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

Assignment 8 - Due date 03/27/23

Tony Jiang

## 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(tseries)
library(tidyverse)
library(lubridate)
library(Kendall)
library(sarima)
library(forecast)
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 of 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.

```
inflow <- read.delim(file = "./Data/inflowtimeseries.txt",
                     header = FALSE) %>%
  separate(col = V1, into = c("Month", "Year", "v0"), sep = " ")

inino <- which(inflow$Month == "Jan" & inflow$Year == "2000")
colno <- ncol(inflow)
rowno <- nrow(inflow) - 24
rowno1 <- rowno + 12

# retrieve the last column to have a full dataset, including year 2010
```

```
# but excluding year 2011
ts_inflow_full <- inflow[c(inino:rowno1), colno-1] %>%
    ts(start = c(2000, 1), frequency = 12)

str(ts_inflow_full)
```

```
##  Time-Series [1:132] from 2000 to 2011: 22107 29247 36651 34089 25821 18007 12027 7557 7670 5777 ...
```

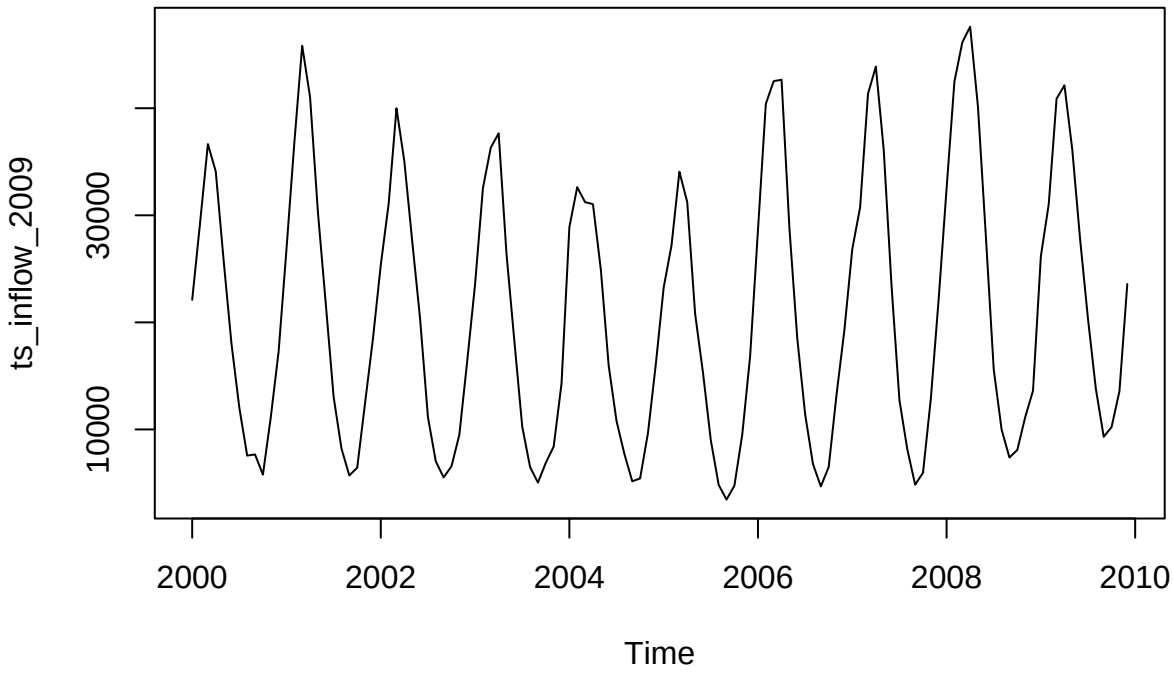
```
tail(ts_inflow_full)
```

```
##        Jul   Aug   Sep   Oct   Nov   Dec
## 2010 14043  8815  6512  7492 11387 17839
```

