12-month-ahead forecast with nonseasonal data (2000-2009)","original nonseasonal data(2000-
  col = c("black", "red"),
  lty=1:1,
  cex=0.7)
```



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.

```
#Seasonal component from your decompose object
sample_seas<-decompose_sample_ts$seasonal
tail(sample_seas,12)
```

```
##               Jan          Feb          Mar          Apr          May
## 2009   5992.19367  12911.55478  18814.60571  18113.28164   9068.14275
##               Jun          Jul          Aug          Sep          Oct
## 2009    -65.63503  -8741.35262 -13136.25540 -15089.91744 -14374.92670
##               Nov          Dec
## 2009  -9839.24151  -3652.44985
```

```
#Now number
obs <- nrow(sample)

#check last year data
sample_seas[(obs-11):obs]
```

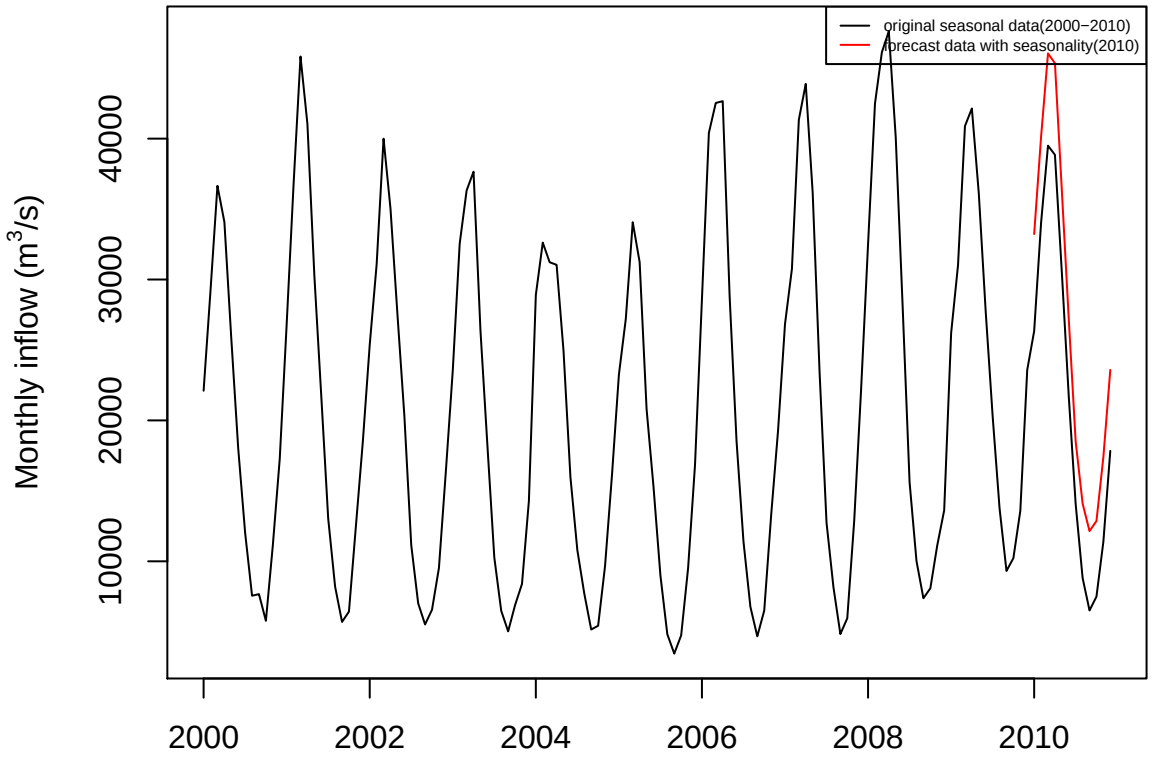
```
##  [1]   5992.19367  12911.55478  18814.60571  18113.28164   9068.14275
##  [6]    -65.63503  -8741.35262 -13136.25540 -15089.91744 -14374.92670
## [11]  -9839.24151  -3652.44985
```

```
forecasted_seasonality <-nonseas_ARIMA_forecast$mean+sample_seas[(obs-11):obs]
forecasted_seasonality
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
```

```
## 2010 33227.64 40147.00 46050.06 45348.73 36303.59 27169.81 18494.10 14099.19
##           Sep      Oct      Nov      Dec
## 2010 12145.53 12860.52 17396.21 23583.00
```

```
#plot(forecasted_seasonality)

#Make new plot
par(mar=c(2,5,3,2))
plot(full_ts, type="l", xlab="Year",ylab=expression(paste("Monthly inflow (", m^3, "/","s)", sep="")))
lines(forecasted_seasonality, col="red")
legend("topright",
  legend=c("original seasonal data(2000-2010)","forecast data with seasonality(2010)"),
  col = c("black", "red"),
  lty=1:1,
  cex=0.5)
```



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().

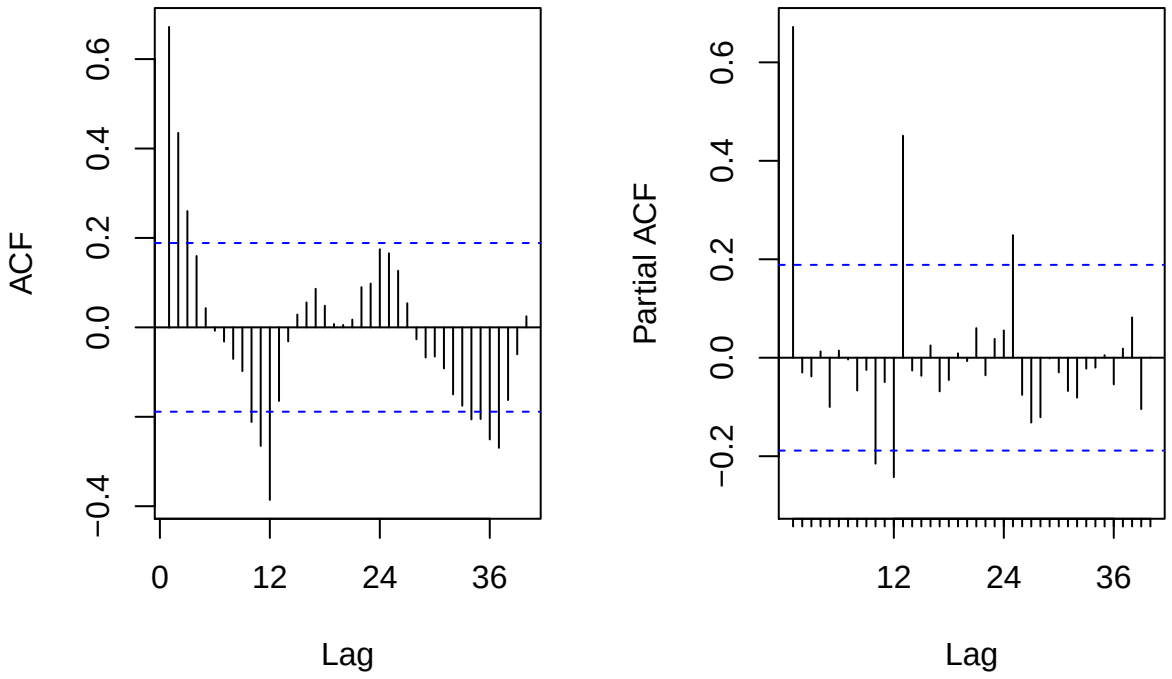
```
#Use the ndiffs() and nsdiffs()to find out. Always start with nsdiffs().
#nsdiffs estimates the number of seasonal differences necessary.
nsdiffs(sample_ts)
```

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

