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

Assignment 8 - Due date 03/27/23

Kelsey Husted

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
require(scales)
library(readxl)
library(openxlsx)
library(forecast)
library(tseries)
library(ggplot2)
library(Kendall)
library(lubridate)
library(tidyverse)
```

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.

```
#Import Data
Hydro_data <- read.table('./Data/inflowtimeseries.txt', sep='\t',stringsAsFactors = TRUE)

#Process Data
Hydro_data <- separate(Hydro_data, col="V1", int=c('Date', 'V'), sep='  ')
```

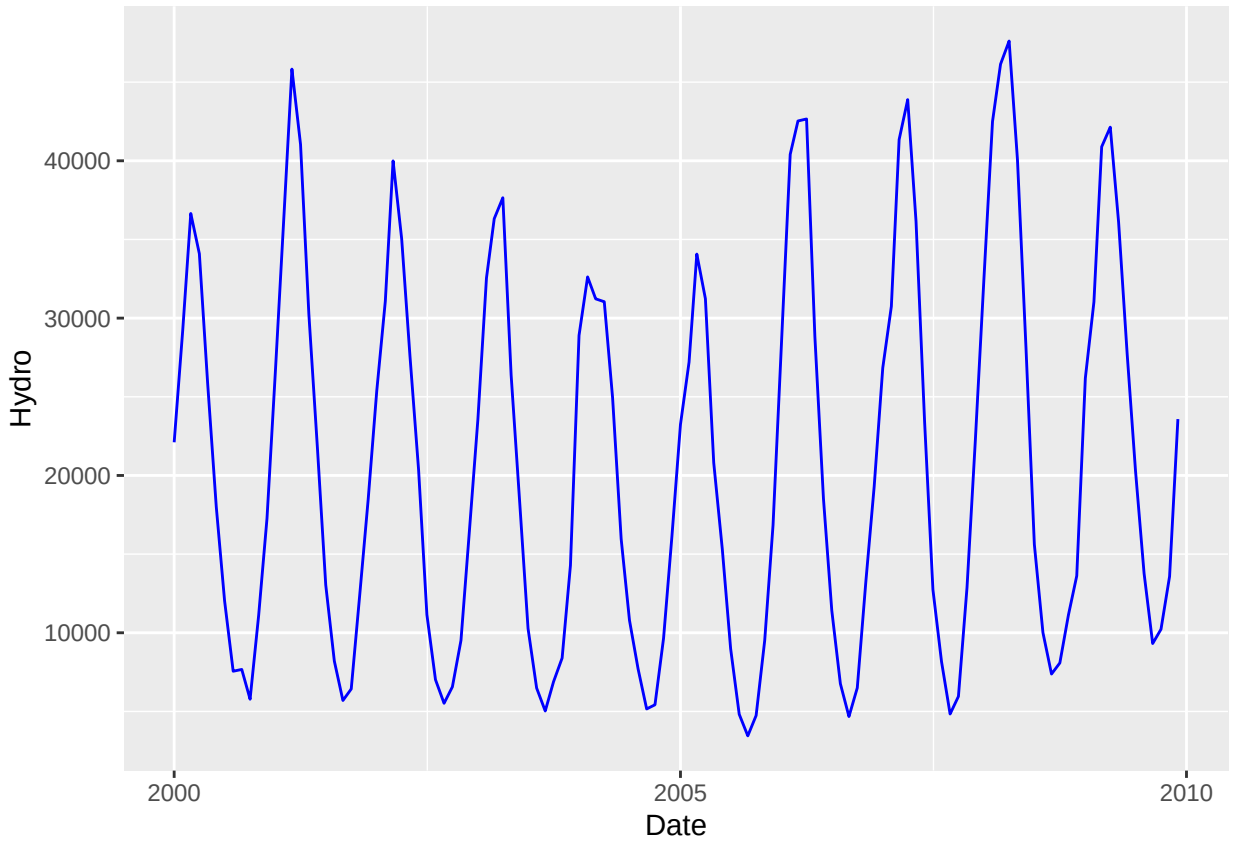
```
## Warning: Expected 2 pieces. Additional pieces discarded in 4 rows [969, 970,
## 971, 972].
```

```
## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 1 rows [626].
```

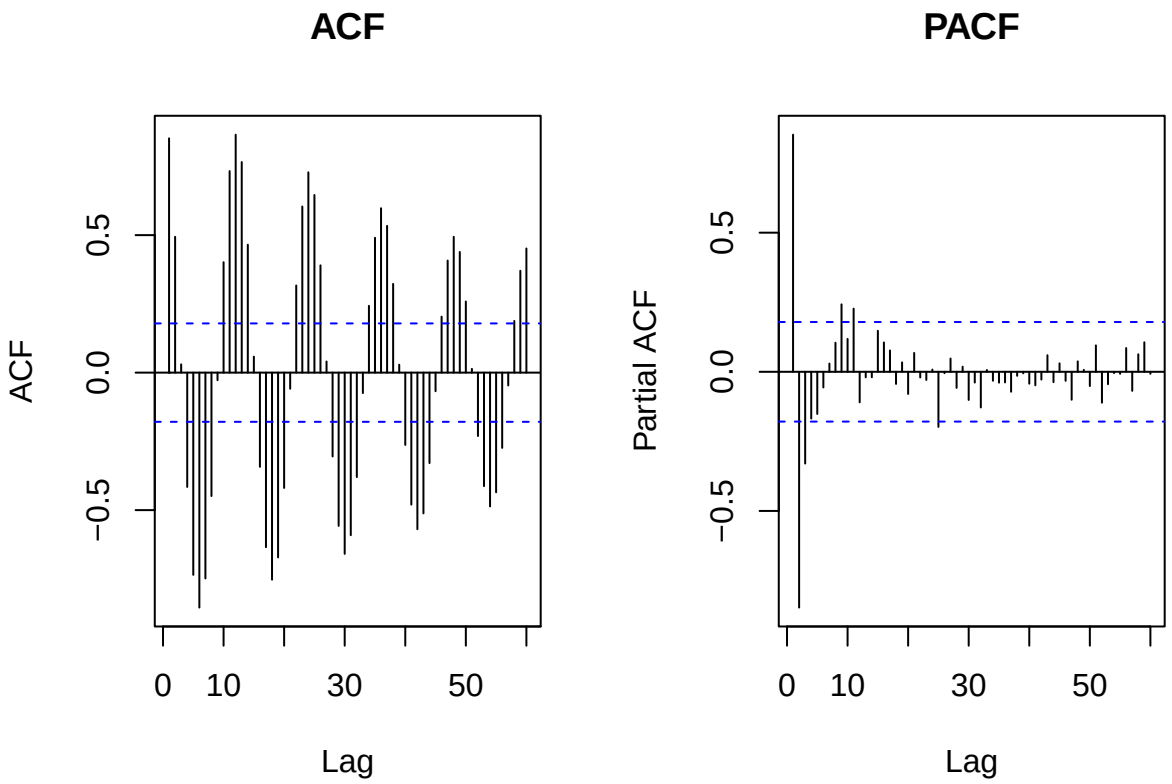
```
Hydro_clean <-
  Hydro_data %>%
  select('Date', 'V15') %>%
  rename(Hydro="V15") %>%
  mutate( Date = my(Date)) %>%
  filter(between(Date, as.Date('2000-01-01'), as.Date('2009-12-01')))
```

