ENV 790.30 - Time Series Analysis for Energy Data | Spring 2023

Assignment 8 - Due date 03/27/23

Hugh Cipparone

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_A08_Sp22.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(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidyr)
library(ggplot2)
library(lubridate)
```

```
## Loading required package: timechange
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(forecast)
```

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

```
library(Kendall)
library(tseries)
library(outliers)
```

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.

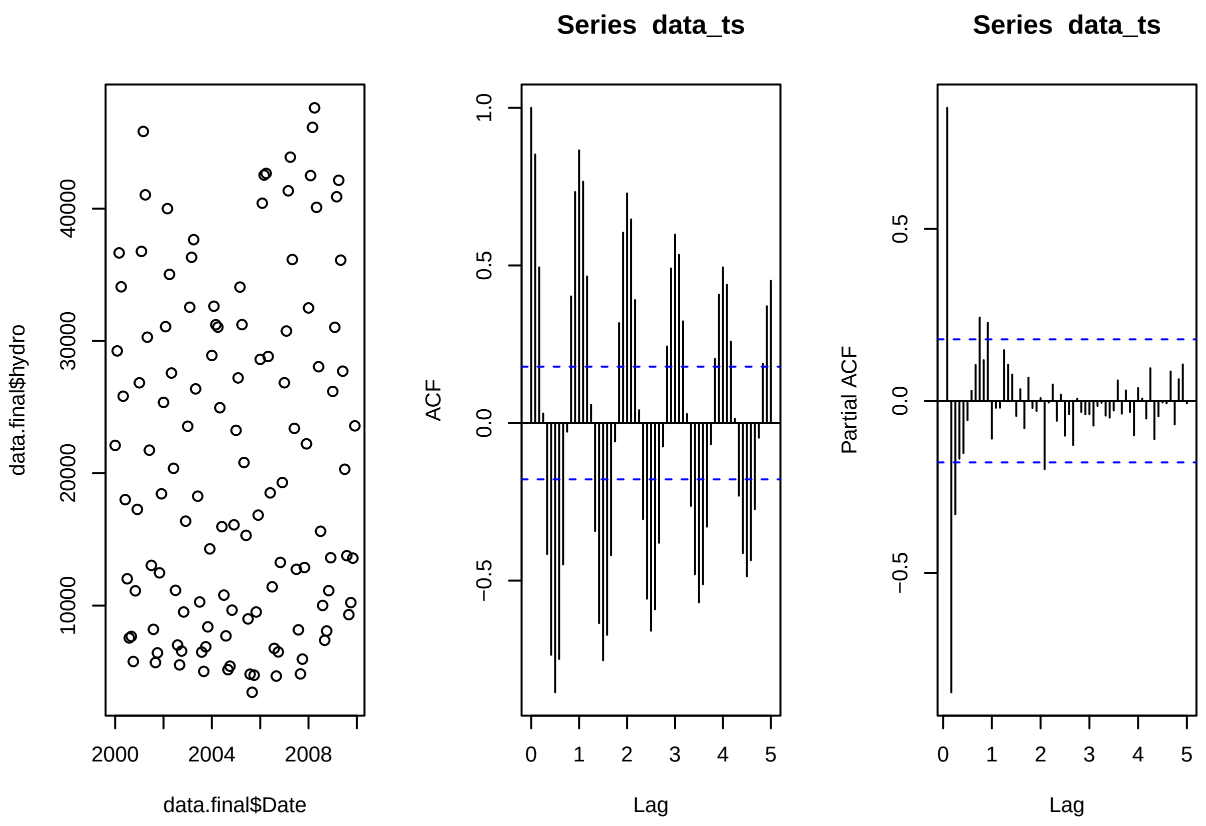
Part I: Preparing the data sets

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.

