

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

Assignment 8 - Due date 03/27/23

John Rooney

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(forecast)
```

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

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

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

```
## Package "greybox", v1.0.7 loaded.
```

```
## This is package "smooth", v3.2.0
```

```
library(here)
```

```
## here() starts at /Users/jrooney/Library/Mobile Documents/com~apple~CloudDocs/TSA_Sp23_new
```

```
library(lubridate)
```

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

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

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

```
library(tidyverse)
```

```
## -- Attaching packages ----- tidyverse 1.3.2 --

## v ggplot2 3.4.0      v purrr  1.0.1
## v tibble  3.1.8      v dplyr  1.1.0
## v tidyr   1.3.0      v stringr 1.5.0
## v readr   2.1.3      v forcats 1.0.0
## -- Conflicts ----- tidyverse_conflicts() --
## x lubridate::as.difftime() masks base::as.difftime()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks stats::filter()
## x lubridate::hm()          masks greybox::hm()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x lubridate::setdiff()     masks base::setdiff()
## x tidyr::spread()          masks greybox::spread()
## x lubridate::union()       masks base::union()
```

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 from 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 file into data frame
data = "Data/"
raw_inflow_data <- read.table(
  here(data, "inflowtimeseries.txt"), header=F, sep = "")

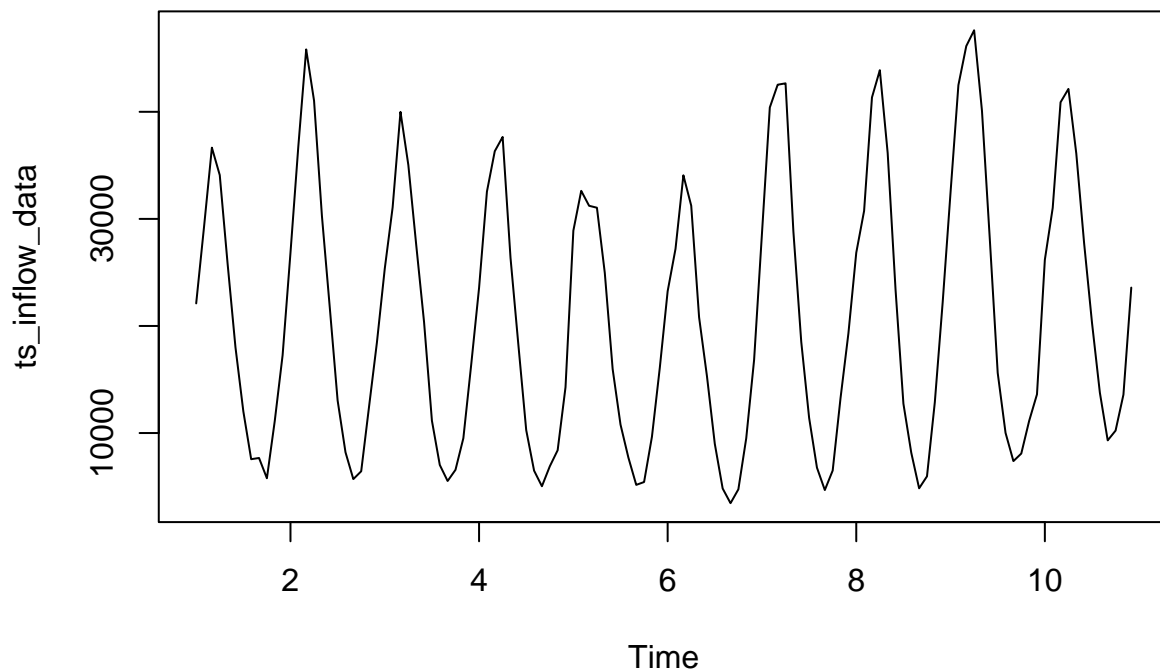
#add column names
colnames(raw_inflow_data)=c("Month", "Year", "HP1", "HP2", "HP3", "HP4", "HP5",
  "HP6", "HP7", "HP8", "HP9", "HP10", "HP11", "HP12",
  "HP13", "HP14", "HP15")

#wrap data set
inflow_data<-
  raw_inflow_data %>%
  dplyr::filter(Year >= 2000 & Year <= 2009) %>%
  dplyr::select(Month, Year, HP15)

#make a time series for HP15 from January 2000 to December 2009
ts_inflow_data <- ts(inflow_data[,3], start=1, frequency=12)

#plot the time series
plot(ts_inflow_data)