```
## Warning: 1 failed to parse.
```

```
#Create ts
ts_Hydro <- ts(Hydro_clean[,2],frequency = 12, start = c(2000,1))

#TS Plot
TS_Plot <-
  ggplot(Hydro_clean, aes(x=Date, y=Hydro)) +
      geom_line(color="blue")
plot(TS_Plot)
```




```
#ACF and PACF plots
par(mfrow=c(1,2))
ACF_Plot <- Acf(Hydro_clean$Hydro, lag = 60, plot = TRUE,main="ACF")
PACF_Plot <- Pacf(Hydro_clean$Hydro, lag = 60, plot = TRUE,main="PACF")
```

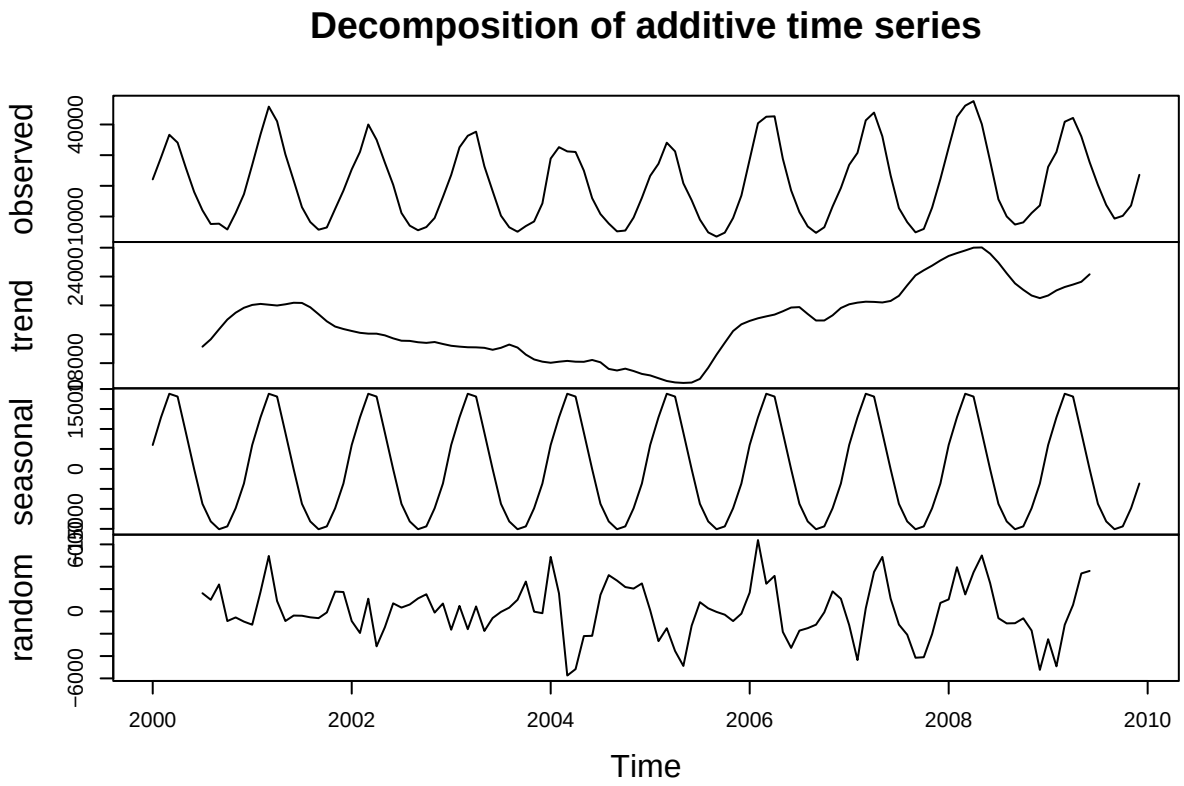


Q2

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.

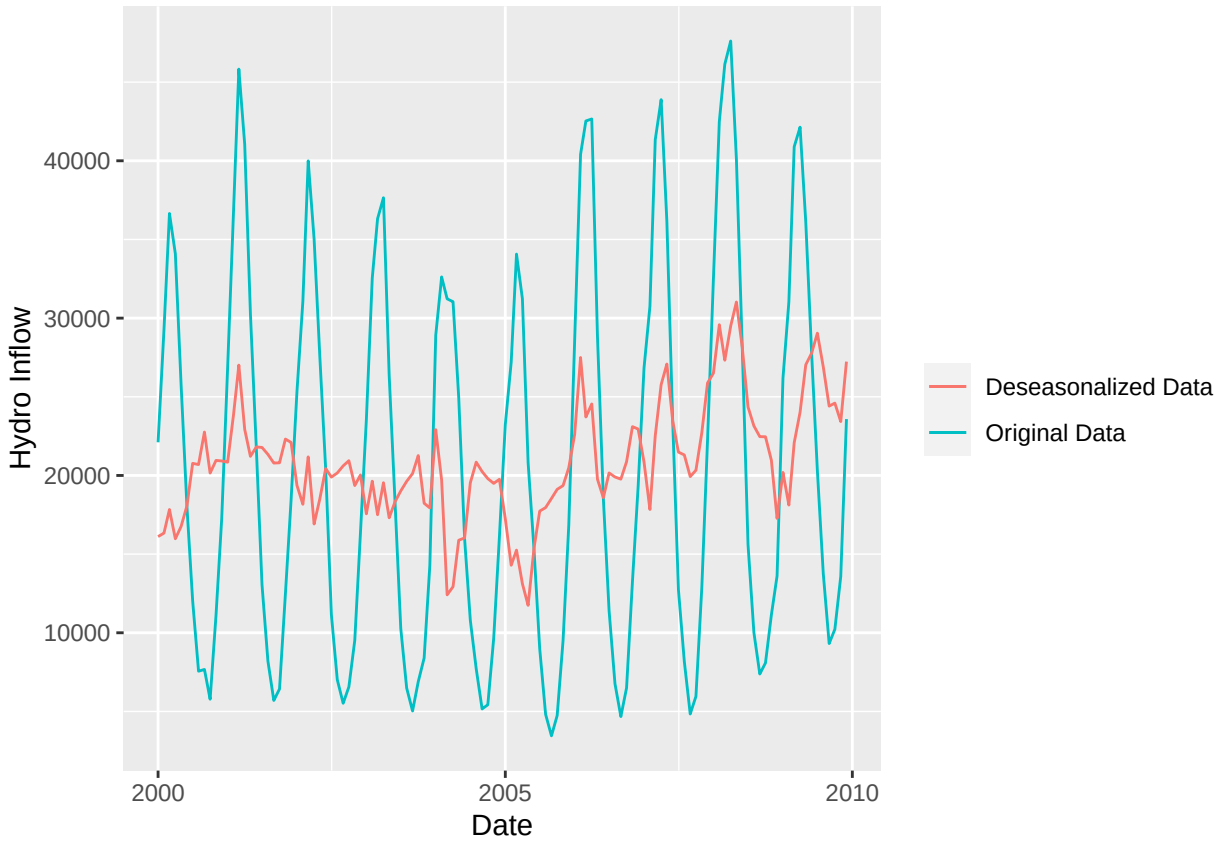
> Answer: In comparison to Q1, the ACF from Q2 looks less cyclic after seasonality was removed. The PACF in Q2 lost some initial significant lags in the beginning of the plot as well in comparison the the PACF in Q1.