```
data<-read.delim("./Data/inflowtimeseries.txt", header = FALSE, sep = "", dec = ".")

data.edit<-data %>%
  unite("date", V1, V2, sep=" ", remove=TRUE)

Date<-my(data.edit$date)

data.final<-cbind(Date,data.edit) %>%
  select(Date,V17) %>%
  rename(hydro=V17) %>%
  filter(year(Date)<=2009 & year(Date)>=2000)


data_ts <- ts(data.final$hydro,start=c(2000,1),frequency = 12)
```

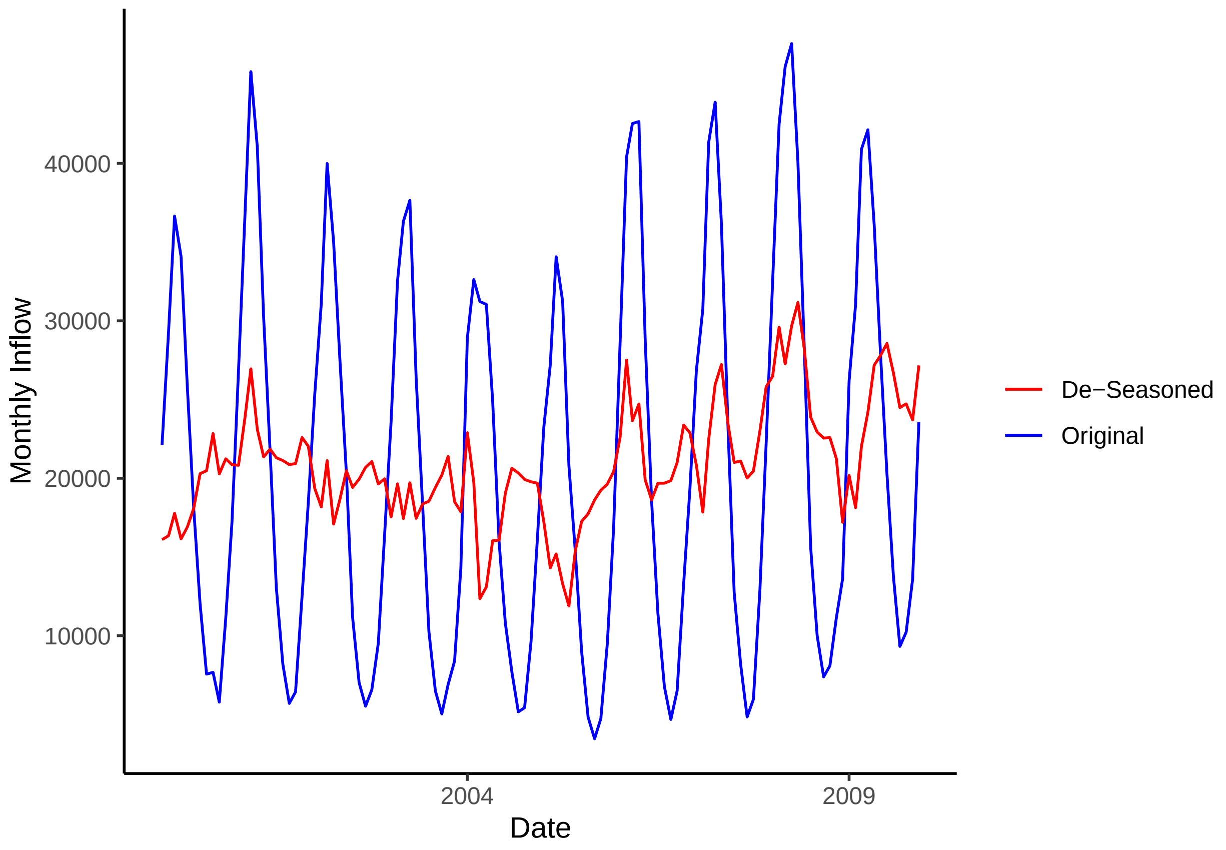
```
par(mfrow=c(1,3))

plot(data.final$Date, data.final$hydro)

acf(data_ts,lag.max=60, plot=TRUE)
pacf(data_ts,lag.max=60, plot=TRUE)
```



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.

