Rolling Window

Nikhil Gupta

2020-02-25 17:48:32

Table of Contents

[Setup 1](#_Toc34313553)

[Sliding Window 1](#_Toc34313554)

[Model 1 1](#_Toc34313555)

[Model 2 1](#_Toc34313556)

[Model 3 1](#_Toc34313557)

[Model 4 1](#_Toc34313558)

[Comparison 1](#_Toc34313559)

[Analysis of Variance 1](#_Toc34313560)

[Tukey HSD 1](#_Toc34313561)

# Setup

rm(list = ls()) # Remove existing variables

library(tswge)

## Warning: package 'tswge' was built under R version 3.5.3

# library(tswgewrapped)  
library(dplyr)

## Warning: package 'dplyr' was built under R version 3.5.3

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

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

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 3.5.3

## -- Attaching packages -------------------------------------------------------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.2.0 v readr 1.3.1  
## v tibble 2.1.3 v purrr 0.3.2  
## v tidyr 0.8.3 v stringr 1.4.0  
## v ggplot2 3.2.0 v forcats 0.4.0

## Warning: package 'ggplot2' was built under R version 3.5.3

## Warning: package 'tibble' was built under R version 3.5.3

## Warning: package 'tidyr' was built under R version 3.5.3

## Warning: package 'readr' was built under R version 3.5.2

## Warning: package 'purrr' was built under R version 3.5.3

## Warning: package 'stringr' was built under R version 3.5.3

## Warning: package 'forcats' was built under R version 3.5.3

## -- Conflicts ----------------------------------------------------------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(DT)

## Warning: package 'DT' was built under R version 3.5.3

library(ggplot2)  
source("common\_functions.R")

# sliding\_ase = function(x, phi = 0, theta = 0, d = 0, s = 0, n.ahead = NA, batch\_size = NA){  
# # Sliding CV ... batches are mutually exclusive  
#   
# n = length(x)  
#   
# if (is.na(batch\_size)){  
# warning("Batch Size has not been specified. Will assume a single batch")  
# cat("\n")  
# batch\_size = n  
# }  
#   
# if (is.na(n.ahead)){  
# stop("Number of points to be used for forecasting has not been specified. Please specify n.ahead")  
# }  
#   
# start = 1  
# num\_batches = n-batch\_size+1  
# ASEs = numeric(num\_batches)  
#   
# for (i in 0:(num\_batches-1))  
# {  
# subset = x[start:(batch\_size+i)]  
# test\_data = x[(batch\_size+i-n.ahead+1):(batch\_size+i)]  
#   
# # print(paste("i: ", i, "Start: ", start, " Stop: ", batch\_size+i))  
# # print(paste(" Test Start: ", (batch\_size+i-n.ahead+1), "Test End: ", (batch\_size+i)))  
#   
# forecasts = fore.aruma.wge(x = subset, phi = phi, theta = theta, d = d, s = s,  
# n.ahead = n.ahead, lastn = TRUE, plot = FALSE)  
# ASEs[i+1] = mean((test\_data - forecasts$f)^2)  
# start = start+1  
# }  
#   
# return(ASEs)  
# }

# Sliding Window

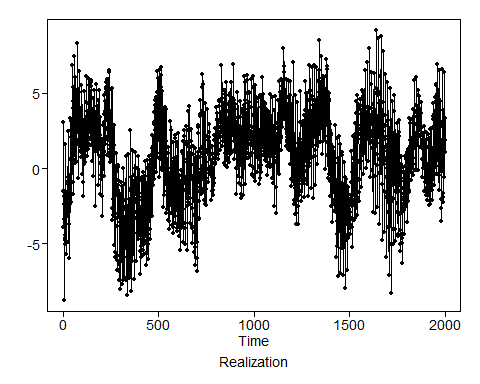
amtrack = read.csv(file = "../MSDS-6373-Time-Series/Unit 7/AmtrakPassengersMonthly.csv")  
amtrack %>% glimpse()

## Observations: 159  
## Variables: 5  
## $ Month <fct> Jan-91, Feb-91, Mar-91, Apr-91, May-91, Jun-91, Jul-...  
## $ Ridership <int> 1709, 1621, 1973, 1812, 1975, 1862, 1940, 2013, 1596...  
## $ X <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...  
## $ X.1 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...  
## $ X.2 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...