```
#Decompose data
decomposed_Hydro <- decompose(ts_Hydro, type="additive")
plot(decomposed_Hydro)
```

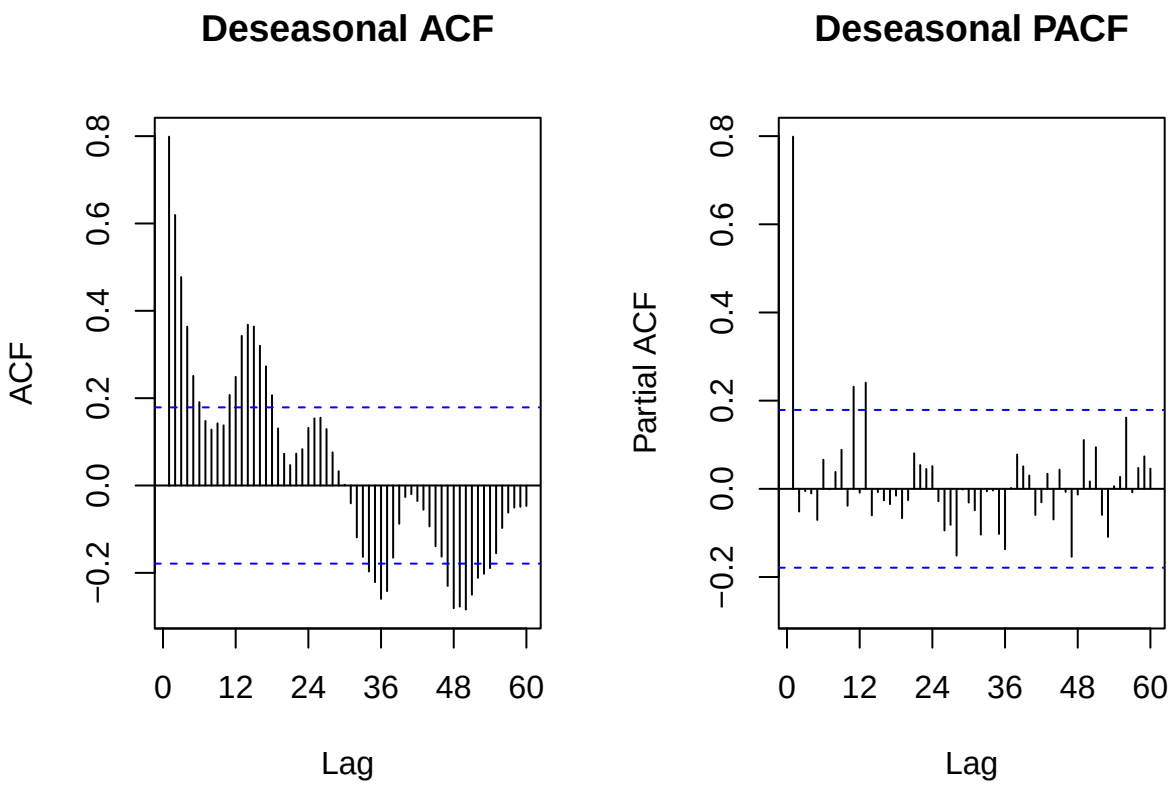



```
#Remove seasonality because some models cannot handle seasonality
deseasonal_ts <- seasadj(decomposed_Hydro)

#Create data frame for plots
deseasoned_Hydro <- as.data.frame(deseasonal_ts)
date <- Hydro_clean[,1]
Hydro <- Hydro_clean[,2]
Hydro_df <- data.frame(date , Hydro, deseasoned_Hydro)