```
## output: 1

diff_sample_ts <- diff(sample_ts, differences = 1, lag = 12)
ndiffs(diff_sample_ts)
```

```
## [1] 0
```

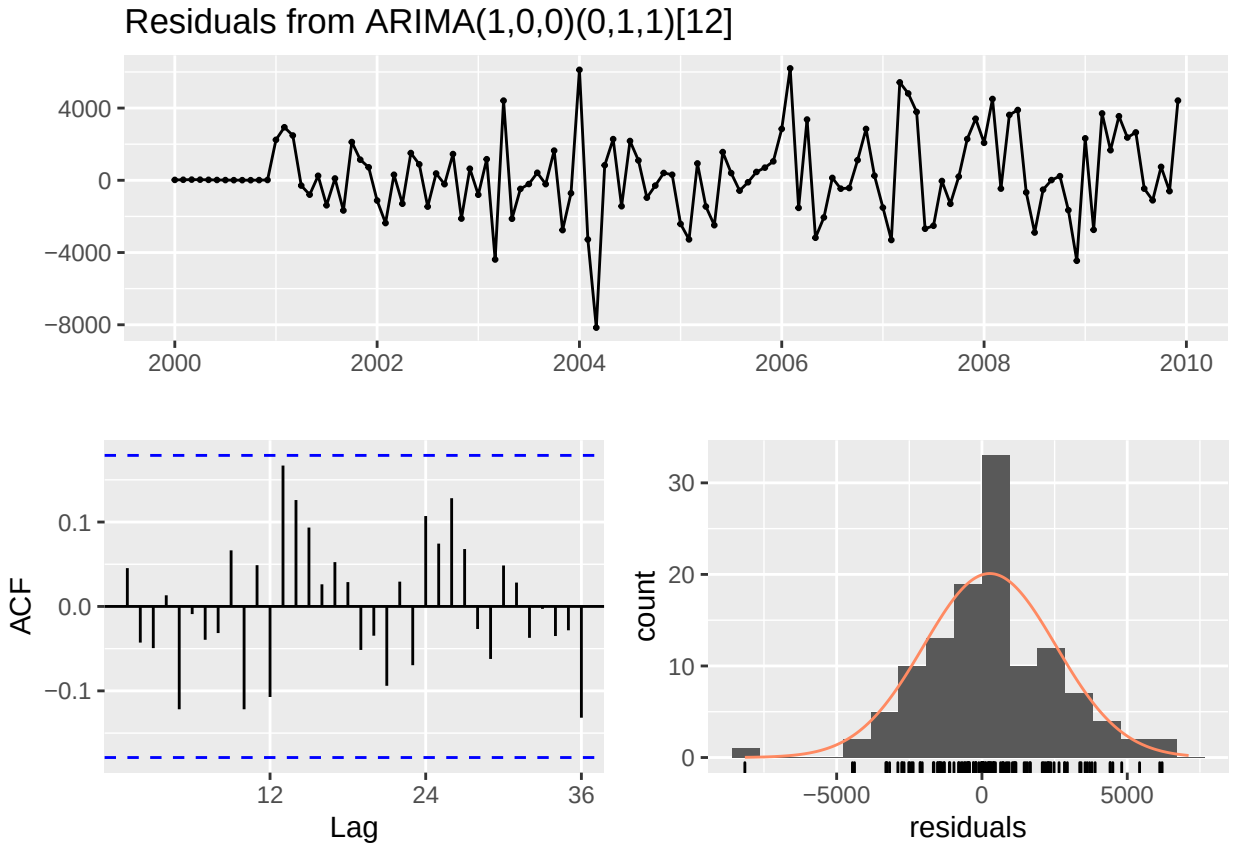
```
## output: 0
#We got D=1 and d=0

#Need to look at ACF/PACF of the differenced series to find order of the model
par(mfrow=c(1,2))
ACF_Plot <- Acf(diff_sample_ts, lag = 40, plot = TRUE,main="")
PACF_Plot <- Pacf(diff_sample_ts, lag = 40, plot = TRUE,main="")
```




```
# the seasonal component inverted. Now multiple spikes on PACF and only 1 on ACF, meaning we have a SMA
# And for the non-seasonal lags we see positive values at lag1, meaning we could have a AR model. But t