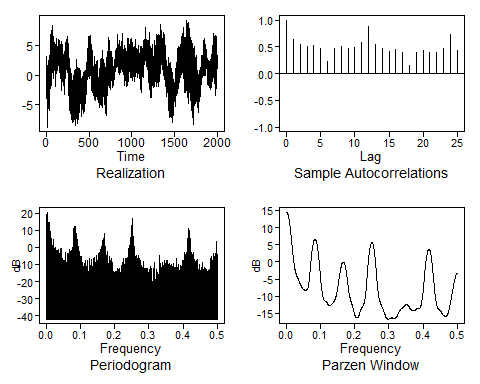
x = amtrack$Ridership  
n = length(x)  
n.ahead = 12  
  
results = tribble(~model, ~ASEs, ~ASE\_mean, ~ASE\_sd) # Placeholder for results

## Model 1

phi = c(0.5511, 0.1680, -0.0145, 0.0651, 0.1388, -0.2966, 0.1539, 0.1270, -0.1815, 0.0364, 0.1456, 0.6287, -0.3832, -0.0199, -0.1679)  
theta = 0  
d = 0  
s = 0  
  
# Compare data to a realization from the model  
xr = gen.aruma.wge(n = 2000, phi = phi, theta = theta, d = d, s = s, sn = 101)



px = plotts.sample.wge(xr)



# Without Sliding Window (usual method)  
  
# # Can use fore.aruma with d = 0 and s = 0 instead of using fore.arma. This yields the same results.  
# f = fore.arma.wge(x, phi=phi, theta = theta,  
# n.ahead = n.ahead, limits=FALSE, lastn = TRUE, plot = FALSE)  
# ase = mean((x[(n-n.ahead+1):n] - f$f)^2)  
# ase  
#   
# comparison = cbind(f$f, f$ll, f$ul, x[(n-n.ahead+1):n])  
# comparison = data.frame(comparison)  
# names(comparison) = c("Forecast", "Lower Limit", "Upper Limit", "Actual")  
# comparison = comparison %>%   
# mutate(half\_width = (`Upper Limit` - `Lower Limit`)/2,  
# perc\_error = round((Forecast - Actual) / Actual \* 100, 2),  
# ll\_satisfied = ifelse(Actual >= `Lower Limit`, TRUE, FALSE),  
# ul\_satisfied = ifelse(Actual <= `Upper Limit`, TRUE, FALSE),  
# within\_limits = ifelse(ll\_satisfied & ul\_satisfied, TRUE, FALSE))  
# comparison  
  
  
f = fore.aruma.wge(x, phi=phi, theta = theta, d = d, s = s,  
 n.ahead = n.ahead, limits=FALSE, lastn = TRUE, plot = FALSE)  
  
# comparison = cbind(f$f, f$ll, f$ul, x[(n-n.ahead+1):n])  
# comparison = data.frame(comparison)  
# names(comparison) = c("Forecast", "Lower Limit", "Upper Limit", "Actual")  
# comparison = comparison %>%   
# mutate(half\_width = (`Upper Limit` - `Lower Limit`)/2,  
# perc\_error = round((Forecast - Actual) / Actual \* 100, 2),  
# ll\_satisfied = ifelse(Actual >= `Lower Limit`, TRUE, FALSE),  
# ul\_satisfied = ifelse(Actual <= `Upper Limit`, TRUE, FALSE),  
# within\_limits = ifelse(ll\_satisfied & ul\_satisfied, TRUE, FALSE))  
# comparison  
  
ase = mean((x[(n-n.ahead+1):n] - f$f)^2)  
ase

## [1] 22588.56

# Default with sliding window function  
# Assumes only 1 batch, so results must match the one without sliding window (usual method) above  
ASEs = sliding\_ase(x, phi = phi, theta = theta, d = d, s = s, n.ahead = n.ahead)