#Deseasonalized and original data plots
ggplot(Hydro_df, aes(x=date, y=Hydro, colour = "Original Data", color="blue")) +
  geom_line() +
  geom_line(aes(y=x, colour = "Deseasonalized Data", col = "red")) +
  labs(y="Hydro Inflow", x = "Date", color = " ")
```

```
## Warning: Duplicated aesthetics after name standardisation: colour
## Duplicated aesthetics after name standardisation: colour
```




```
#ACF and PACF deseasonal plots
par(mfrow=c(1,2))
deseasonal_ACF_Plot <- Acf(deseasonal_ts, lag = 60, plot = TRUE,main="Deseasonal ACF")
deseasonal_PACF_Plot <- Pacf(deseasonal_ts, lag = 60, plot = TRUE,main="Deseasonal PACF")
```



Part II: Forecasting with ARIMA models and its variations

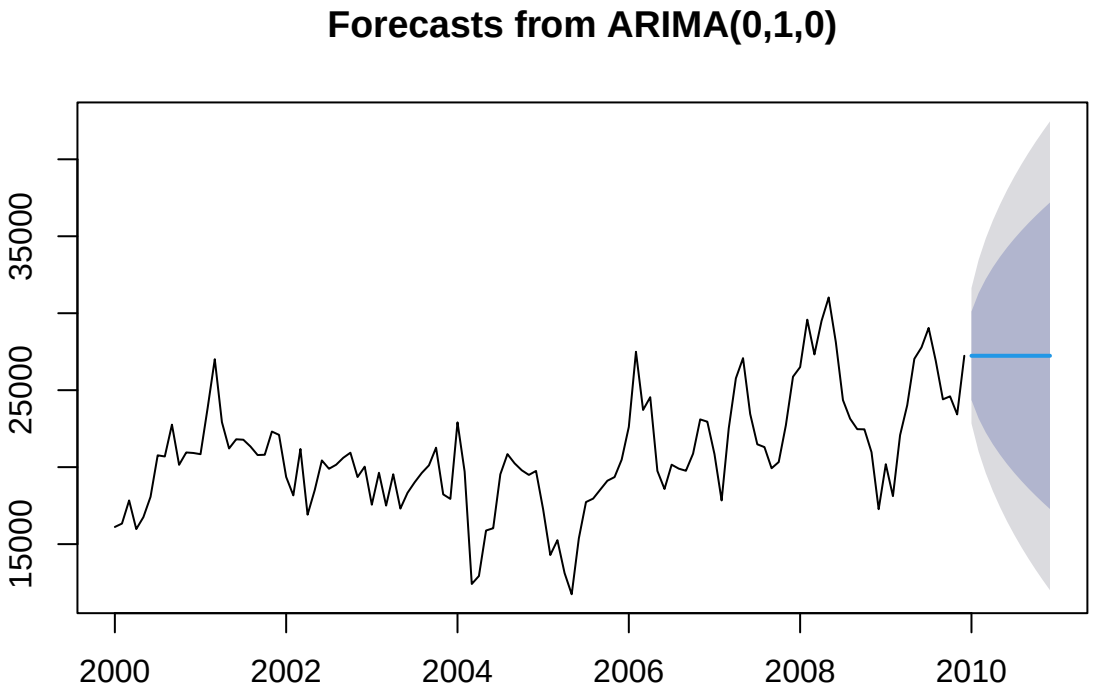
Q3

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).

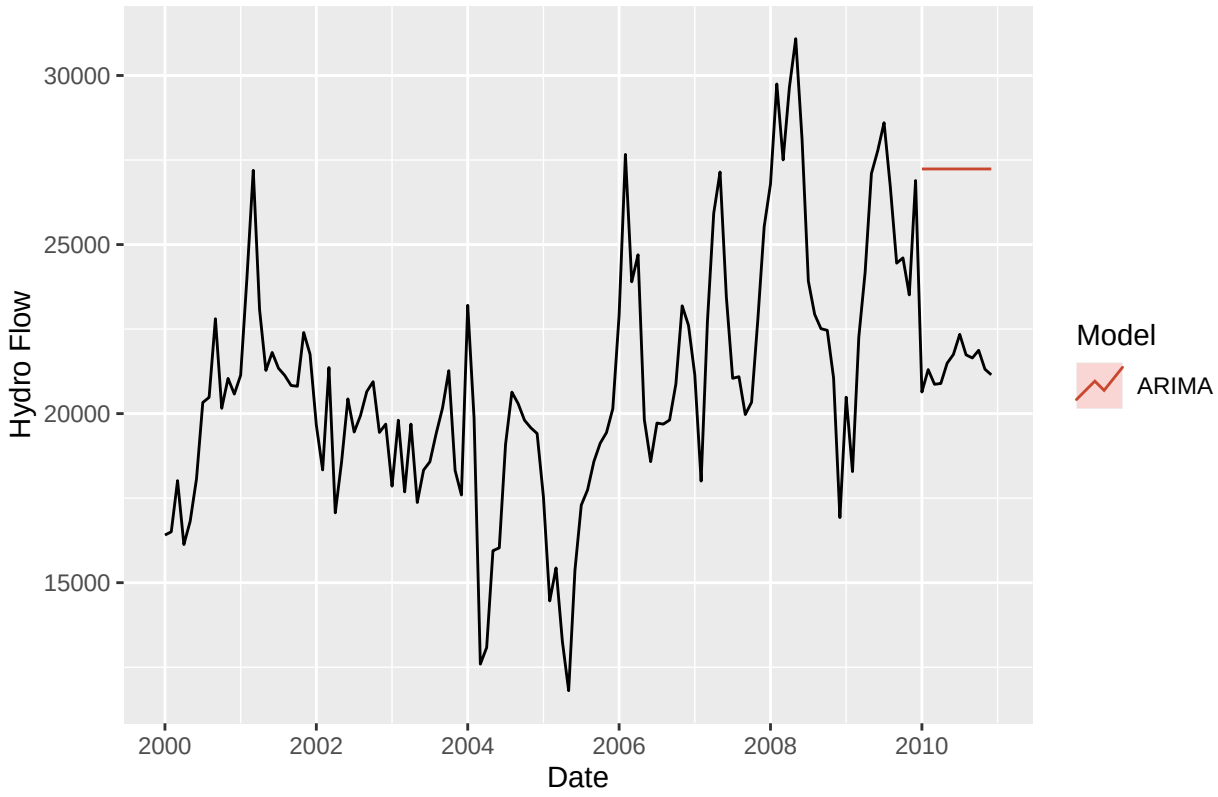
```
# Non-seasonal ARIMA
ARIMA_autofit <- auto.arima(deseasonal_ts)
print(ARIMA_autofit)
```

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