```r
data_decomp <- stl(data_ts, s.window = "periodic")

data_decomp_deseas <- seasadj(data_decomp)

ggplot()+
  geom_line(aes(x=data.final$Date, y=data.final$hydro, color="Original"))+
  geom_line(aes(x=data.final$Date, y=data_decomp_deseas, color="De-Seasoned"))+
  labs(color="")+
  scale_color_manual(values = c("Original" = "blue", "De-Seasoned" = "red")) +
  theme(legend.position = "bottom") +
  ylab(label="Monthly Inflow") +
  theme_classic()+
  scale_x_date(date_breaks = "5 years", date_labels = "%Y")+
  xlab("Date")
```



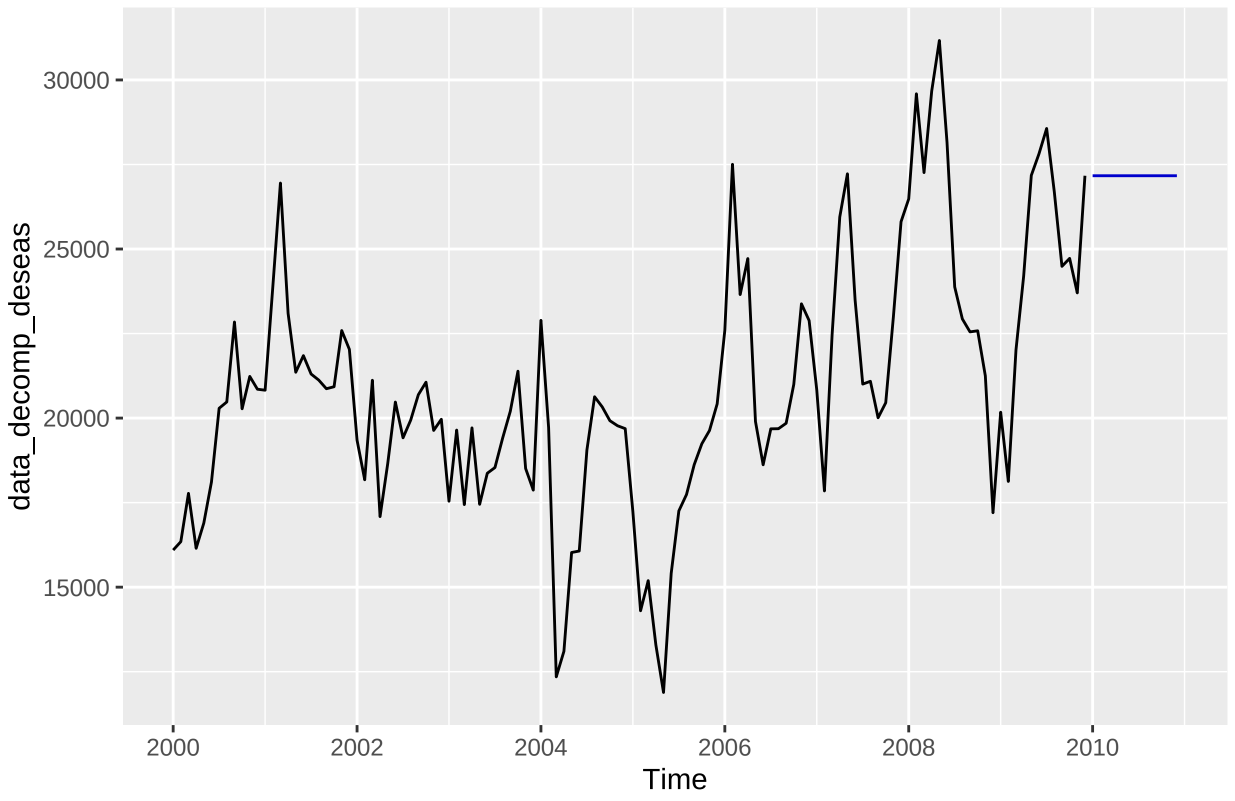
Answer: The deseasoned data looks much less variable! Less periodicity - less seasoned :)

Part II: Forecasting with ARIMA models and its variations

Q3

Fit a non-seasonal ARIMA(p, d, q) model using 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).

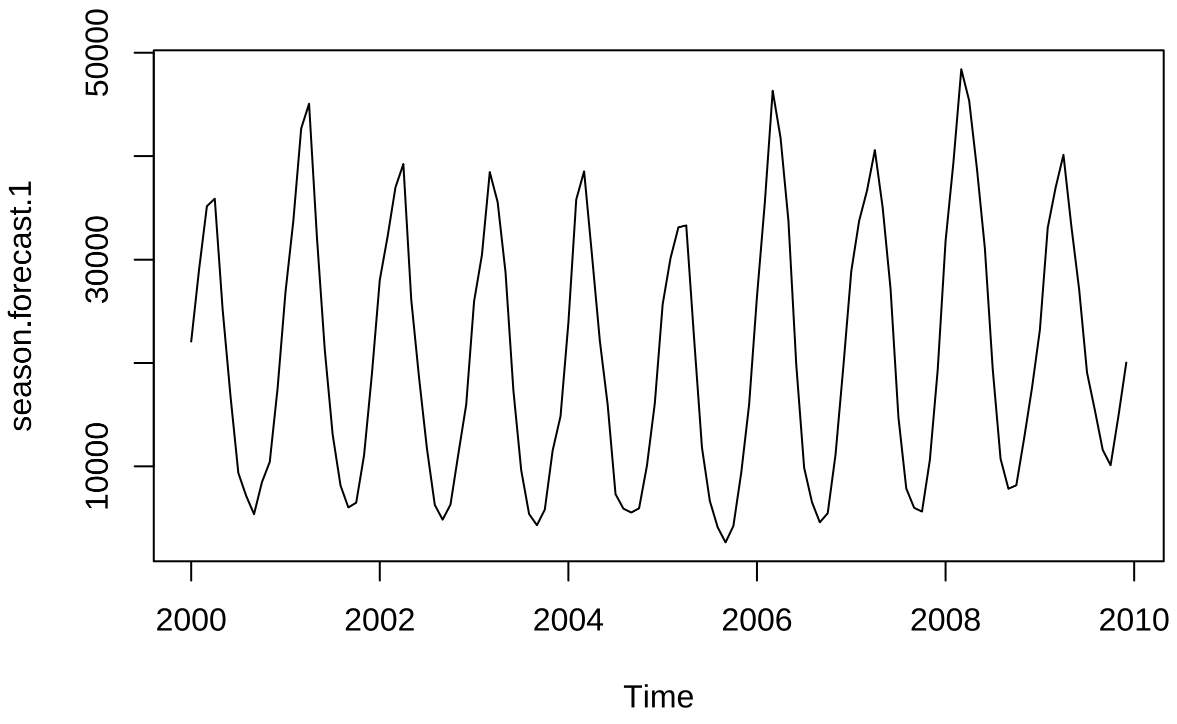
```
arima.1<-auto.arima(data_decomp_deseas)