#Using Arima
fit_sample_original <- Arima(sample_ts, order=c(1,0,0), seasonal=list(order=c(0,1,1),include.constant=T
print(fit_sample_original)
```

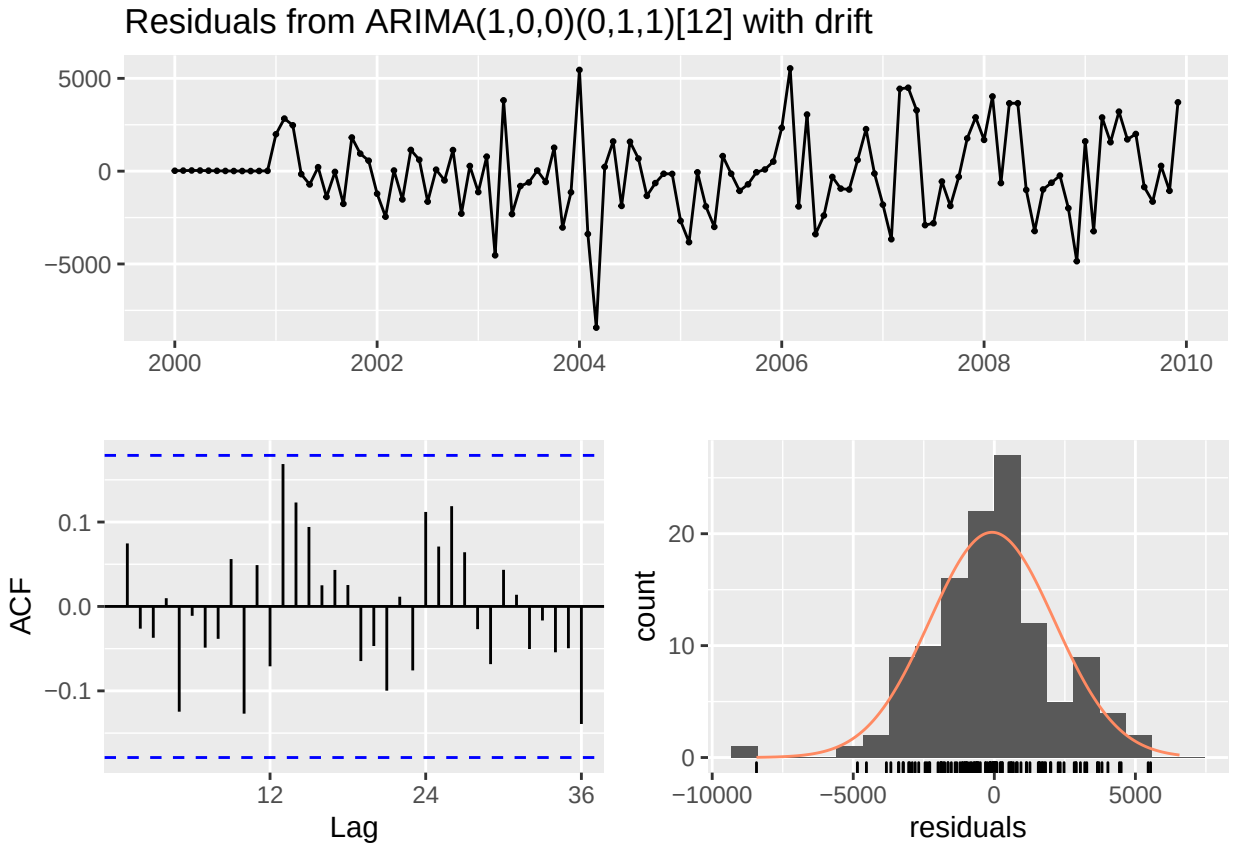
```
## Series: sample_ts
## ARIMA(1,0,0)(0,1,1)[12]
##
## Coefficients:
##          ar1     sma1
##       0.8136  -0.7989
## s.e.  0.0604   0.1211
##
## sigma^2 estimated as 5919121:  log likelihood=-1000.8
## AIC=2007.6   AICc=2007.83   BIC=2015.65
```

```
checkresiduals(fit_sample_original)
```



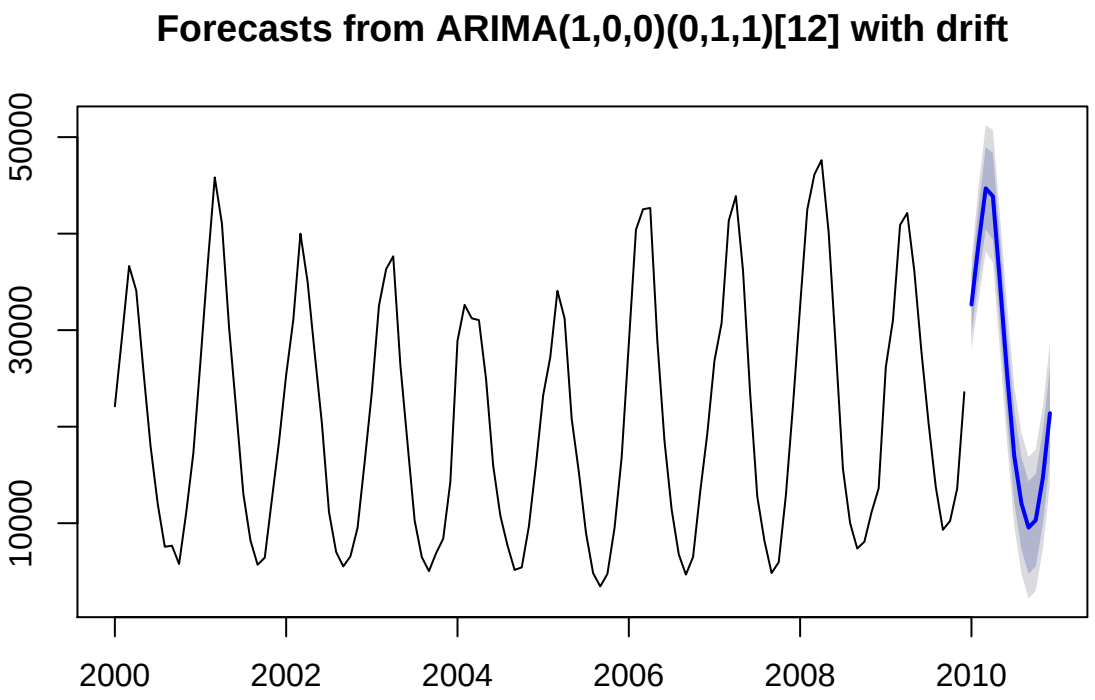

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(0,1,1)[12]
## Q* = 19.781, df = 22, p-value = 0.5968
##
## Model df: 2.   Total lags used: 24
```

```
#Using auto.arima
SARIMA_autofit <- auto.arima(sample_ts)
checkresiduals(SARIMA_autofit)
```