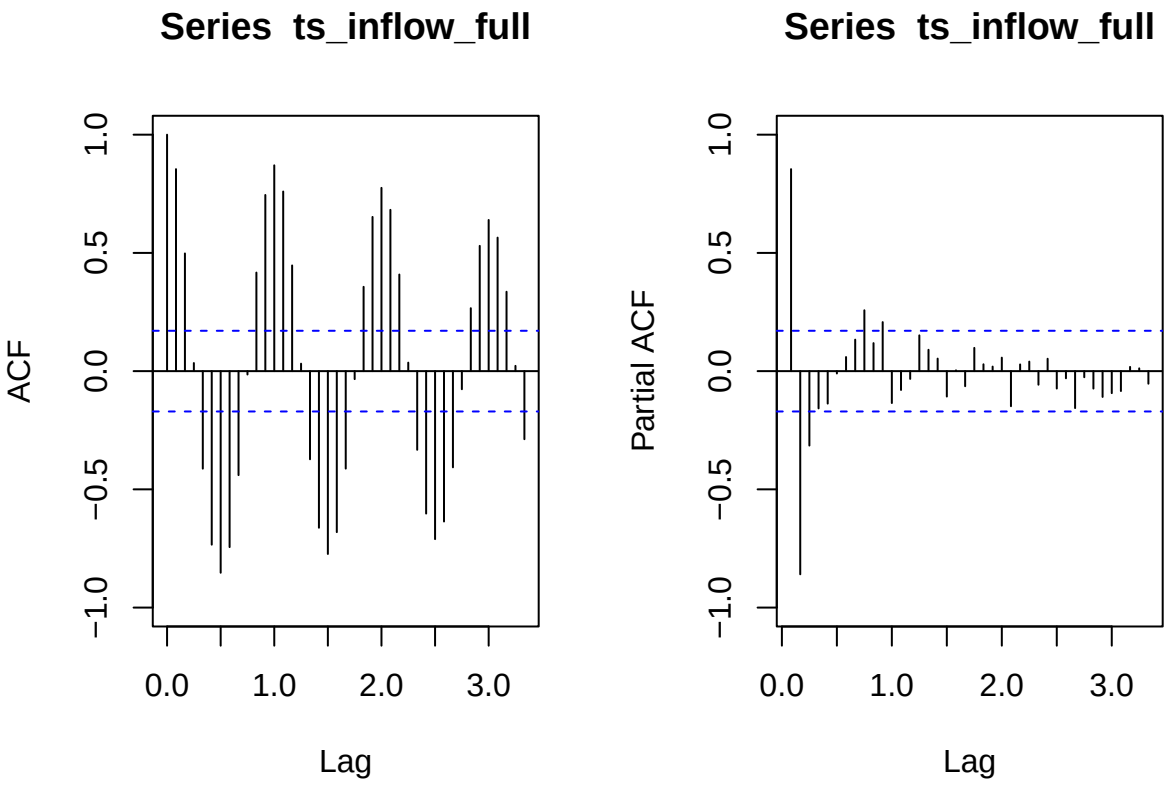
```
ts_inflow_2009 <- inflow[c(inino:rowno), colno-1] %>%
  ts(start = c(2000, 1), frequency = 12)

tail(ts_inflow_2009)
```

```
##        Jul   Aug   Sep   Oct   Nov   Dec
## 2009 20302 13772  9316 10225 13586 23583
```

```
# plot time series plot, ACF, and PACF
plot(ts_inflow_2009)
```




```
par(mfrow = c(1,2))
acf(ts_inflow_full, lag.max = 40, ylim = c(-1, 1))
pacf(ts_inflow_full, lag.max = 40, ylim = c(-1, 1))
```