```

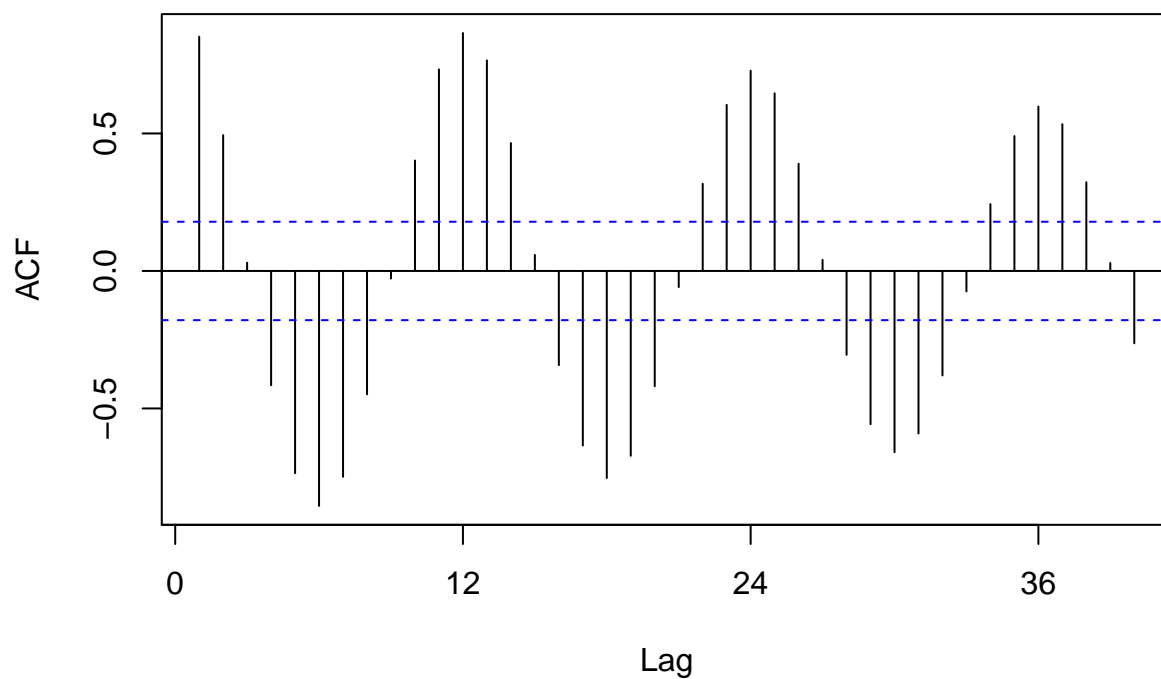


```

#Acf and Pacf for HP15
Acf(ts_inflow_data, lag.max=40, main=paste("Inflows HP15", sep=""))

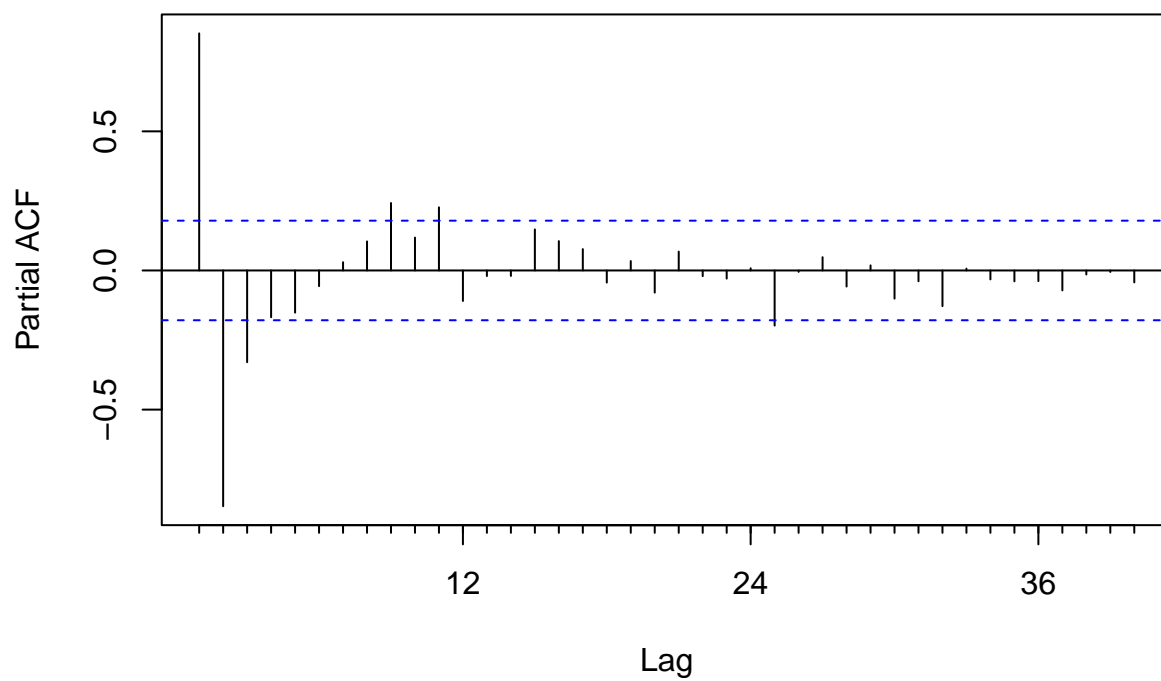
```