```
# Forecast
ARIMA_forecast <- forecast(object = ARIMA_autofit, h = 12)
plot(ARIMA_forecast)
```




```
# Plot
autoplot(deseasonal_tsV2) +
  autolayer(ARIMA_forecast, series="ARIMA", PI=FALSE) +
  ylab("Hydro Flow") +
  xlab("Date") +
  labs(col="Model")
```

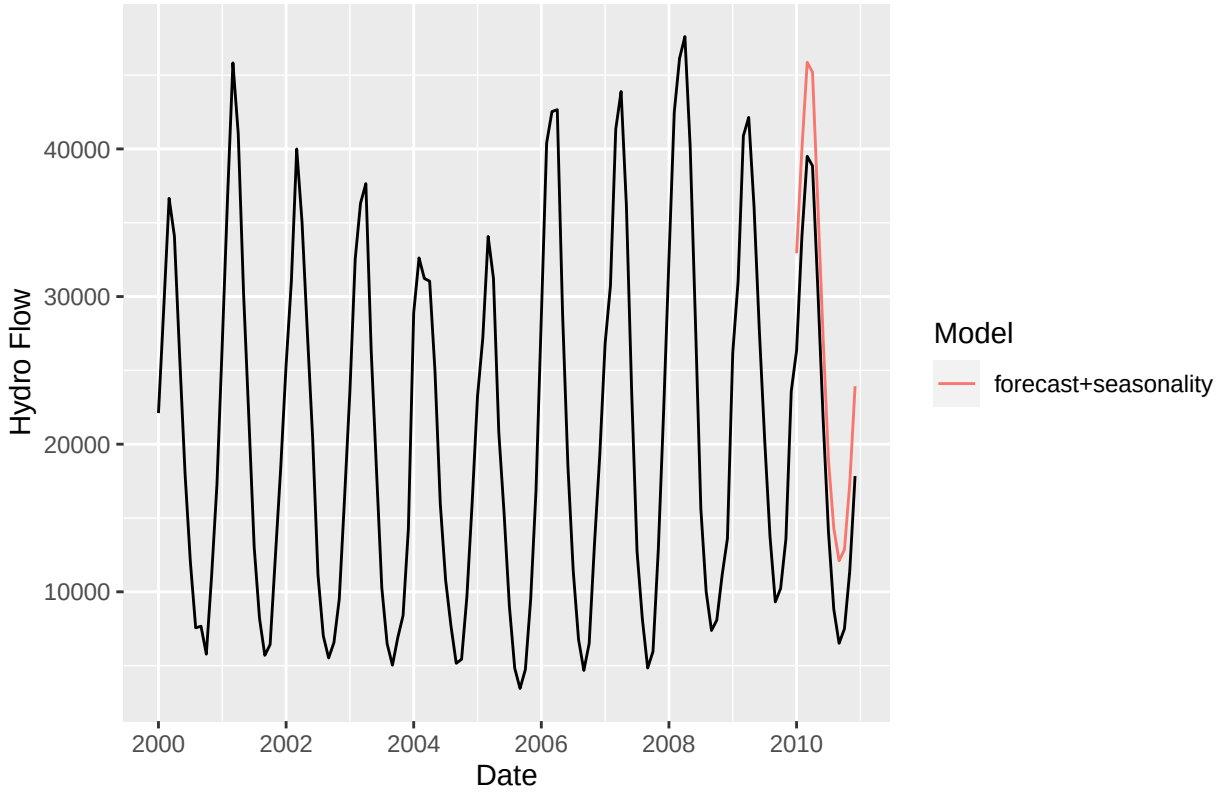


Q4

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 for 2010
season <- decomposed_HydroV2$seasonal
season_select <- season[121:132]

#Add seasonality back into forecasted ts
forecast <- as.data.frame(ARIMA_forecast)

#new data
new_data <- forecast$'Point Forecast' + season_select

ts_new_data <- ts(new_data, start = c(2010,1), frequency = 12)

# Plot to compare original data
autoplot(ts_HydroV2) +
  autolayer(ts_new_data, series = "forecast+seasonality", PI=FALSE) +
  ylab("Hydro Flow") +
  xlab("Date") +
  labs(col="Model")
```

```
## Warning in ggplot2::geom_line(ggplot2::aes_(x = ~timeVal, y = ~seriesVal, :
## Ignoring unknown parameters: `PI`
```