arima.forecast.1 <- forecast(object = arima.1, h = 12)


autoplot(data_decomp_deseas) +
  autolayer(arima.forecast.1, PI=FALSE)
```



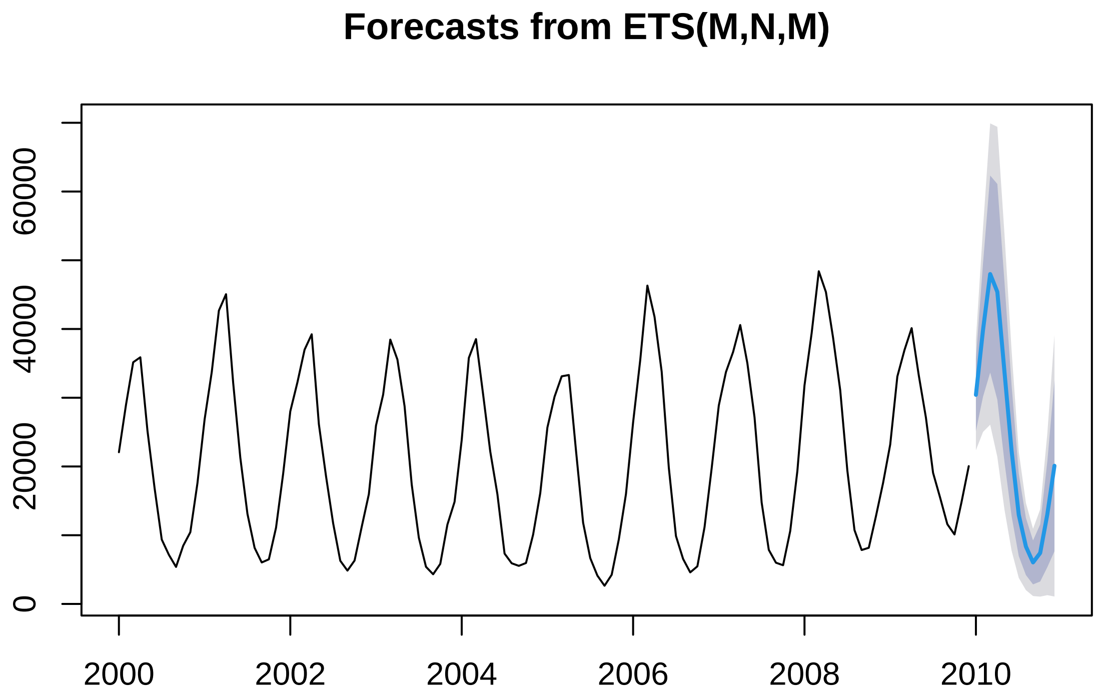
Q4

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.

```
decomp.data<-decompose(data_ts)

season.forecast<-data.frame(arima.forecast.1$fitted, decomp.data$seasonal)

season.forecast.1<-Reduce("+", season.forecast)

plot(season.forecast.1)
```



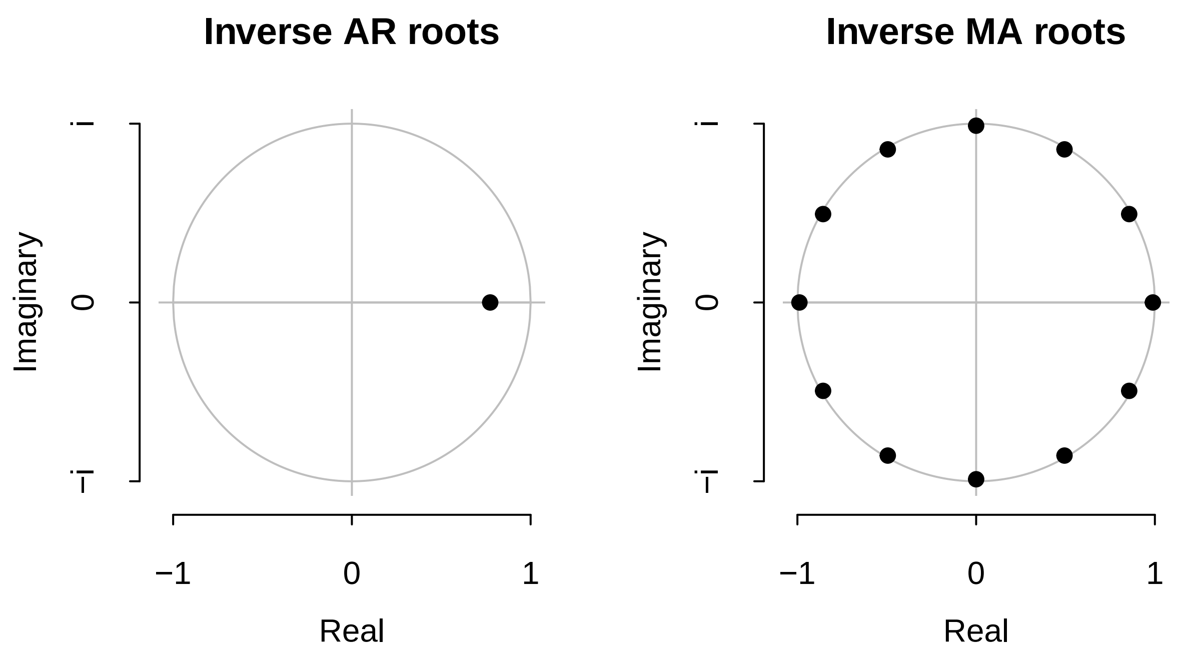
```
reseasoned.arima.forecast<-forecast(object=season.forecast.1, h=12)