Inflows HP15



```
Pacf(ts_inflow_data,lag.max=40,main=paste("Inflows HP15",sep=""))
```

Inflows HP15



Q2

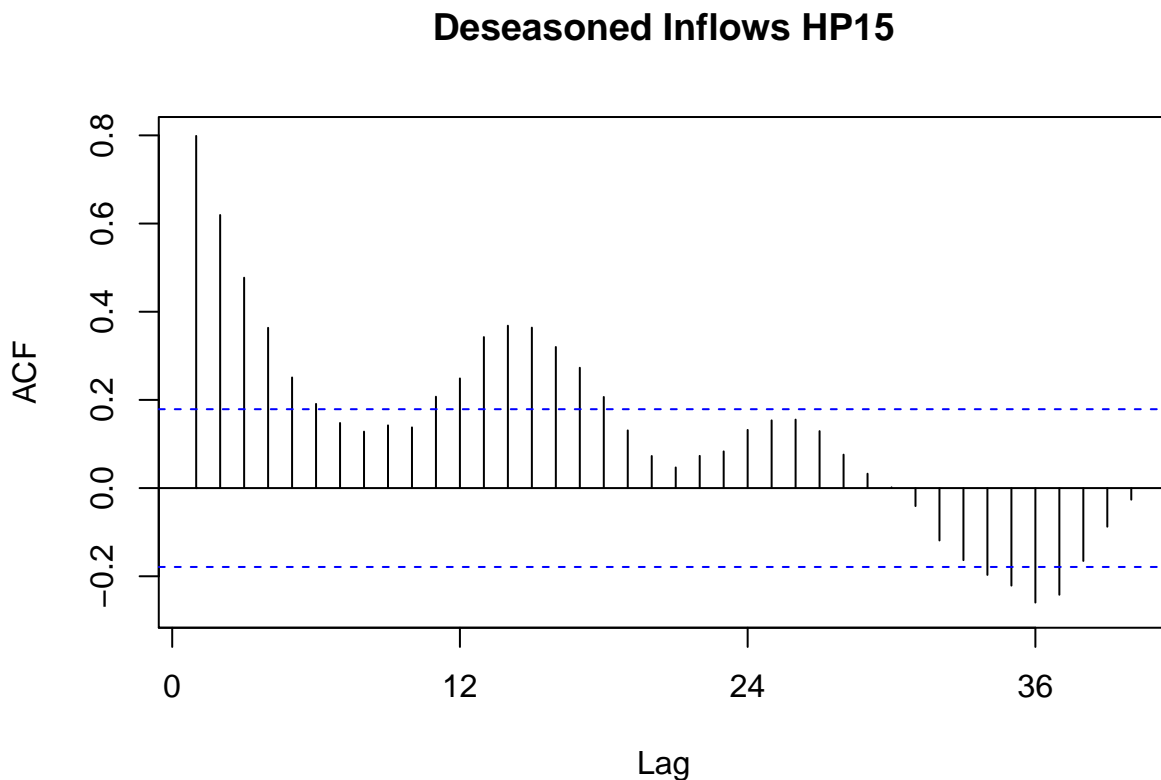
Using the `decompose()` or `stl()` and the `seasadj()` functions create a series without the seasonal component, i.e., a deseasonalized inflow series. Plot the deseasonalized series and original series together using `ggplot`, make sure your plot includes a legend. Plot ACF and PACF for the deseasonalized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