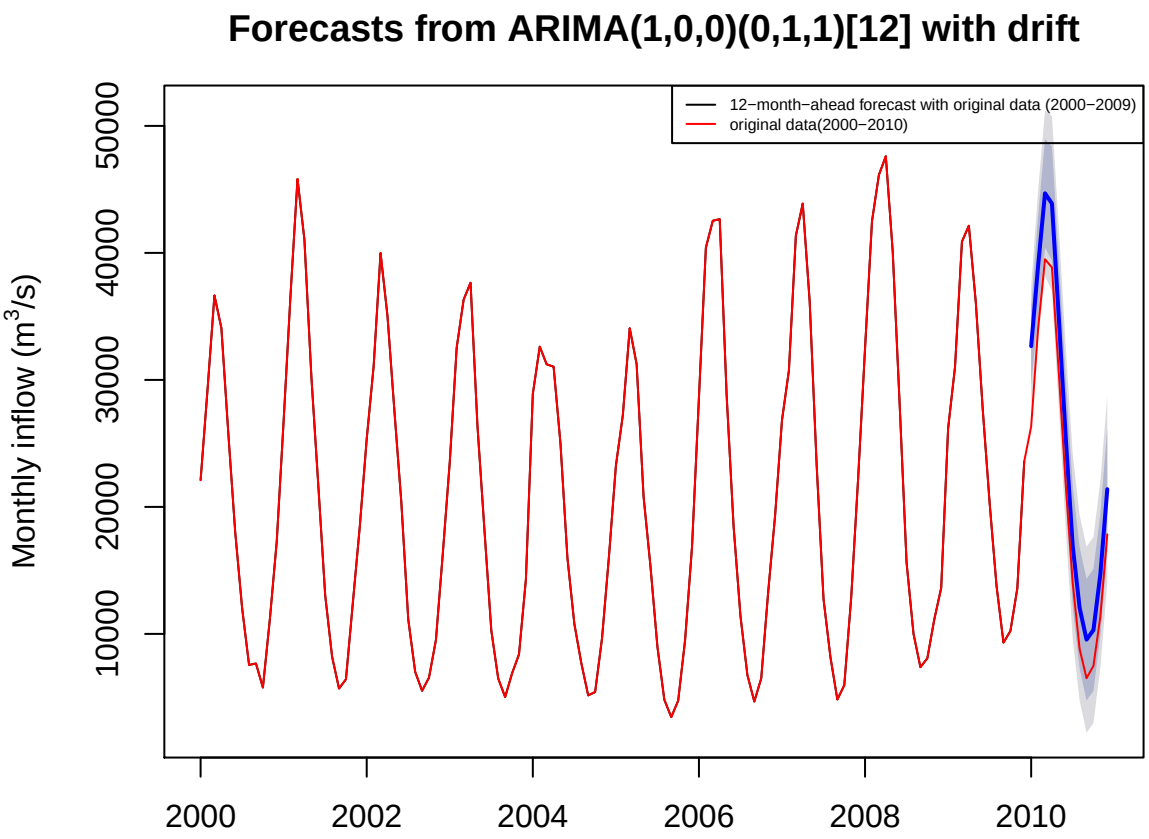



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(0,1,1)[12] with drift
## Q* = 19.881, df = 21, p-value = 0.5288
##
## Model df: 3.   Total lags used: 24
```

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




```
# Using R seasadj function on Full data 2000-2010
#full_deseason <- seasadj(decompose(full_ts))

par(mar=c(2,5,3,2))
plot(SARIMA_forecast, type="l", xlab="Year",ylab=expression(paste("Monthly inflow (", m^3, "/","s)", se
lines(full_ts, col="red")
legend("topright",
  legend=c("12-month-ahead forecast with original data (2000-2009)","original data(2000-2010)"),
  col = c("black", "red"),
  lty=1:1,
  cex=0.5)
```

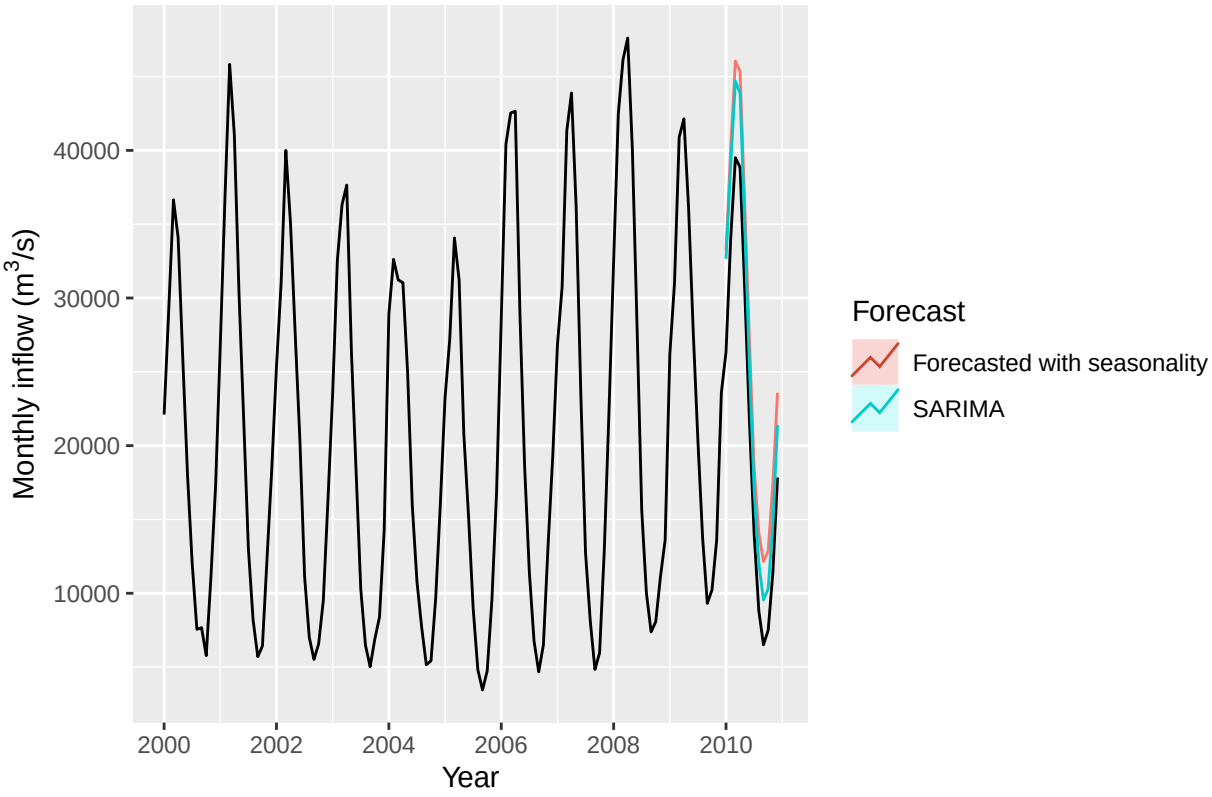


Q6

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