plot(reseasoned.arima.forecast)
```



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().

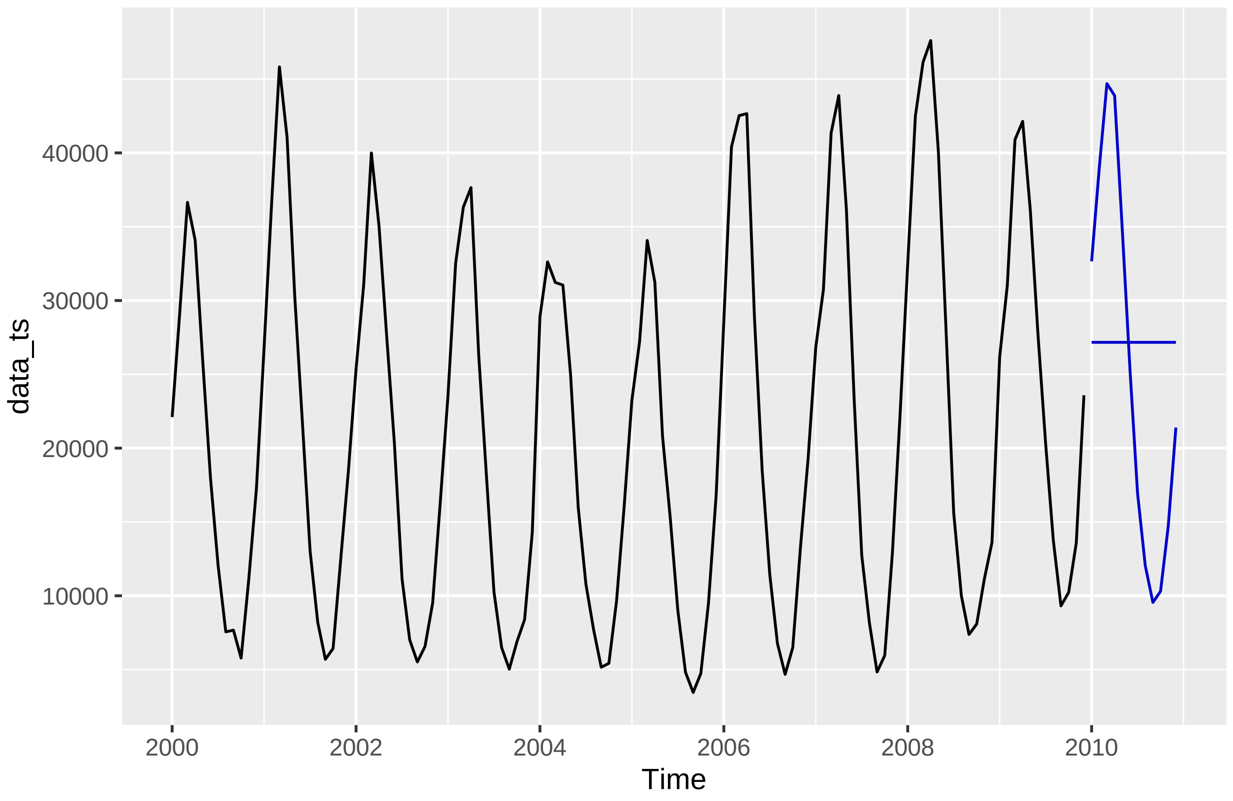
```
sarima.1<-auto.arima(data_ts)
sarima.1.forecast<-forecast(object=sarima.1, h=12)