## Warning in sliding\_ase(x, phi = phi, theta = theta, d = d, s = s, n.ahead =  
## n.ahead): Batch Size has not been specified. Will assume a single batch

ASEs

## [1] 22588.56

# With Sliding Window of batch\_size 50  
batch\_size = 50  
ASEs = sliding\_ase(x, phi = phi, theta = theta, d = d, s = s, n.ahead = n.ahead, batch\_size = batch\_size)  
# ASEs  
# mean(ASEs)  
# sd(ASEs)  
  
results = results %>% add\_row(model = "Model 1", ASEs = ASEs, ASE\_mean = mean(ASEs), ASE\_sd = sd(ASEs))

## Model 2

phi = c(-0.02709541, 0.74213105)  
theta = c(-0.5844596, 0.3836931)  
d = 0  
s = 12  
  
# Without Sliding Window (usual method)  
f = fore.aruma.wge(x, phi=phi, theta = theta, d = d, s = s,  
 n.ahead = n.ahead, limits=FALSE, lastn = TRUE, plot = FALSE)  
ase = mean((x[(n-n.ahead+1):n] - f$f)^2)  
ase

## [1] 17197.77

# Default with sliding window function  
# Assumes only 1 batch, so results must match the one without sliding window (usual method) above  
ASEs = sliding\_ase(x, phi = phi, theta = theta, d = d, s = s, n.ahead = n.ahead)

## Warning in sliding\_ase(x, phi = phi, theta = theta, d = d, s = s, n.ahead =  
## n.ahead): Batch Size has not been specified. Will assume a single batch

ASEs

## [1] 17197.77

# With Sliding Window of batch\_size 50  
batch\_size = 50  
ASEs = sliding\_ase(x, phi = phi, theta = theta, d = d, s = s, n.ahead = n.ahead, batch\_size = batch\_size)  
# ASEs  
# mean(ASEs)  
# sd(ASEs)  
  
results = results %>% add\_row(model = "Model 2", ASEs = ASEs, ASE\_mean = mean(ASEs), ASE\_sd = sd(ASEs))

## Model 3

phi = c(0.306943)  
theta = c(0.7431719)  
d = 1  
s = 12  
  
# Without Sliding Window (usual method)  
f = fore.aruma.wge(x, phi=phi, theta = theta, d = d, s = s,  
 n.ahead = n.ahead, limits=FALSE, lastn = TRUE, plot = FALSE)  
ase = mean((x[(n-n.ahead+1):n] - f$f)^2)  
ase

## [1] 18399.97

# Default with sliding window function  
# Assumes only 1 batch, so results must match the one without sliding window (usual method) above  
ASEs = sliding\_ase(x, phi = phi, theta = theta, d = d, s = s, n.ahead = n.ahead)

## Warning in sliding\_ase(x, phi = phi, theta = theta, d = d, s = s, n.ahead =  
## n.ahead): Batch Size has not been specified. Will assume a single batch

ASEs

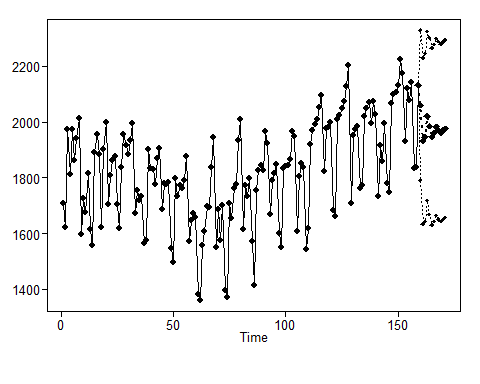
## [1] 18399.97

# With Sliding Window of batch\_size 50  
batch\_size = 50  
ASEs = sliding\_ase(x, phi = phi, theta = theta, d = d, s = s, n.ahead = n.ahead, batch\_size = batch\_size)  
# ASEs  
# mean(ASEs)  
# sd(ASEs)  
  
results = results %>% add\_row(model = "Model 3", ASEs = ASEs, ASE\_mean = mean(ASEs), ASE\_sd = sd(ASEs))