**Q2**

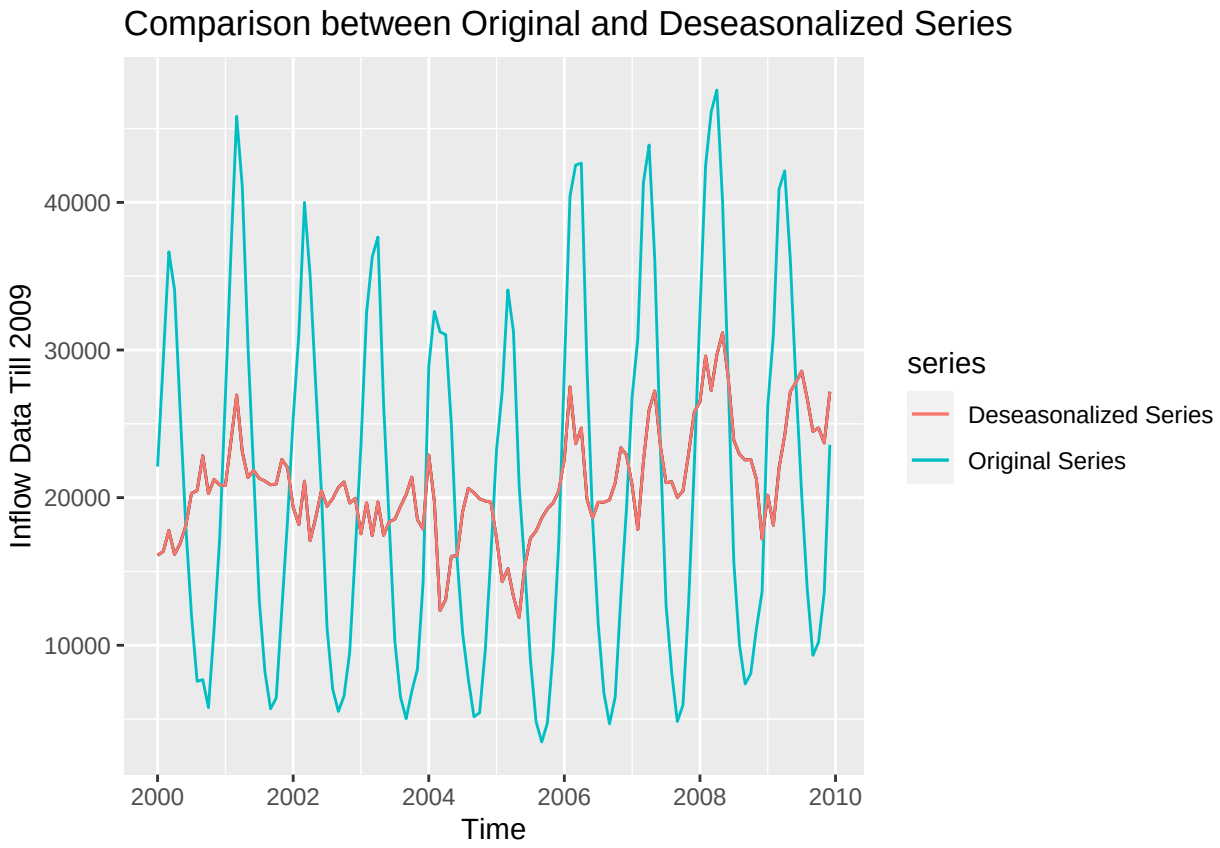
Using the *decompose*() or *stl*() and the *seasadj*() functions create a series without the seasonal component, i.e., a deseasonalized inflow series. Plot the deseasonalized series and original series together using ggplot, make sure your plot includes a legend. Plot ACF and PACF for the deaseasonalized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

```
# decompose and then retrieve the deseasonalized component
inflow_2009_deseas <- ts_inflow_2009 %>%
  stl(s.window = "periodic") %>%
  seasadj()

# examine the ts object to ensure that it is correct
str(inflow_2009_deseas)
```

```
##  Time-Series [1:120] from 2000 to 2010: 16097 16345 17773 16149 16895 ...
```

```
# use ggplot to plot the original and deseasonalized series
autoplot(inflow_2009_deseas) +
  autolayer(ts_inflow_2009, series = "Original Series", PI = FALSE) +
  autolayer(inflow_2009_deseas, series = "Deseasonalized Series", PI = FALSE) +
  ylab("Inflow Data Till 2009") +
  ggtitle("Comparison between Original and Deseasonalized Series")
```