plot(sarima.1)
```



Q6

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

```
autoplot(data_ts)+
  autolayer(arima.forecast.1, PI=FALSE)+
  autolayer(sarima.1.forecast, PI=FALSE)
```

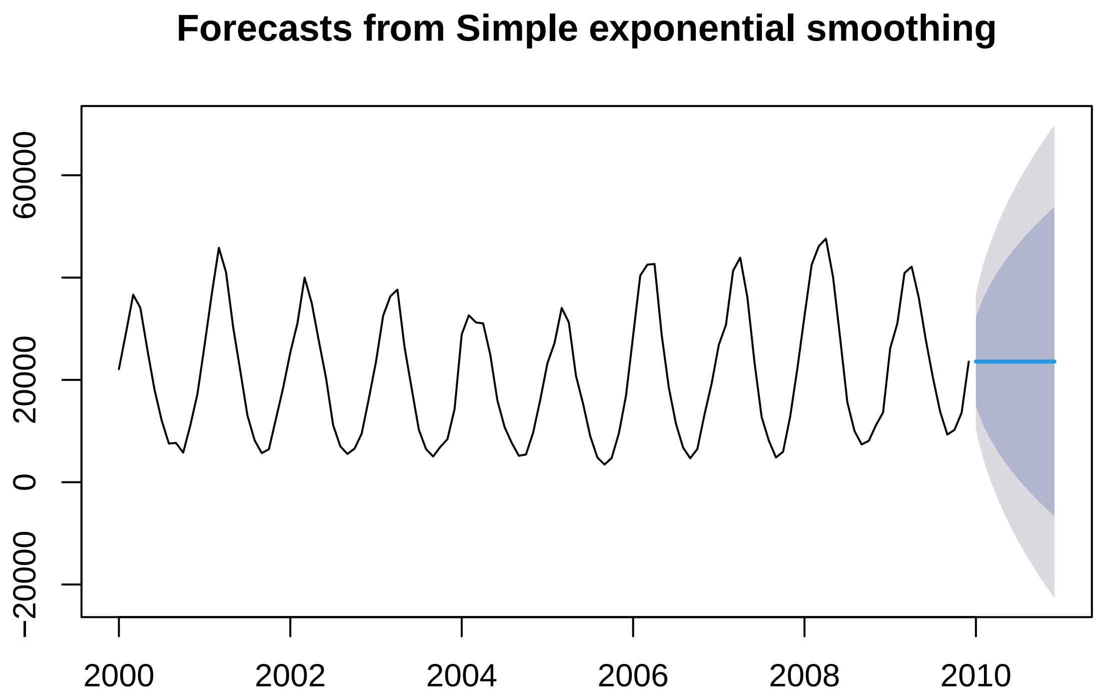


Part III: Forecasting with Other Models

Q7

Fit an exponential smooth model to the original time series using the function *ses*() from package `forecast`. 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.