```
par(mar=c(2,5,3,2))
autoplot(full_ts) +
  autolayer(forecasted_seasonality, PI=FALSE, series="Forecasted with seasonality") +
  autolayer(SARIMA_forecast, PI=FALSE, series="SARIMA") +
  xlab("Year") +
  ylab(expression(paste("Monthly inflow (", m^3, "/","s)", sep=""))) +
  guides(colour=guide_legend(title="Forecast"))
```

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

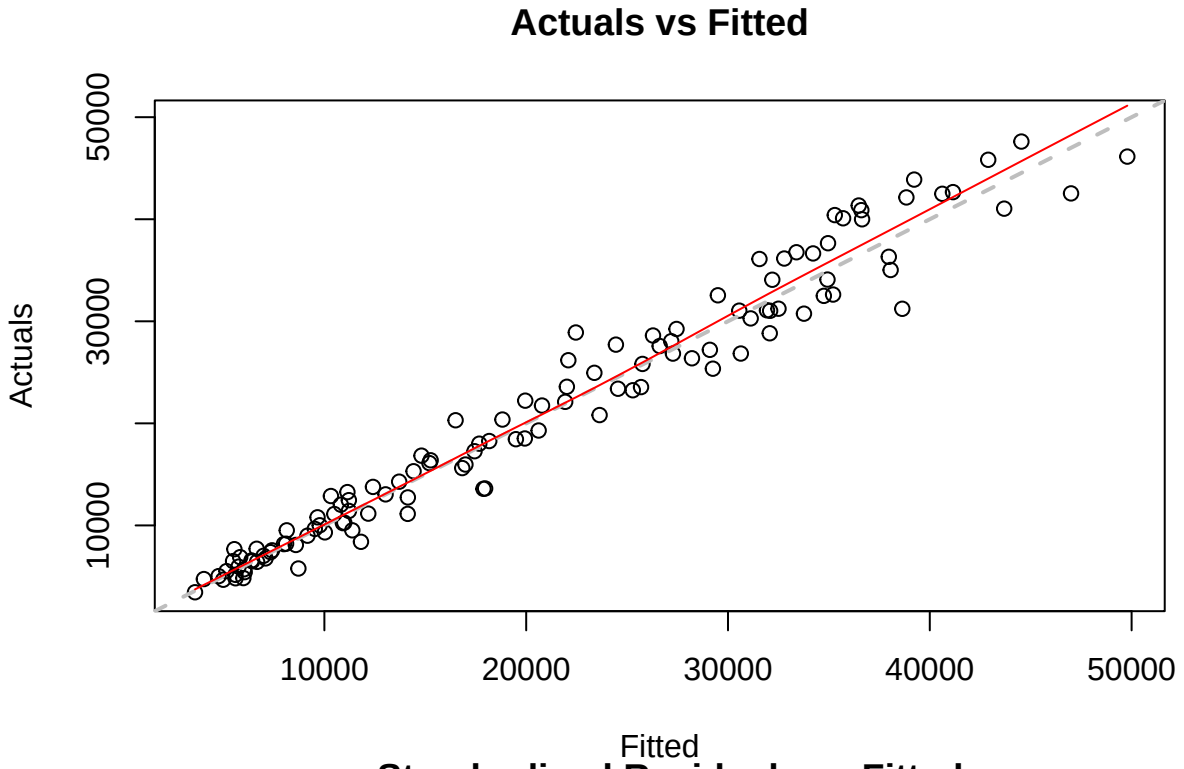


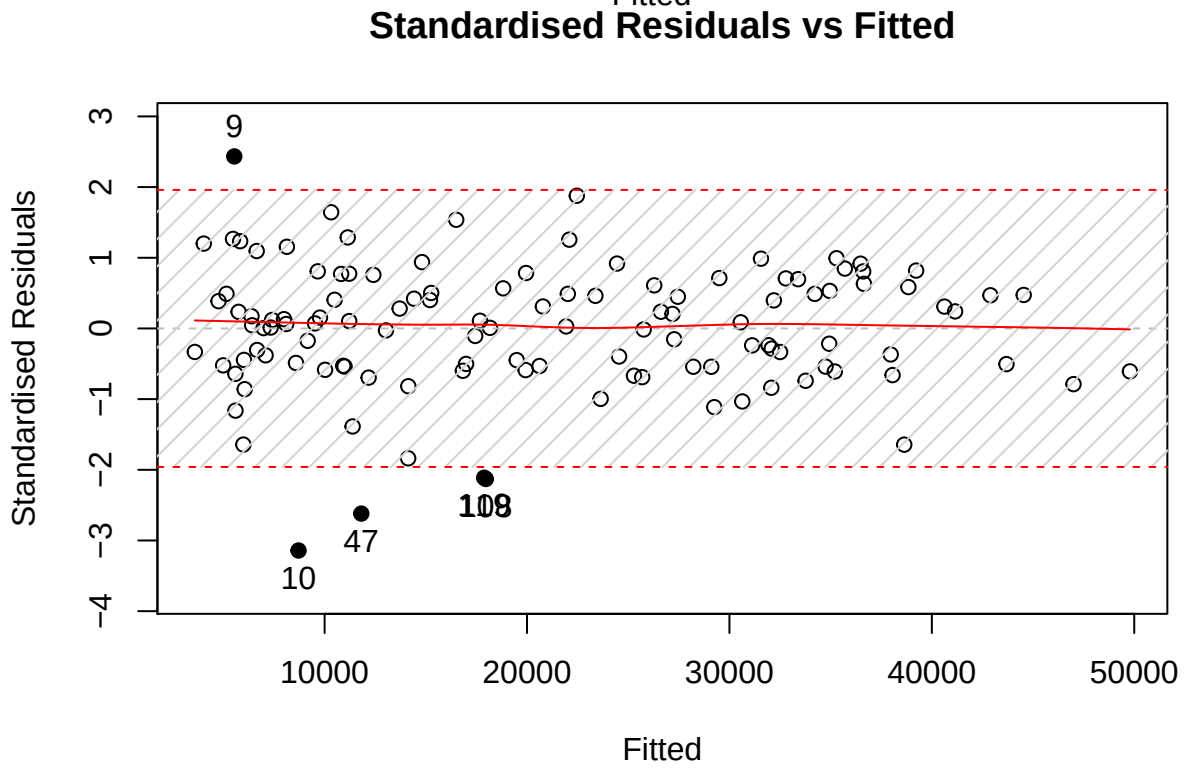
Part III: Forecasting with Other Models

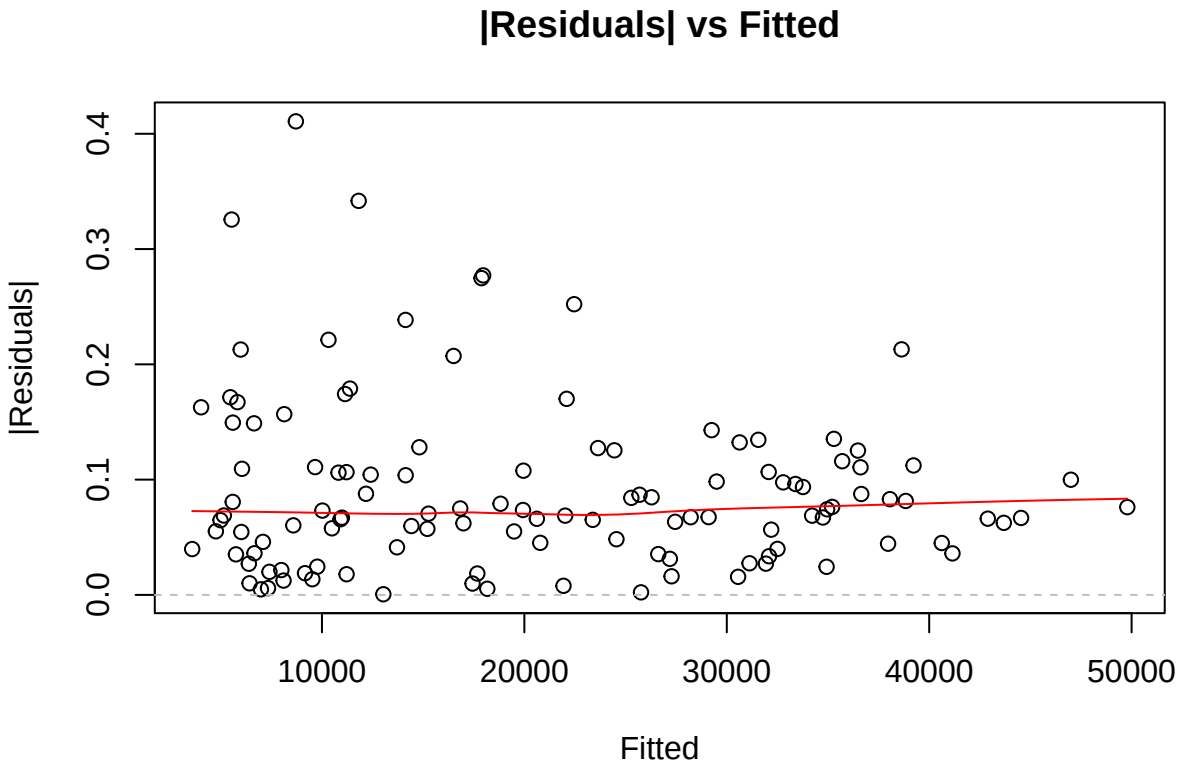
Q7

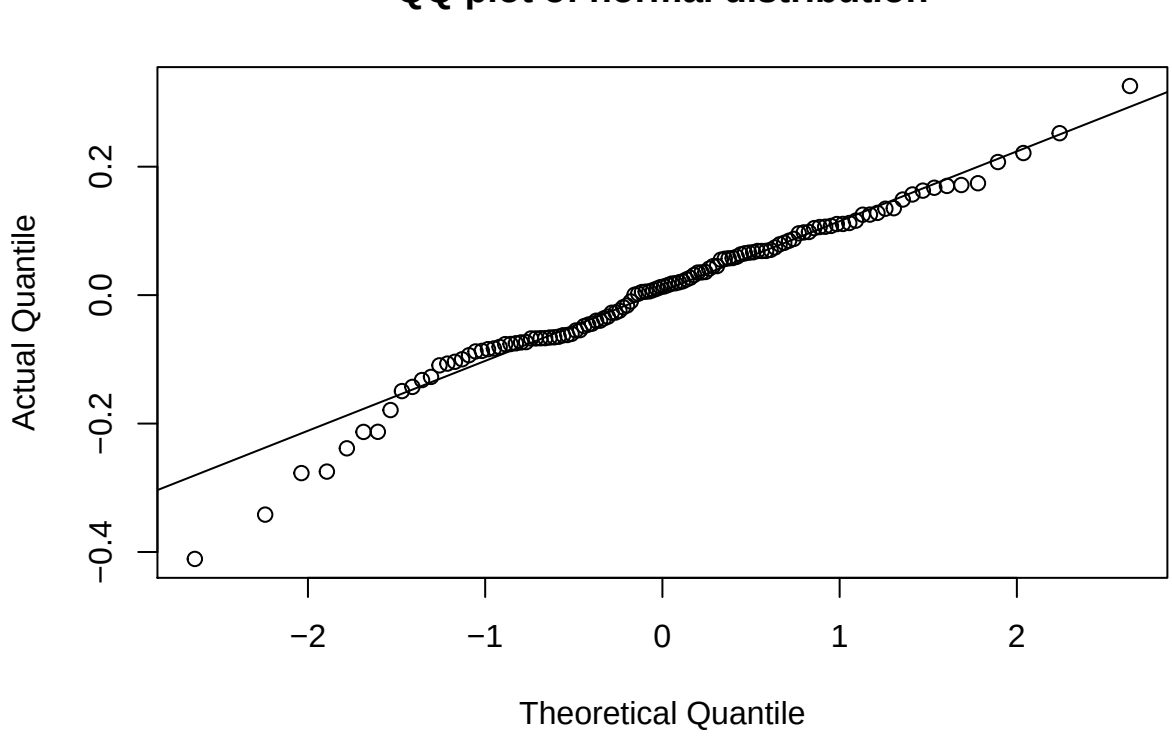
Fit an exponential smooth model to the original time series using the function *es*() from package **smooth**. Note that this function automatically do the forecast. Do not forget to set the arguments: silent=FALSE and holdout=FALSE, so that the plot is produced and the forecast is for the year of 2010.