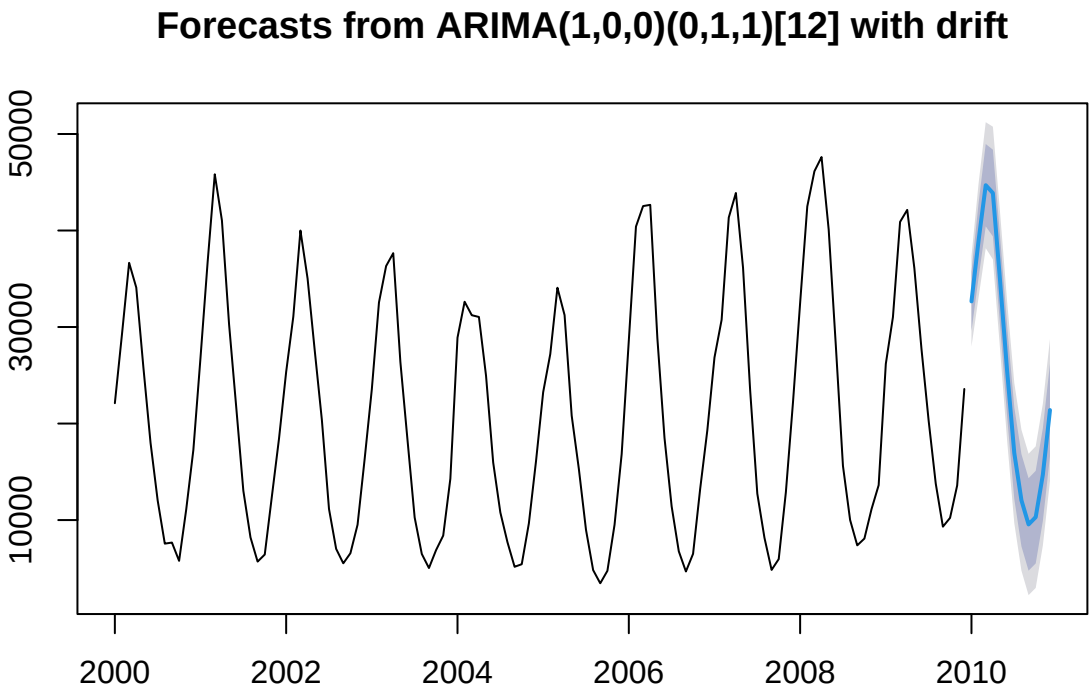
Q5

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().

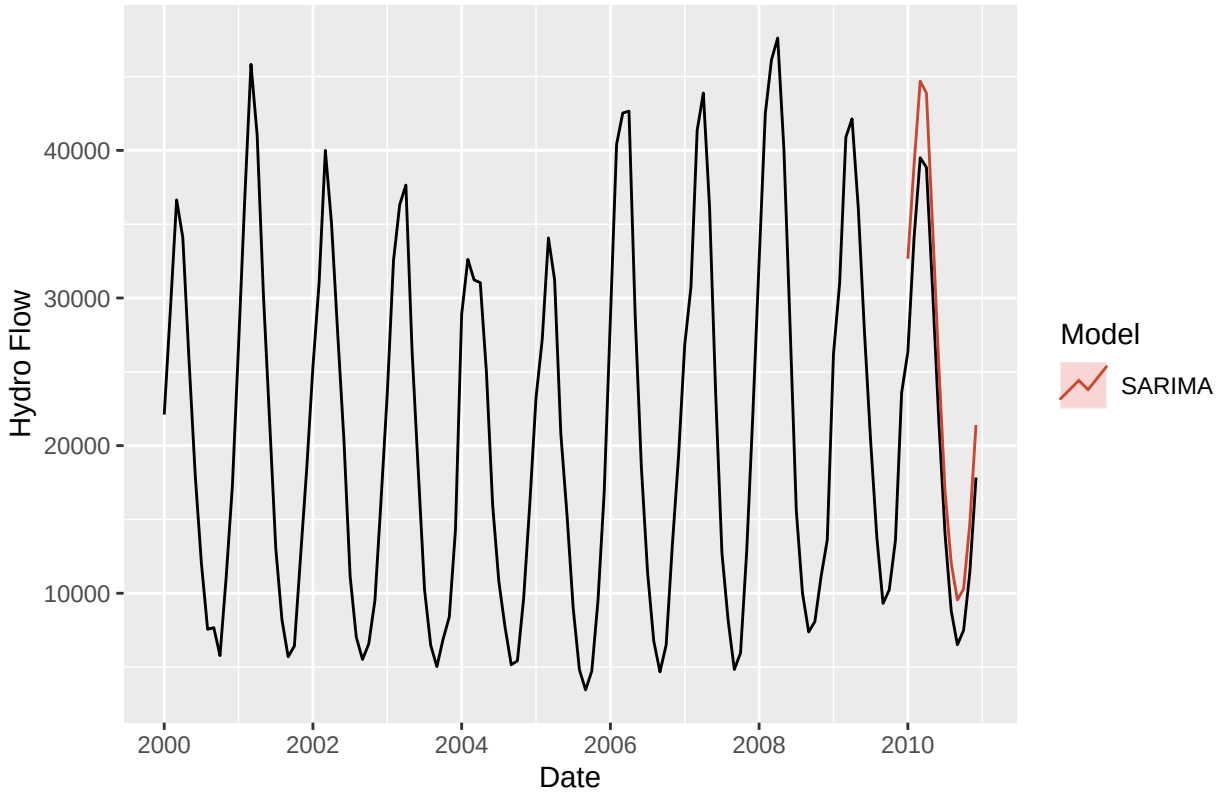
```
#Fit seasonal ARIMA model
SARIMA_autofit <- auto.arima(ts_Hydro)
print(SARIMA_autofit)
```

```
## Series: ts_Hydro
## 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 = 5566545:  log likelihood = -999.07
## AIC=2006.14   AICc=2006.52   BIC=2016.86
```

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




```
# Plot
autoplot(ts_HydroV2) +
  autolayer(SARIMA_forecast, series="SARIMA", PI=FALSE) +
  ylab("Hydro Flow") +
  xlab("Date") +
  labs(col="Model")
```