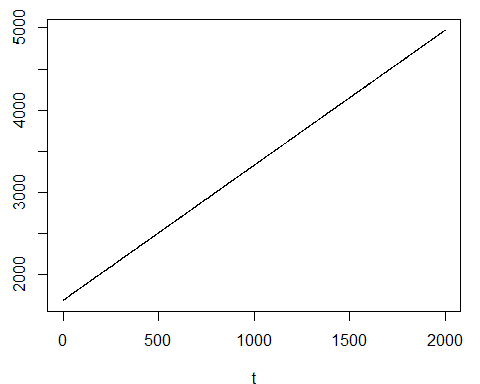
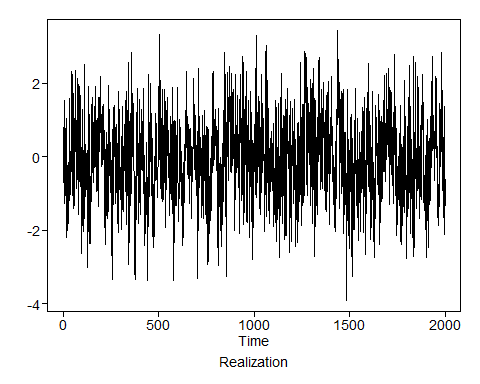
## Model 4

f = fore.sigplusnoise.wge(x = x, linear = TRUE, n.ahead = n.ahead, lastn = FALSE, limits = TRUE)

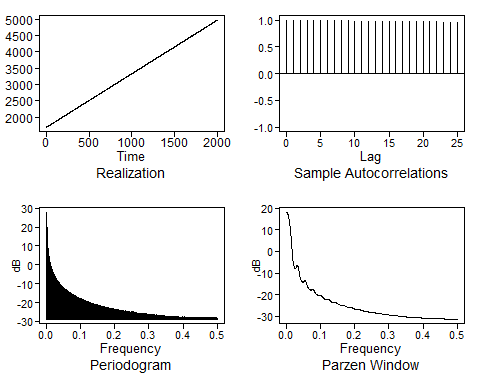
##   
## Coefficients of Original polynomial:   
## 0.4734 -0.0435 0.0667 0.2367 -0.1594   
##   
## Factor Roots Abs Recip System Freq   
## 1+0.0854B+0.5586B^2 -0.0765+-1.3358i 0.7474 0.2591  
## 1+0.6976B -1.4335 0.6976 0.5000  
## 1-1.2565B+0.4091B^2 1.5355+-0.2939i 0.6396 0.0301  
##   
##



xr = gen.sigplusnoise.wge(n = 2000, b0 = f$b0[['(Intercept)']], b1 = f$b0[['tl']], phi = f$phi.z, sn = 101)

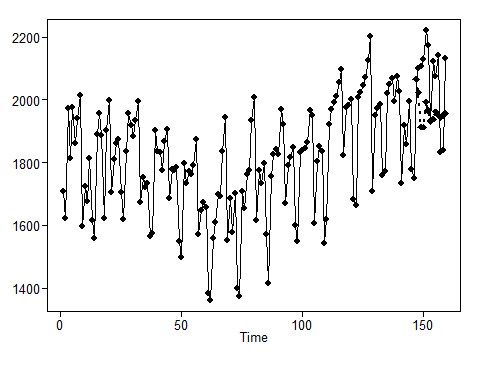


px = plotts.sample.wge(xr)



# ASE Calculation: Without Sliding Window (usual method)  
f = fore.sigplusnoise.wge(x = x, linear = TRUE, n.ahead = n.ahead, lastn = TRUE, limits = FALSE)

##   
## Coefficients of Original polynomial:   
## 0.4734 -0.0435 0.0667 0.2367 -0.1594   
##   
## Factor Roots Abs Recip System Freq   
## 1+0.0854B+0.5586B^2 -0.0765+-1.3358i 0.7474 0.2591  
## 1+0.6976B -1.4335 0.6976 0.5000  
## 1-1.2565B+0.4091B^2 1.5355+-0.2939i 0.6396 0.0301  
##   
##