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




## Part II: Forecasting with ARIMA models and its variations

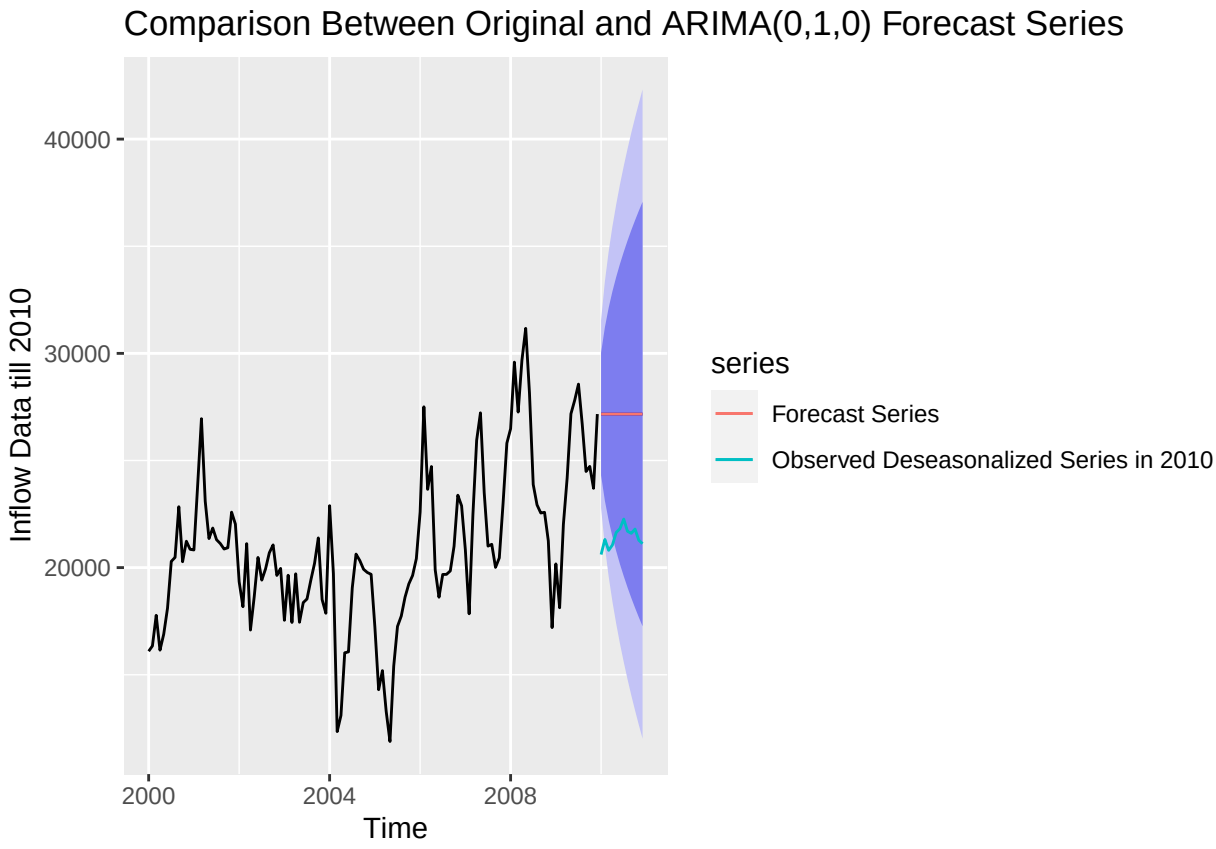
### Q3

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).

```
# model and forecast the inflow data for year 2010
inflow_2009_deseas <- ts_inflow_2009 %>%
  stl(s.window = "periodic") %>%
  seasadj()

inflow_for <- inflow_2009_deseas %>%
  auto.arima() %>%
  forecast(h = 12)

# deseasonalize the original series
inflow_2010_deseas <- ts_inflow_full %>%
  stl(s.window = "periodic") %>%
  seasadj()

# plot the forecast and original series together
inflow_for %>%
  autoplot() +
  autolayer(inflow_for$mean, series = "Forecast Series", PI = FALSE) +
  autolayer(window(inflow_2010_deseas,
```

```
                    start = c(2010, 1),
                    end = c(2010, 12)
                    ),
             series = "Observed Deseasonalized Series in 2010") +
  ylab("Inflow Data till 2010") +
  ggtitle("Comparison Between Original and ARIMA(0,1,0) Forecast Series")
```