```
#decompose the time series
decomp_inflow_data <- decompose(ts_inflow_data, "additive")

deseason_inflow_data <- seasadj(decomp_inflow_data)

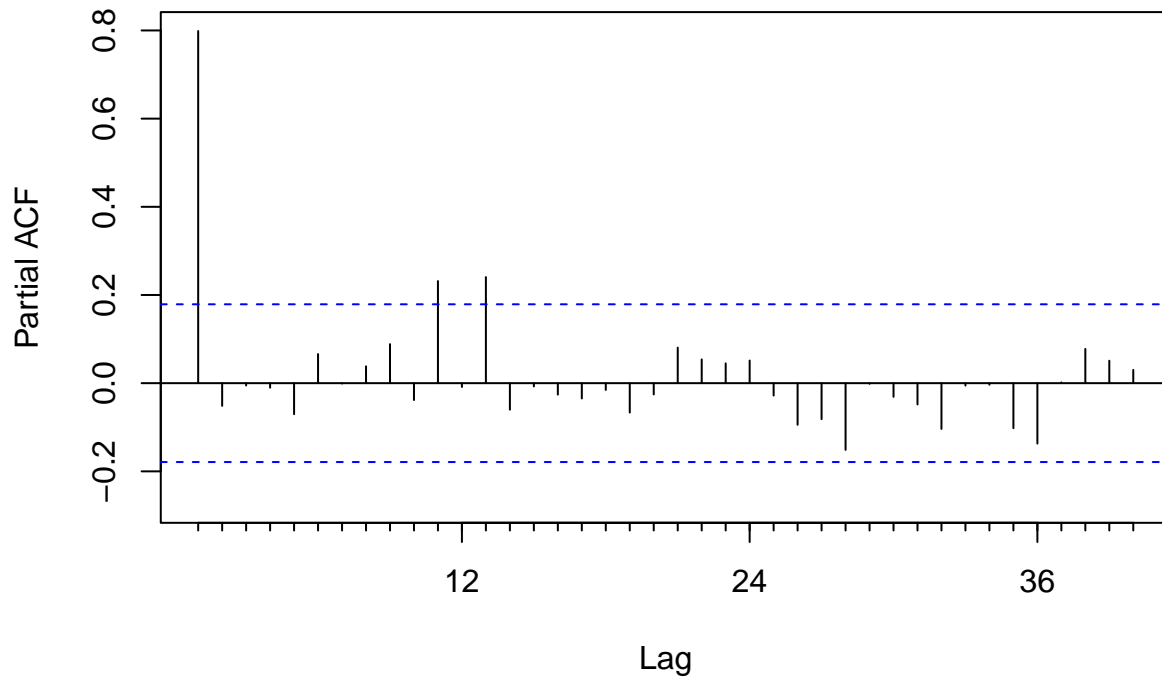
#plot deseasonalized and original series
#deseasonal_and_original_plot

#ACF and PACF
Acf(deseason_inflow_data, lag.max=40, main=paste("Deseasoned Inflows HP15", sep=""))
```



```
Pacf(deseason_inflow_data, lag.max=40, main=paste("Inflows HP15", sep=""))
```