# Without Sliding Window (usual method)  
ase = mean((x[(n-n.ahead+1):n] - f$f)^2)  
ase

## [1] 26921.49

# Default with sliding window function  
# Assumes only 1 batch, so results must match the one without sliding window (usual method) above  
ASEs = sliding\_ase(x, linear = TRUE, n.ahead = n.ahead)

## Warning in sliding\_ase(x, linear = TRUE, n.ahead = n.ahead): Batch Size has  
## not been specified. Will assume a single batch

##   
##   
## Coefficients of Original polynomial:   
## 0.4734 -0.0435 0.0667 0.2367 -0.1594   
##   
## Factor Roots Abs Recip System Freq   
## 1+0.0854B+0.5586B^2 -0.0765+-1.3358i 0.7474 0.2591  
## 1+0.6976B -1.4335 0.6976 0.5000  
## 1-1.2565B+0.4091B^2 1.5355+-0.2939i 0.6396 0.0301  
##   
##

ASEs

## [1] 26921.49

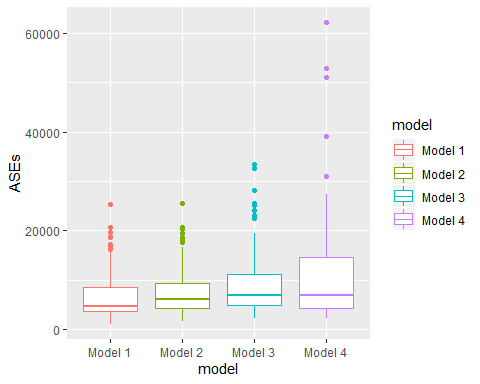
# With Sliding Window of batch\_size 50  
batch\_size = 50  
ASEs = sliding\_ase(x, linear = TRUE, d = d, s = s, n.ahead = n.ahead, batch\_size = batch\_size)  
# ASEs  
# mean(ASEs)  
# sd(ASEs)  
  
results = results %>% add\_row(model = "Model 4", ASEs = ASEs, ASE\_mean = mean(ASEs), ASE\_sd = sd(ASEs))

## Comparison

DT::datatable(results)

## PhantomJS not found. You can install it with webshot::install\_phantomjs(). If it is installed, please make sure the phantomjs executable can be found via the PATH variable.

ggplot(results, aes(x = model, y = ASEs, color = model)) + geom\_boxplot()



results %>%   
 group\_by(model) %>%   
 summarise(ASE\_mean = mean(ASE\_mean), ASE\_sd = mean(ASE\_sd))

## # A tibble: 4 x 3  
## model ASE\_mean ASE\_sd  
## <chr> <dbl> <dbl>  
## 1 Model 1 7024. 5163.  
## 2 Model 2 7623. 4693.  
## 3 Model 3 9313. 6708.  
## 4 Model 4 10980. 10373.

## Analysis of Variance

res.aov = aov(ASEs ~ model, data = results)  
summary(res.aov)

## Df Sum Sq Mean Sq F value Pr(>F)   
## model 3 1.049e+09 349679609 6.949 0.000141 \*\*\*  
## Residuals 436 2.194e+10 50319627   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Tukey HSD

TukeyHSD(res.aov)

## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = ASEs ~ model, data = results)  
##   
## $model  
## diff lwr upr p adj  
## Model 2-Model 1 599.4353 -1867.3593 3066.230 0.9234642  
## Model 3-Model 1 2289.3971 -177.3975 4756.192 0.0798095  
## Model 4-Model 1 3955.7887 1488.9941 6422.583 0.0002476  
## Model 3-Model 2 1689.9618 -776.8328 4156.756 0.2908210  
## Model 4-Model 2 3356.3534 889.5588 5823.148 0.0027859  
## Model 4-Model 3 1666.3916 -800.4030 4133.186 0.3031387