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



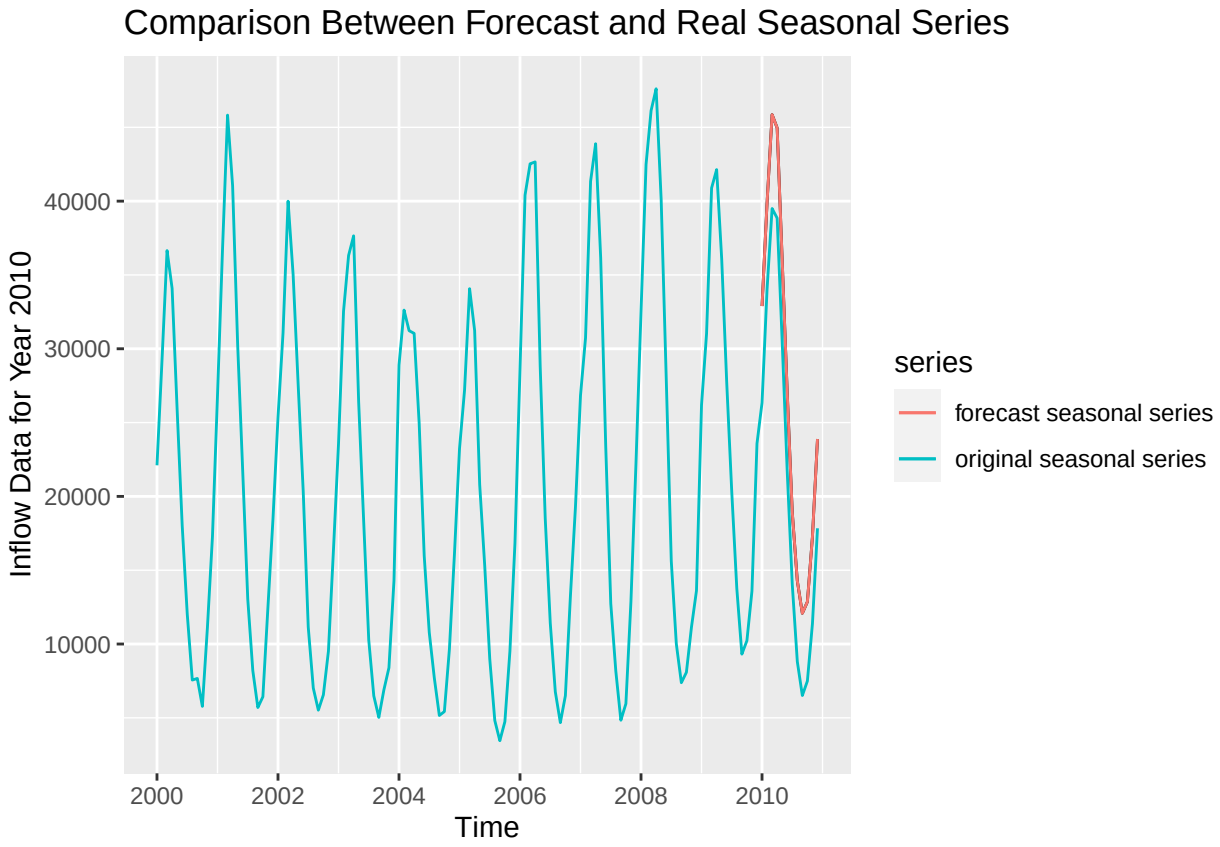

**Q4**

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.

```
# retrieve the seasonal component of the observed series
seasonalpart_2010 <- ts_inflow_full %>%
  stl(s.window = "periodic") %>%
  seasonal()

# add the seasonal component of the last year to
# the forecast deseasonalized values
for_full_2010 <- seasonalpart_2010[c(121:132)] + inflow_for$mean

# plot the forecast series and observed series
for_full_2010 %>%
  autoplot() +
```

```
  autolayer(for_full_2010, series = "forecast seasonal series", PI = FALSE) +
  autolayer(ts_inflow_full, series = "original seasonal series") +
  ylab("Inflow Data for Year 2010") +
  ggtitle("Comparison Between Forecast and Real Seasonal Series")
```