```
SSES_seas <- es(sample_ts,model="ZZZ",h=12,holdout=FALSE)
plot(SSES_seas)
```


Actuals vs Fitted
Actuals
Fitted
10000
20000
30000
40000
50000

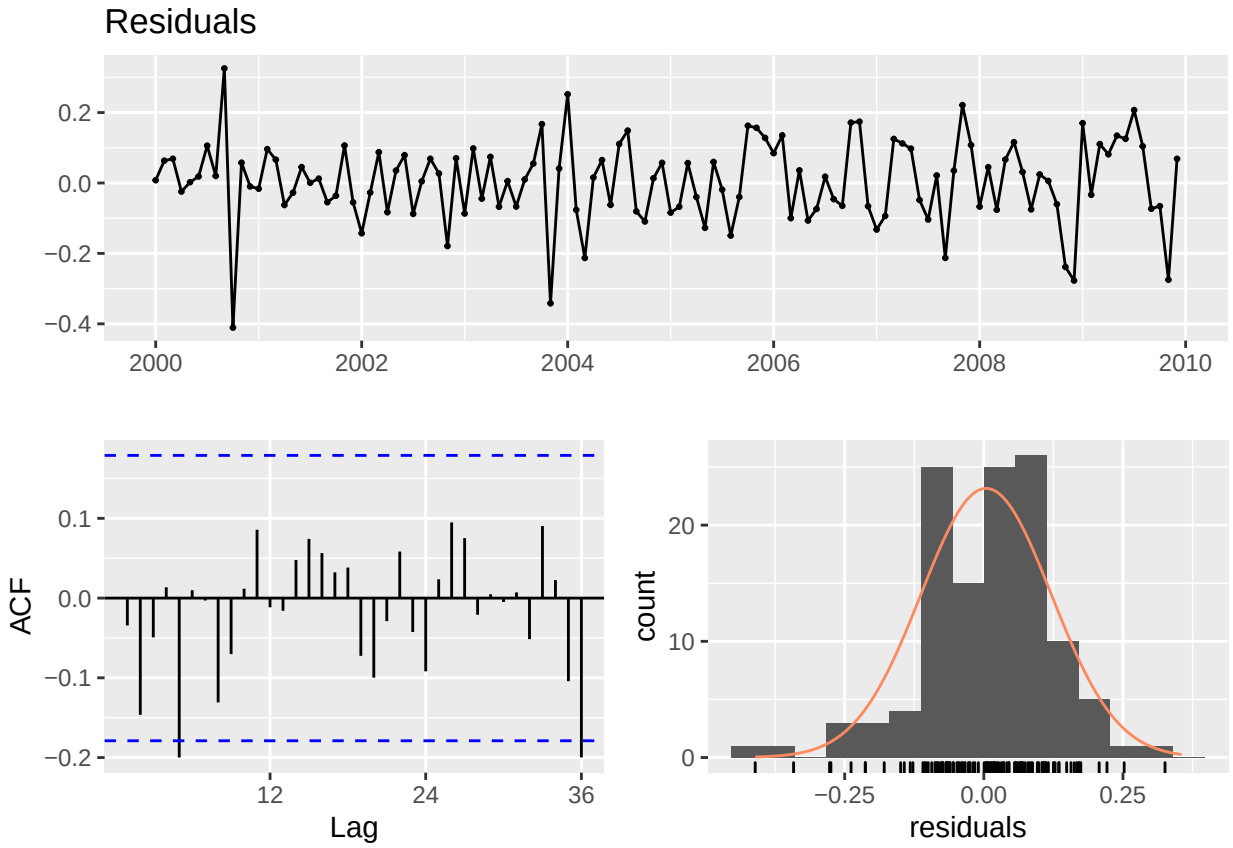

Standardised Residuals vs Fitted
Standardised Residuals
Fitted
9
47
10
108
−4
−3
−2
−1
0
1
2
3
10000
20000
30000
40000
50000