Inflows HP15



Part II: Forecasting with ARIMA models and its variations

Q3

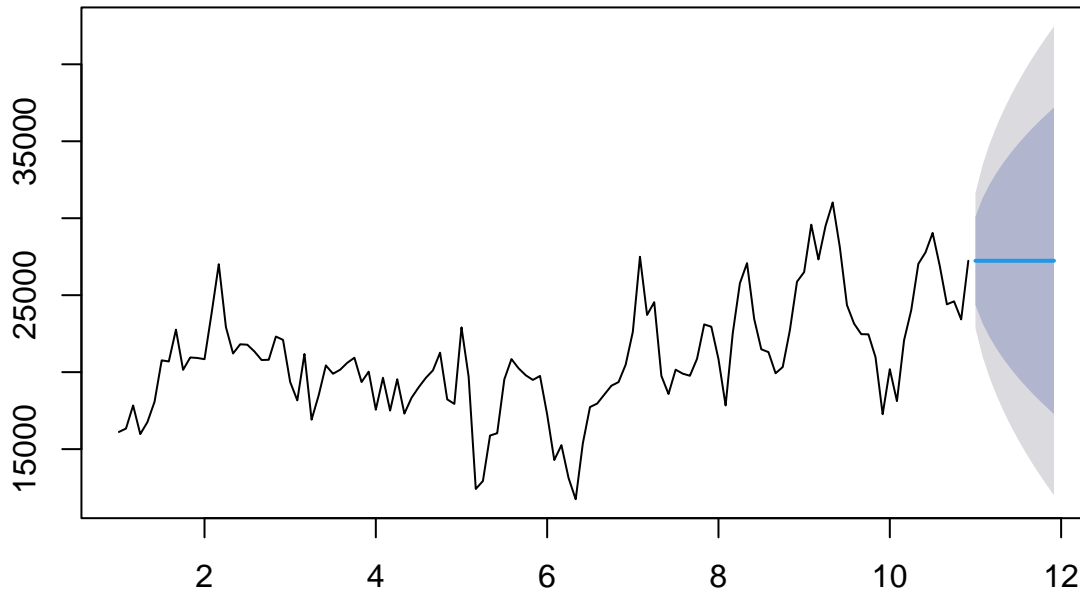
Fit a non-seasonal ARIMA(p, d, q) model using the `auto.arima()` function to the non-seasonal data. Forecast 12 months ahead of time using the `forecast()` function. Plot your forecasting results and further include on the plot the last year of non-seasonal data to compare with forecasted values (similar to the plot on the lesson file for M9).

```
#non-seasonal ARIMA
ARIMA_autofit <- auto.arima(deseason_inflow_data, max.D = 0, max.P = 0, max.Q = 0)
print(ARIMA_autofit)
```

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

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

Forecasts from ARIMA(0,1,0)



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.

Q5

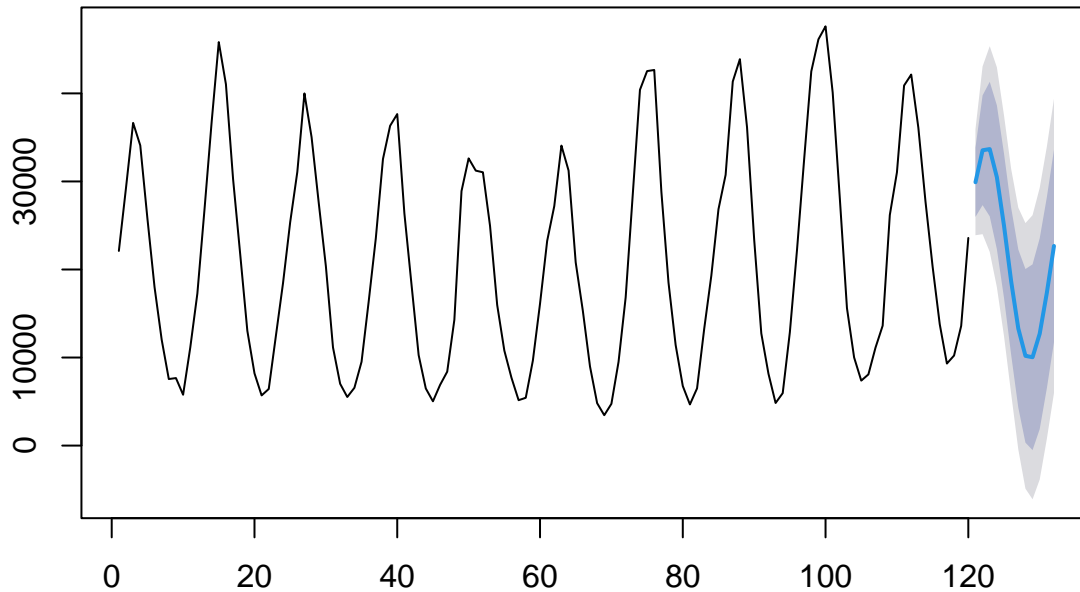
Repeat Q3 for the original data, but now fit a seasonal ARIMA(p, d, q) $x(P, D, Q)_{12}$ also using the `auto.arima()`.

```
SARIMA_autofit <- auto.arima(inflow_data[,3])
print(SARIMA_autofit)
```

```
## Series: inflow_data[, 3]
## ARIMA(2,0,1) with non-zero mean
##
## Coefficients:
##          ar1          ar2          ma1          mean
##          1.6854   -0.9457   -0.4578   20874.2709
## s.e.    0.0283    0.0266    0.0777    582.1295
##
## sigma^2 = 9421335:  log likelihood = -1133.95
## AIC=2277.91   AICc=2278.43   BIC=2291.85
```