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

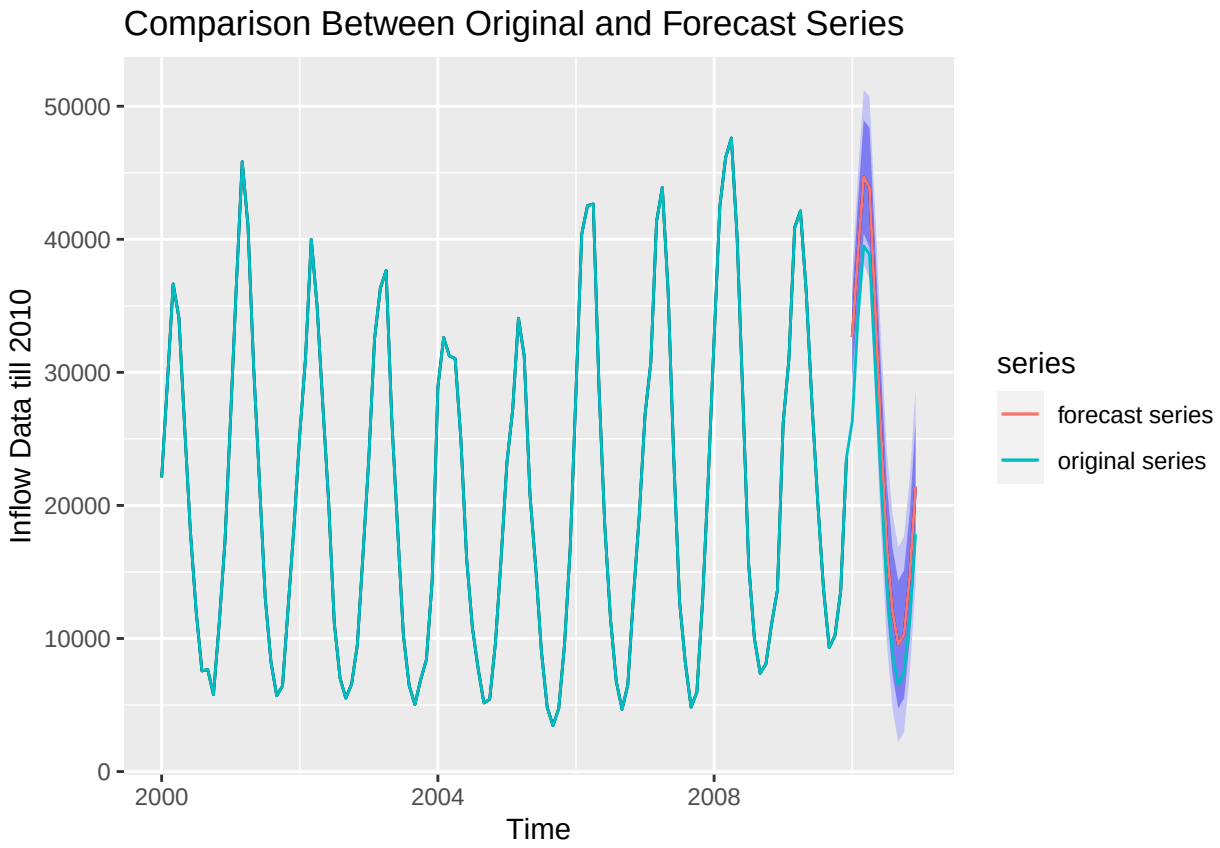



**Q5**

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().

```
# forecast the data till 2009 without deseasonalizing the series
inflow_for_original <- ts_inflow_2009 %>%
  auto.arima() %>%
  forecast(h = 12)

# plot the forecast series and the observed series together
inflow_for_original %>%
  autoplot() +
  autolayer(inflow_for_original$mean, series = "forecast series", PI = FALSE) +
  autolayer(ts_inflow_full, series = "original series") +
  ylab("Inflow Data till 2010") +
  ggtitle("Comparison Between Original and Forecast Series")
```

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

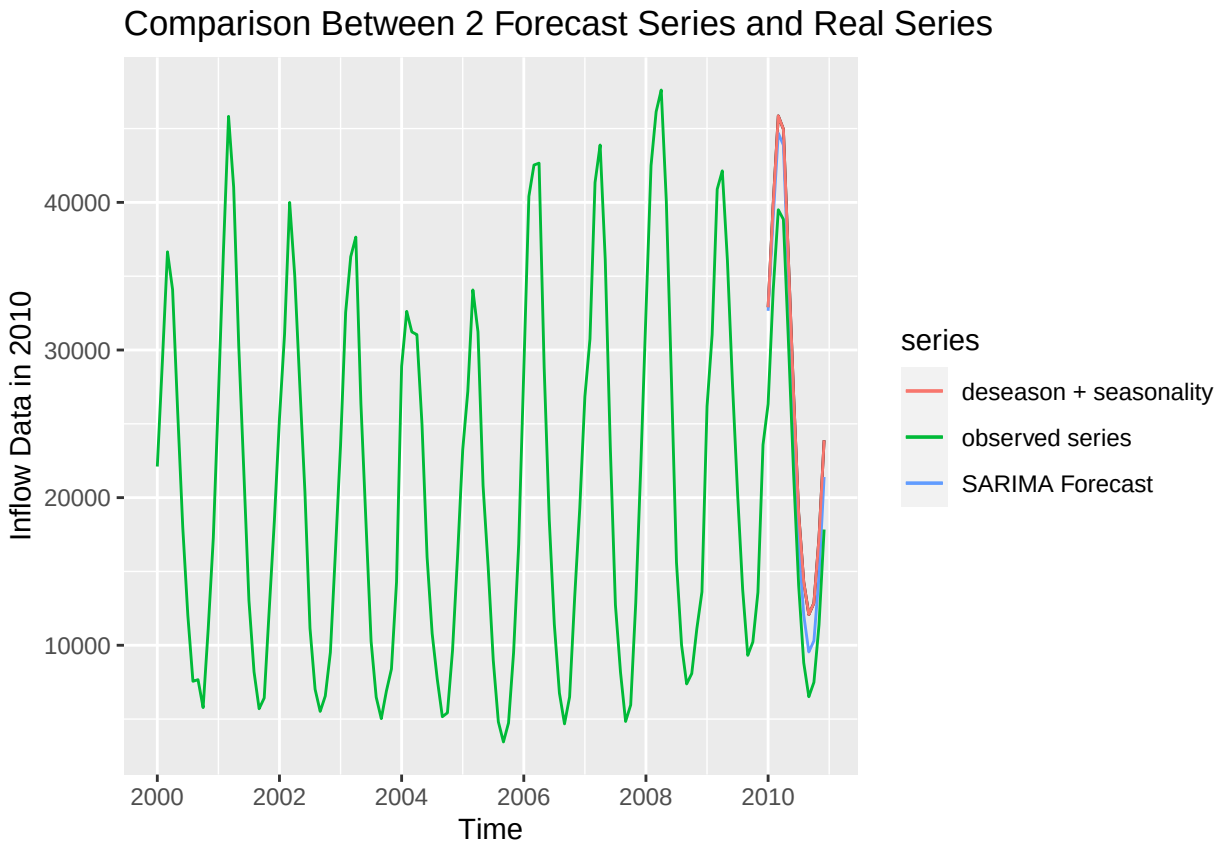



**Q6**

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

```
# plot two forecast series and the observed series in the same plot to compare
autoplot(for_full_2010) +
  autolayer(inflow_for_original$mean, series = "SARIMA Forecast", CI = FALSE) +
  autolayer(for_full_2010, series = "deseason + seasonality", CI = FALSE) +
  autolayer(ts_inflow_full, series = "observed series") +
  ylab("Inflow Data in 2010") +
  ggtitle("Comparison Between 2 Forecast Series and Real Series")
```