```
checkresiduals(SSES_seas)
```

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

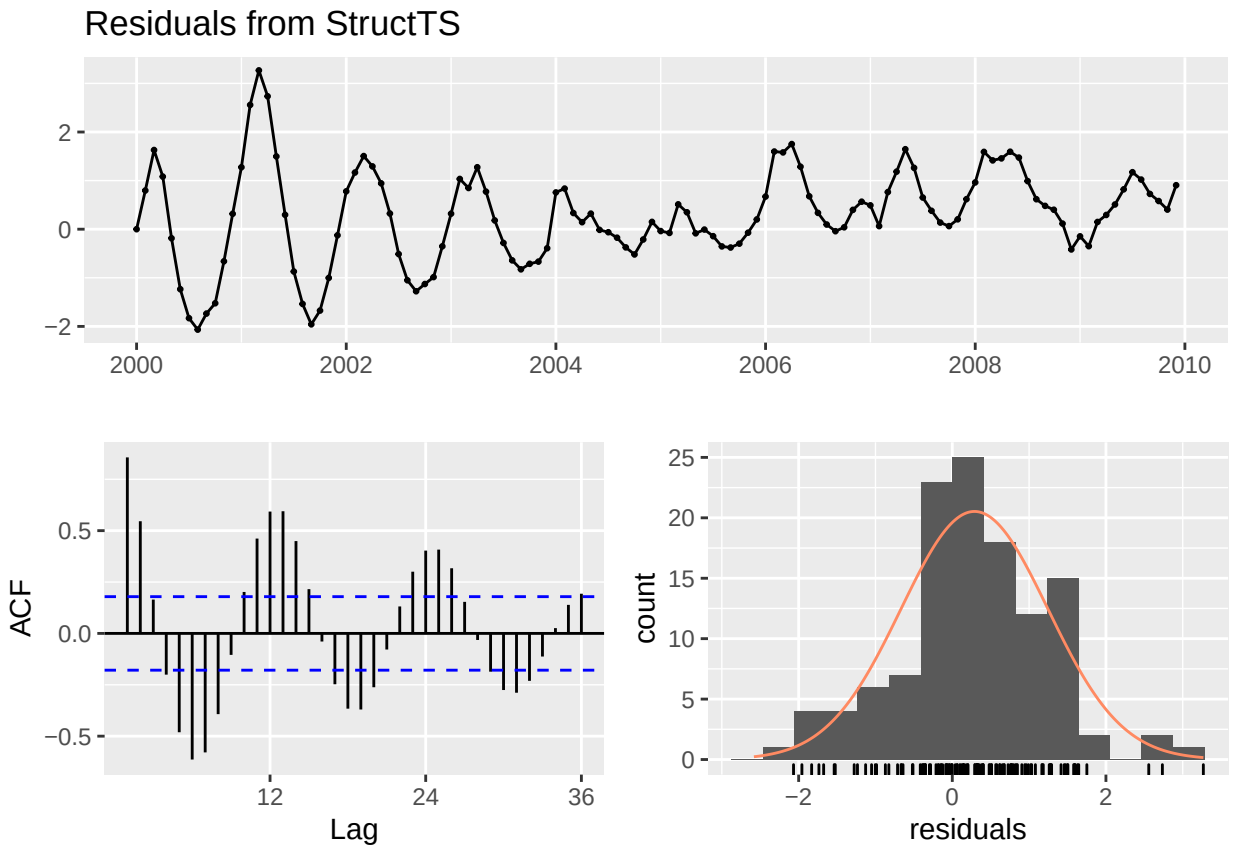


Q8

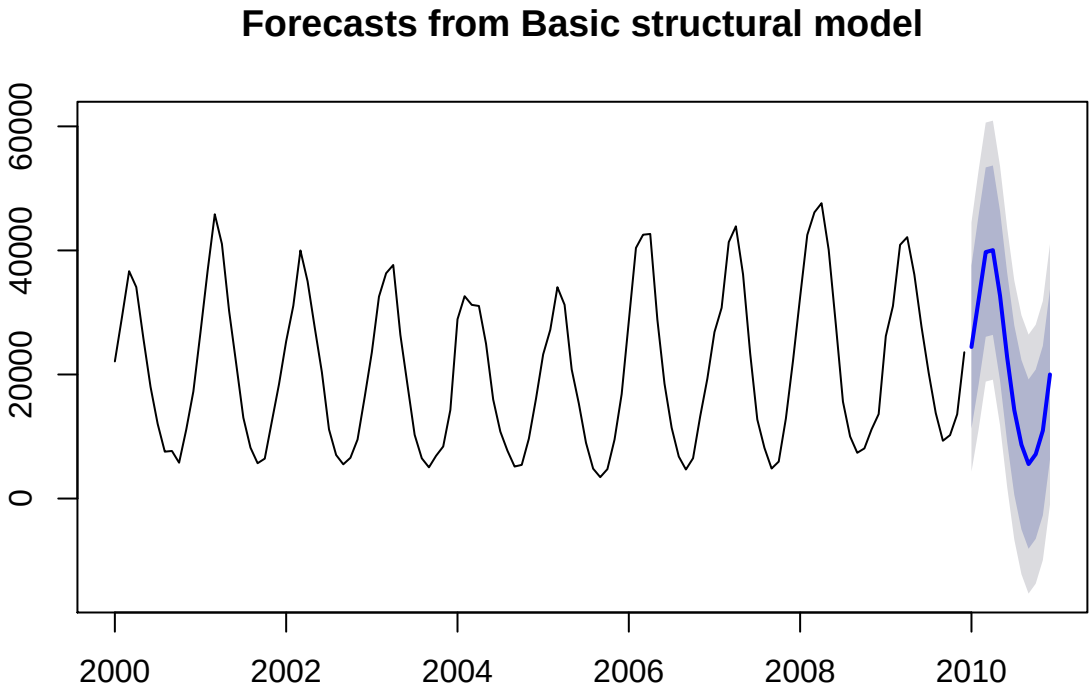
Fit a state space model to the original time series using the function $StructTS()$ from package `stats`. Which one of the tree model we learned should you try: "local", "trend", or "BSM". Why? Play with argument `fixed` a bit to try to understand how the different variances can affect the model. If 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.