```
SES_data=ses(y = data_ts, h = 12, holdout = FALSE, silent = FALSE)
plot(SES_data)
```

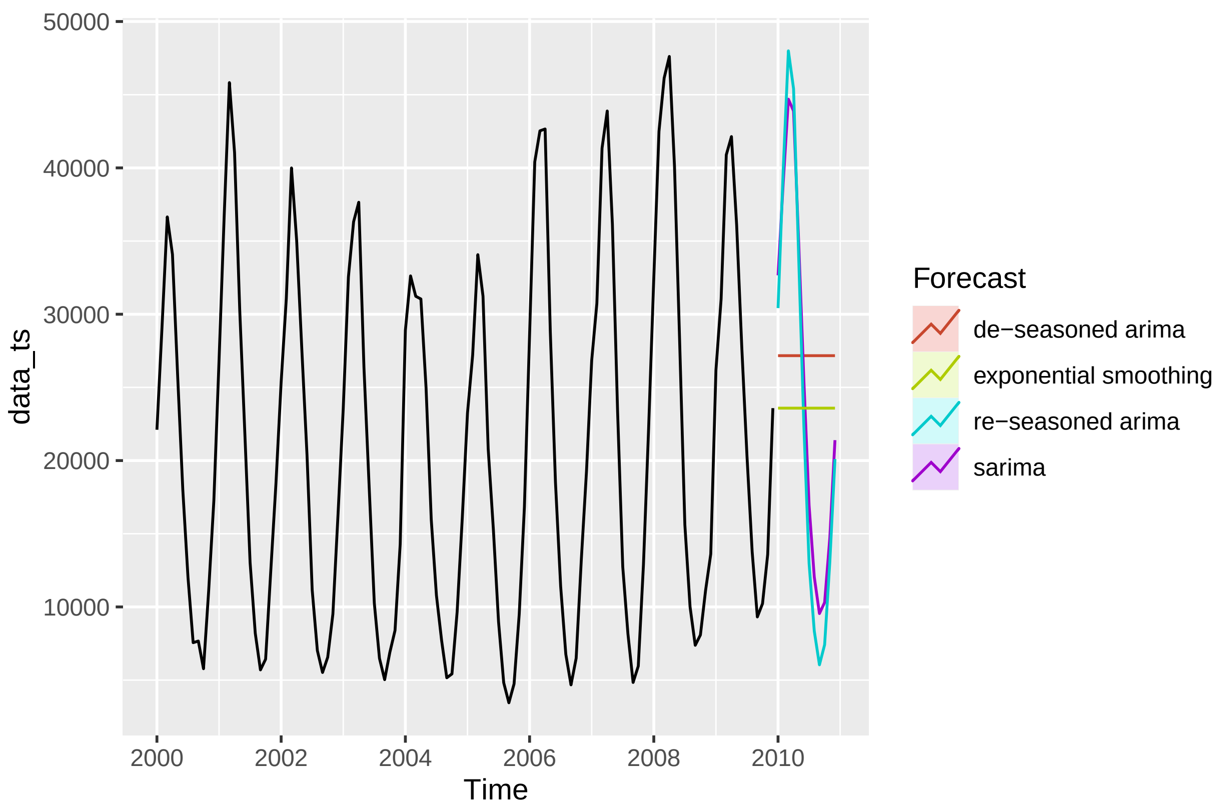


Part IV: Checking Forecast Accuracy

Q8

Make one plot with the complete original seasonal historical data (Jan 2000 to Dec 2009). 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(data_ts)+
  autolayer(arima.forecast.1, PI=FALSE, series="de-seasoned arima")+
  autolayer(sarima.1.forecast, PI=FALSE, series="sarima")+
  autolayer(reseasoned.arima.forecast, PI=FALSE, series = "re-seasoned arima")+
  autolayer(SES_data, PI=FALSE, series="exponential smoothing")+
  guides(colour=guide_legend(title="Forecast"))
```



Q9

From the plot in Q8 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.

Answer: The re-seasoned ARIMA and the SARIMA models clearly forecast better than the de-seasoned ARIMA or the exponential smoothing. This isn't particularly surprising, as the re-seasoned ARIMA and the SARIMA both have seasonal aspects to them (like the original data), while the ARIMA was explicitly de-seasoned and the exponential smoothing doesn't include seasonality in its forecasts - it is only based on the single previous observation. Between the two seasonal forecasts, the re-seasoned ARIMA appears to have larger variability - its low values better capture the general trend of low inflow months. In contrast the SARIMA forecast has less variability - its high values better seem to capture high inflow months than the re-seasoned ARIMA.

Q10

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 Q9?

Answer: It doesn't match at all! The de-seasoned ARIMA was the best bet according to MAPE…

```
#Exercise 4: Apply function accuracy() to the forecasts generated with each model you fit for the non-s

arima_scores <- accuracy(arima.forecast.1)
expon_scores<-accuracy(SES_data)
res_arima_scores<-accuracy(reseasoned.arima.forecast)
sarima_scores<-accuracy(sarima.1.forecast)
```

```
all_scores <- as.data.frame(rbind(arima_scores,expon_scores,res_arima_scores,sarima_scores))
row.names(all_scores) <- c("ARIMA", "ExponSmoothing","ReSeasonedARIMA", "SARIMA")

# Exercise 6: Decide which model is the best fit by comparing the RMSE metric, i.e, choose model with l
best_model_index <- which.min(all_scores[,"MAPE"])
cat("The best model by MAPE is:", row.names(all_scores[best_model_index,]))
```

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