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

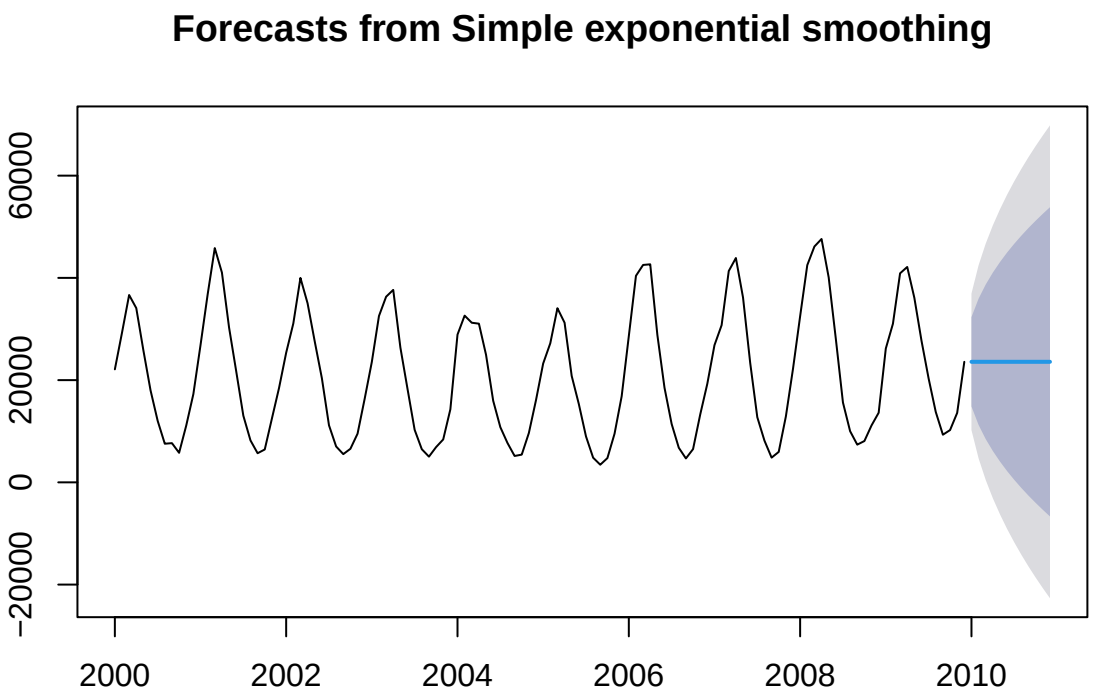



## Part III: Forecasting with Other Models

### 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.

```
# forecast the series using exponential smooth model
expo_smooth <- ts_inflow_2009 %>%
  ses(h = 12, silent = FALSE, holdout = FALSE)

# plot the forecast series
plot(expo_smooth)
```