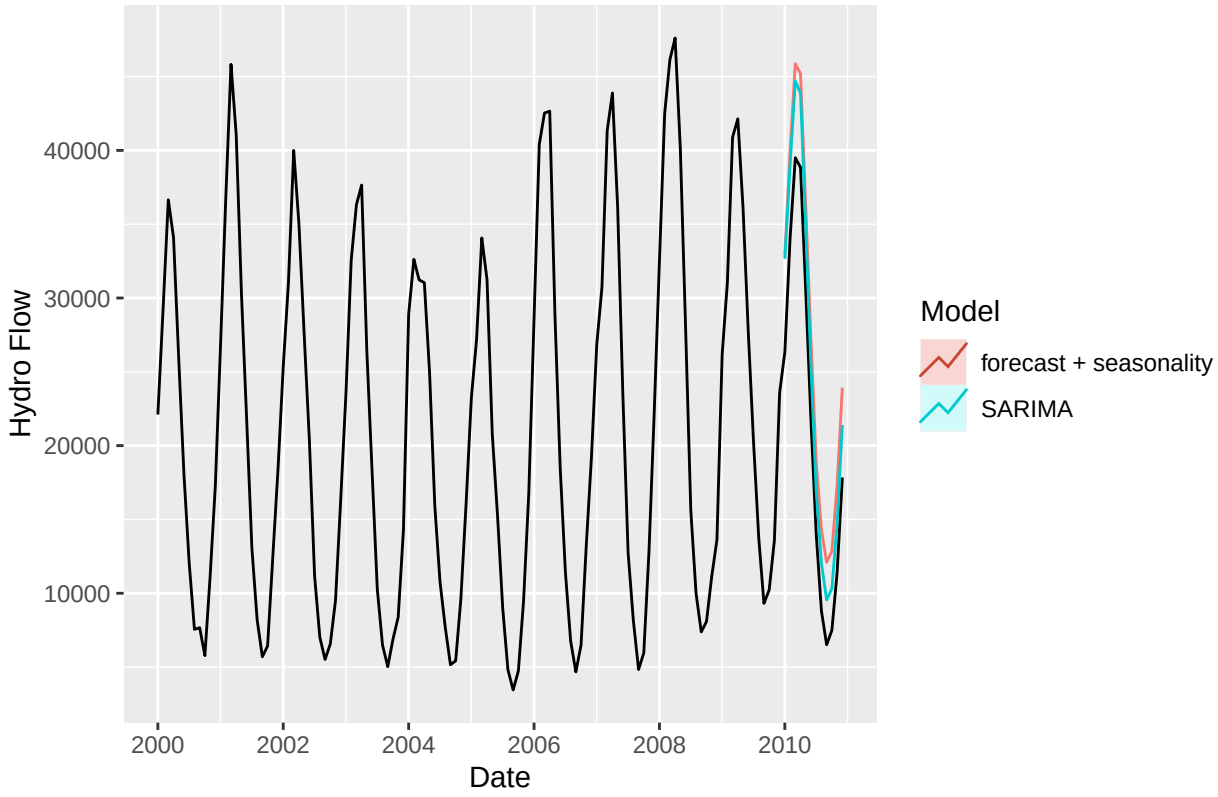


Q6

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

```
# Plot
autoplot(ts_HydroV2) +
  autolayer(ts_new_data, series="forecast + seasonality", PI=FALSE) +
  autolayer(SARIMA_forecast, series="SARIMA", PI=FALSE) +
  ylab("Hydro Flow") +
  xlab("Date") +
  labs(col="Model")
```

```
## Warning in ggplot2::geom_line(ggplot2::aes_(x = ~timeVal, y = ~seriesVal, :
## Ignoring unknown parameters: `PI`
```



Part III: Forecasting with Other Models

Q7

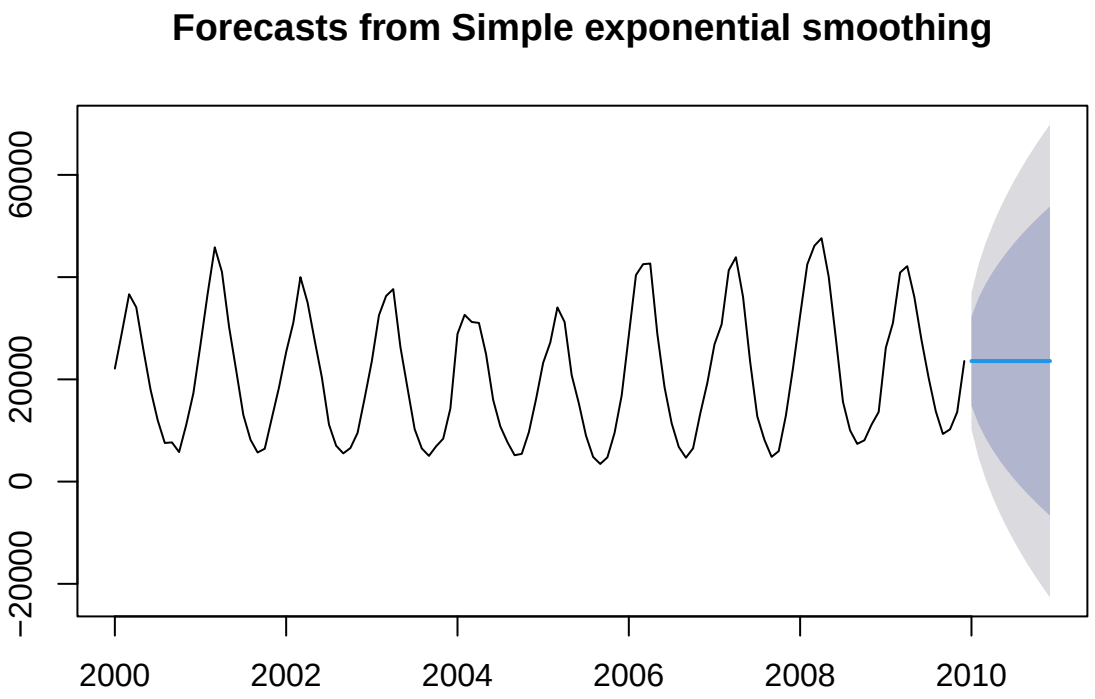
Fit an exponential smooth model to the original time series using the function *ses*() from package `forecast`. Note that this function automatically do the forecast. Do not forget to set the arguments: silent=FALSE and holdout=FALSE, so that the plot is produced and the forecast is for the year of 2010.

```
SES_seas_fit <- ses(y = ts_Hydro, h = 12, holdout = FALSE, silent = FALSE)
summary(SES_seas_fit)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = ts_Hydro, h = 12, holdout = FALSE, silent = FALSE)
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = 21885.2463
##
##   sigma:  6813.76
##
##      AIC     AICc      BIC
## 2696.890 2697.097 2705.252
```

```
##
## Error measures:
##                     ME    RMSE      MAE      MPE     MAPE     MASE      ACF1
## Training set 14.14103 6756.74 5787.733 -7.09394 34.05251 1.758853 0.7154525
##
## Forecasts:
##          Point Forecast      Lo 80    Hi 80      Lo 95    Hi 95
## Jan 2010          23582 14849.8153 32314.19  10227.276 36936.72
## Feb 2010          23582 11233.4433 35930.56   4696.512 42467.49
## Mar 2010          23582  8458.4205 38705.58    452.481 46711.52
## Apr 2010          23582  6118.9402 41045.06  -3125.445 50289.45
## May 2010          23582  4057.8032 43106.20  -6277.682 53441.68
## Jun 2010          23582  2194.3852 44969.62  -9127.534 56291.53
## Jul 2010          23582   480.7907 46683.21 -11748.251 58912.25
## Aug 2010          23582 -1114.1874 48278.19 -14187.559 61351.56
## Sep 2010          23582 -2612.2260 49776.23 -16478.612 63642.61
## Oct 2010          23582 -4029.1079 51193.11 -18645.546 65809.55
## Nov 2010          23582 -5376.7480 52540.75 -20706.583 67870.58
## Dec 2010          23582 -6664.4029 53828.40 -22675.882 69839.88
```

```
plot(SES_seas_fit)
```