```
SS_seas <- StructTS(sample_ts,
                    type="BSM",fixed=c(0.1, 0.001, NA, NA))   #this function has convergence issues
checkresiduals(SS_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
SS_forecast <- forecast(SS_seas,h=12)
plot(SS_forecast)
```



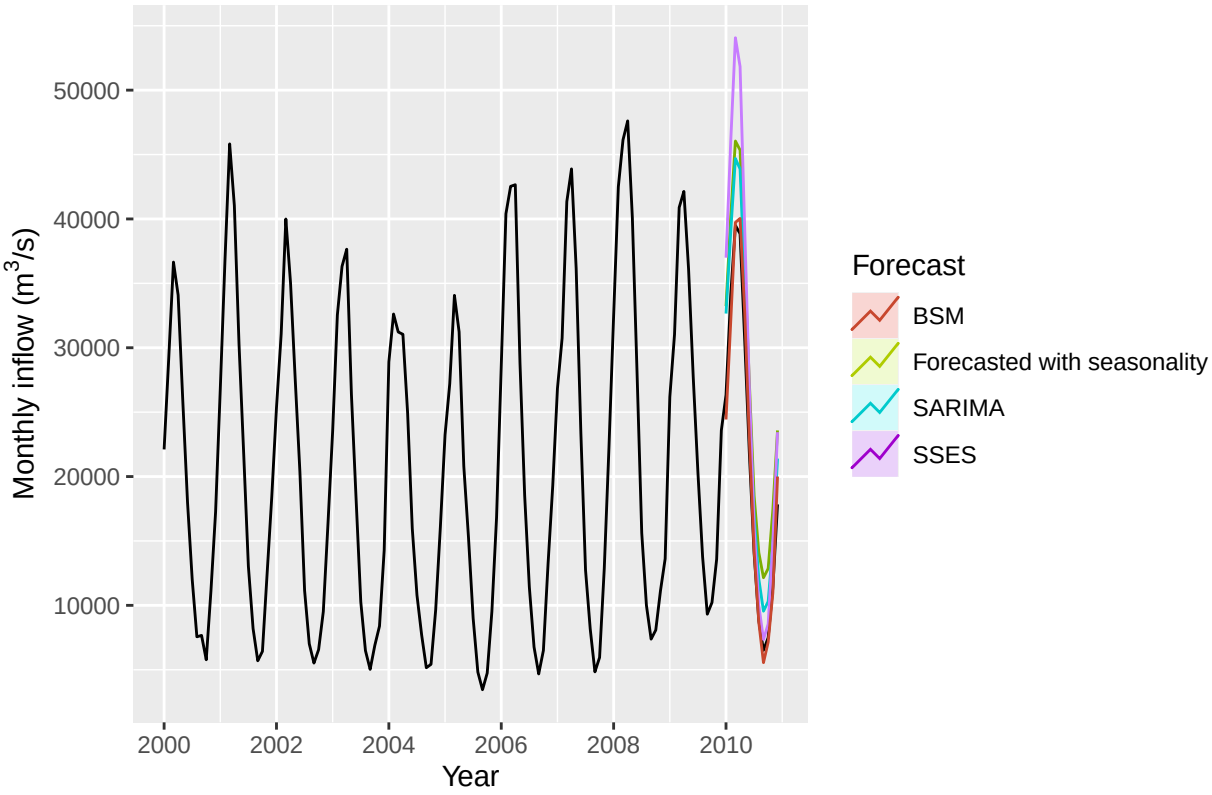
Part IV: Checking Forecast Accuracy

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(full_ts) +
  autolayer(forecasted_seasonality, PI=FALSE, series="Forecasted with seasonality") +
  autolayer(SARIMA_forecast, PI=FALSE, series="SARIMA") +
  autolayer(SSES_seas$forecast, series="SSES") +
  autolayer(SS_forecast,PI=FALSE,series="BSM") +
  xlab("Year") +
  ylab(expression(paste("Monthly inflow (", m^3, "/","s)", sep=""))) +
  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.

```
# BSM is leading to the better forecasts, because the BSM line is most overlapped with original data li
```

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?

```
#sample_ts<-ts(sample$MonthlyInflow, frequency = 12, start = c(2000,1))
#full_ts<-ts(full$MonthlyInflow, frequency = 12, start = c(2000,1))

nobs <- nrow(full)

last_obs <- full_ts[(nobs-11):nobs]
#no need to make obs_2020 a ts object

#If you not sure what we did here comment the following lines and check the values/dates
tail(full_ts,12)
```

```
##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
## 2010 26337 34045 39501 38851 30495 21687 14043  8815  6512  7492 11387 17839
```

```
last_obs
```

```
##  [1] 26337 34045 39501 38851 30495 21687 14043  8815  6512  7492 11387 17839
```

```
# Model 1:
ForecastedSeasonality_scores <- accuracy(forecasted_seasonality,last_obs)  #store the performance metri

# Model 2:
SARIMA_forecast_scores <- accuracy(SARIMA_forecast$mean,last_obs)

# Model 3:  SSES
SSES_scores <- accuracy(SSES_seas$forecast,last_obs)

# Model 4:  BSM
SS_scores <- accuracy(SS_forecast$mean,last_obs)
```

```
#create data frame
seas_scores <- as.data.frame(rbind(ForecastedSeasonality_scores,
                                   SARIMA_forecast_scores,
                                   SSES_scores,
                                   SS_scores))
row.names(seas_scores) <- c("Forecasted with seasonality", "SARIMA","SSES","BSM")

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

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

```
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
```

Table 1: Forecast Accuracy for Data

	ME	RMSE	MAE	MPE	MAPE
Forecasted with seasonality	-5818.4498	5852.520	5818.450	-38.0417	38.0417
SARIMA	-4031.8177	4171.259	4031.818	-23.8732	23.8732
SSES	-6502.7064	8098.586	6502.706	-26.2114	26.2114
BSM	**-106.3944**	**1332.997**	**1076.021**	**0.6930**	**5.6184**

```
##     group_rows
```

```
kbl(seas_scores,
    caption = "Forecast Accuracy for Data",
    digits = array(4,ncol(seas_scores))) %>%
  kable_styling(full_width = FALSE, position = "center") %>%
  #highlight model with lowest MAPE
  #kable_styling(latex_options="striped", stripe_index = which.min(seas_scores[,"MAPE"]))
  row_spec(best_model_index, bold=T, color="black", background = "yellow")
```