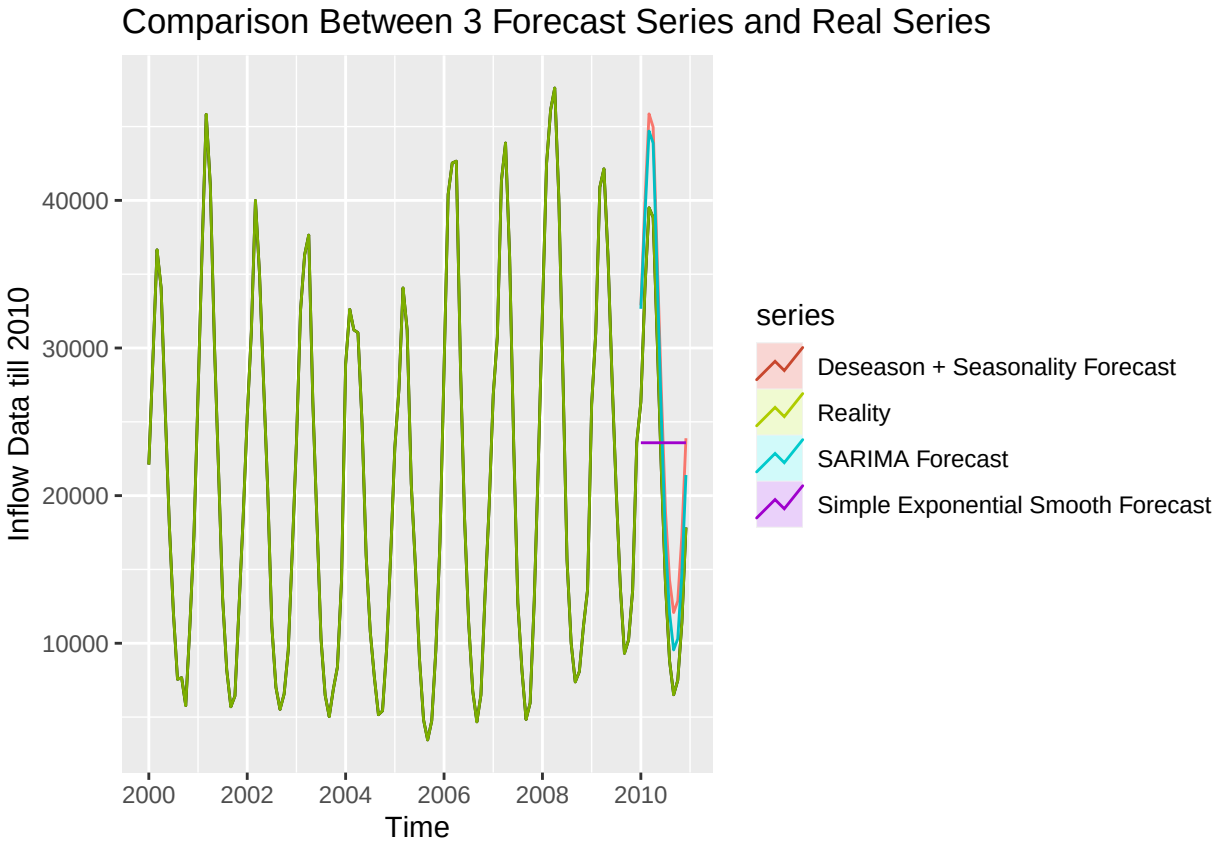

## Part IV: Checking Forecast Accuracy

### 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 all three forecast series and the observed series
# to compare their performances
autoplot(ts_inflow_full) +
  autolayer(ts_inflow_full, series = "Reality") +
  autolayer(for_full_2010, series = "Deseason + Seasonality Forecast", PI = FALSE) +
  autolayer(inflow_for_original$mean, series = "SARIMA Forecast", PI = FALSE) +
  autolayer(expo_smooth, series = "Simple Exponential Smooth Forecast", PI = FALSE) +
  ylab("Inflow Data till 2010") +
  ggtitle("Comparison Between 3 Forecast Series and Real Series")
```

```
## Warning in ggplot2::geom_line(ggplot2::aes_(x = ~timeVal, y = ~seriesVal, : Ignoring unknown paramet
## 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: Forecast in Q5, using seasonal Arima to forecast, has the best forecast. In the graph in Q8, we can see that forecast in Q5 is the closest to the peak and trough of the reality series, which means it should have the smallest standard error (the difference between the fitted value and the real values).

**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: Yes, in the table, SARIMA has the lowest RMSE and, therefore, best performance, which aligns with my judgment in Q9.

```
# get the metrics of the three forecast series
original_inflow_2010 <- ts_inflow_full %>%
  window(start = c(2010, 1),
         end = c(2010, 12))

deandse_score <- accuracy(inflow_for$mean, original_inflow_2010)

onestep_score <- accuracy(inflow_for_original$mean, original_inflow_2010)
```

Table 1: Forecast Accuracy for Seasonal Inflow Data

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| ARIMA | -5750.384 | 13027.497 | 11454.128 | -84.02912 | 99.43078 | 0.87107 | 4.31878 |
| SARIMA | -4031.818 | 4171.259 | 4031.818 | -23.87315 | 23.87315 | 0.65236 | 0.80654 |
| SES | -2165.000 | 11888.481 | 10718.167 | -59.74209 | 83.65282 | 0.87107 | 3.51040 |

```
ses_score <- accuracy(expo_smooth$mean, original_inflow_2010)

# formulate a table with the metrics of the three forecast series
compare_scores <- as.data.frame(rbind(deandse_score, onestep_score, ses_score))
row.names(compare_scores) <- c("ARIMA", "SARIMA","SES")

# output a table highlighting the forecast series with the lowest RMSE
kbl(compare_scores,
      caption = "Forecast Accuracy for Seasonal Inflow Data",
      digits = array(5,ncol(compare_scores)
                     )
    ) %>%
  kable_styling(full_width = FALSE, position = "center") %>%
  #highlight model with lowest RMSE
  kable_styling(latex_options="striped",
                stripe_color = "red",
                stripe_index = which.min(compare_scores[,"RMSE"])
                )
```