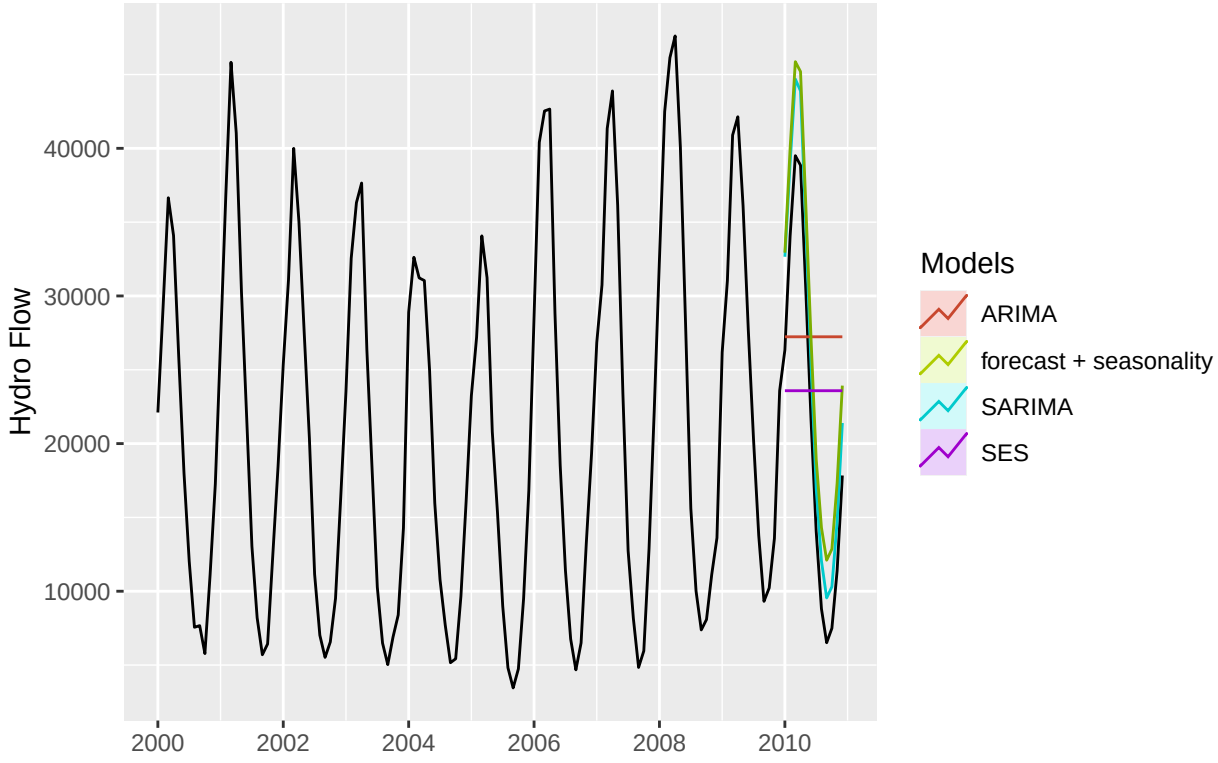
Part IV: Checking Forecast Accuracy

Q8

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.

```
#Plot ARIMA, SARIMA, Added Seasonality, and Exponential Smooth Models
autoplot(ts_HydroV2) +
    autolayer(SARIMA_forecast,series="SARIMA",PI=FALSE) +
    autolayer(ARIMA_forecast,series="ARIMA",PI=FALSE) +
    autolayer(ts_new_data, series="forecast + seasonality", PI=FALSE) +
    autolayer(SES_seas_fit,series="SES",PI=FALSE) +
    ylab("Hydro Flow") +
    xlab("") +
    labs(col="Models")
```

```
## Warning in ggplot2::geom_line(ggplot2::aes_(x = ~timeVal, y = ~seriesVal, :
## Ignoring unknown parameters: `PI`
```



Q9

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.

> Answer: According to the plot in Q8, the SARIMA model and forecast+seasonality models are leading to the better forecast since the pattern is matching the original data the best when compared to the other models as seen above.

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 Q10?

> Answer: Looking at the Table, the lowest MAPE does not correspond to my answer for the

Table 1: Forecast Accuracy for Hydro Inflow Data

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
SARIMA	-80.63461	2206.972	1611.593	-2.60900	8.99301	0.48975	0.07464
ARIMA	92.80632	2233.708	1750.409	-0.20111	8.64088	0.53194	-0.01906
SES	14.14103	6756.740	5787.733	-7.09394	34.05251	1.75885	0.71545

previous question.Even so, the RMSE matches my answer for the previous question.

```
library(kableExtra)
```

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

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

```
#Store performance metrics
ARIMA_scores <- accuracy(ARIMA_forecast)
SARIMA_scores <- accuracy(SARIMA_forecast)
SES_scores <- accuracy(SES_seas_fit)

#create data frame that combines metrics for all the models you created on Ex 3.
Scores <- as.data.frame(rbind(SARIMA_scores, ARIMA_scores, SES_scores))
row.names(Scores) <- c("SARIMA", "ARIMA","SES")

#Generate Table

kbl(Scores,
      caption = "Forecast Accuracy for Hydro Inflow Data",
      digits = array(5,ncol(Scores))) %>%
  kable_styling(full_width = FALSE, position = "center") %>%
  kable_styling(latex_options="striped", stripe_index = which.min(Scores[,"RMSE"])) %>%
  kable_styling(latex_options="striped", stripe_index = which.min(Scores[,"MAPE"]))
```