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

Forecasts from ARIMA(2,0,1) with non-zero mean



Q6

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

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_inflow_data, h = 12, holdout = FALSE, silent = FALSE)
summary(SES_seas_fit)
```

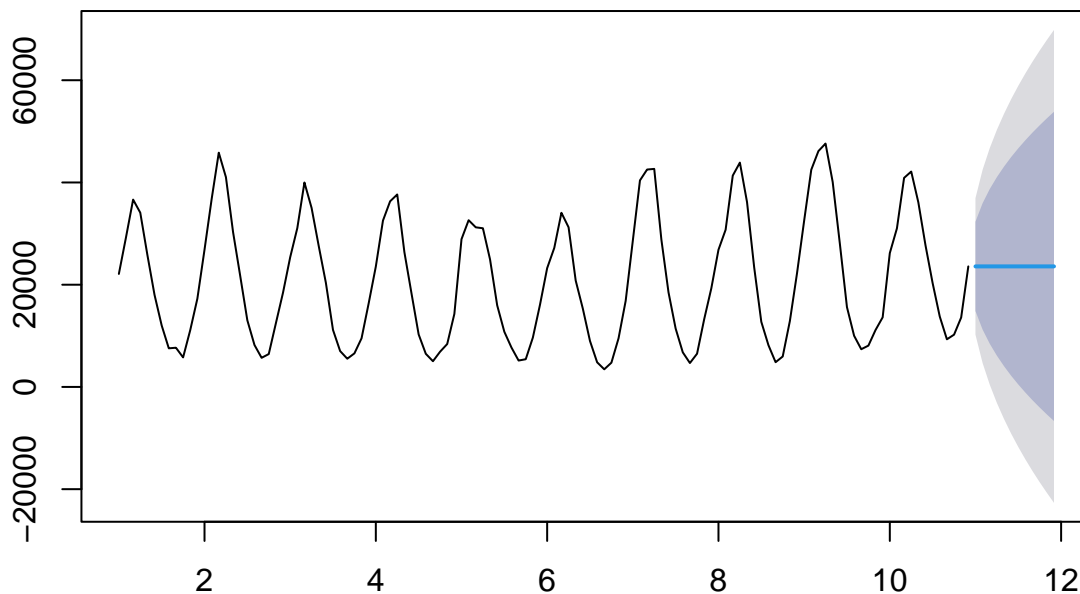
```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
## ses(y = ts_inflow_data, h = 12, holdout = FALSE, silent = FALSE)
##
## Smoothing parameters:
##   alpha = 0.9999
##
## Initial states:
##   l = 21885.2463
##
```



```
## 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 11      23582 14849.8153 32314.19 10227.276 36936.72
## Feb 11      23582 11233.4433 35930.56 4696.512 42467.49
## Mar 11      23582 8458.4205 38705.58 452.481 46711.52
## Apr 11      23582 6118.9402 41045.06 -3125.445 50289.45
## May 11      23582 4057.8032 43106.20 -6277.682 53441.68
## Jun 11      23582 2194.3852 44969.62 -9127.534 56291.53
## Jul 11      23582 480.7907 46683.21 -11748.251 58912.25
## Aug 11      23582 -1114.1874 48278.19 -14187.559 61351.56
## Sep 11      23582 -2612.2260 49776.23 -16478.612 63642.61
## Oct 11      23582 -4029.1079 51193.11 -18645.546 65809.55
## Nov 11      23582 -5376.7480 52540.75 -20706.583 67870.58
## Dec 11      23582 -6664.4029 53828.40 -22675.882 69839.88
```

```
plot(SES_seas_fit)
```

Forecasts from Simple exponential smoothing



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.

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.

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 you 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?