Frequency-based Ensemble Forecasting Model for Time Series Forecasting

Code

This is an R Markdown Notebook for a research paper titled **Frequency-based Ensemble Forecasting Model for Time Series Forecasting**.

Hide

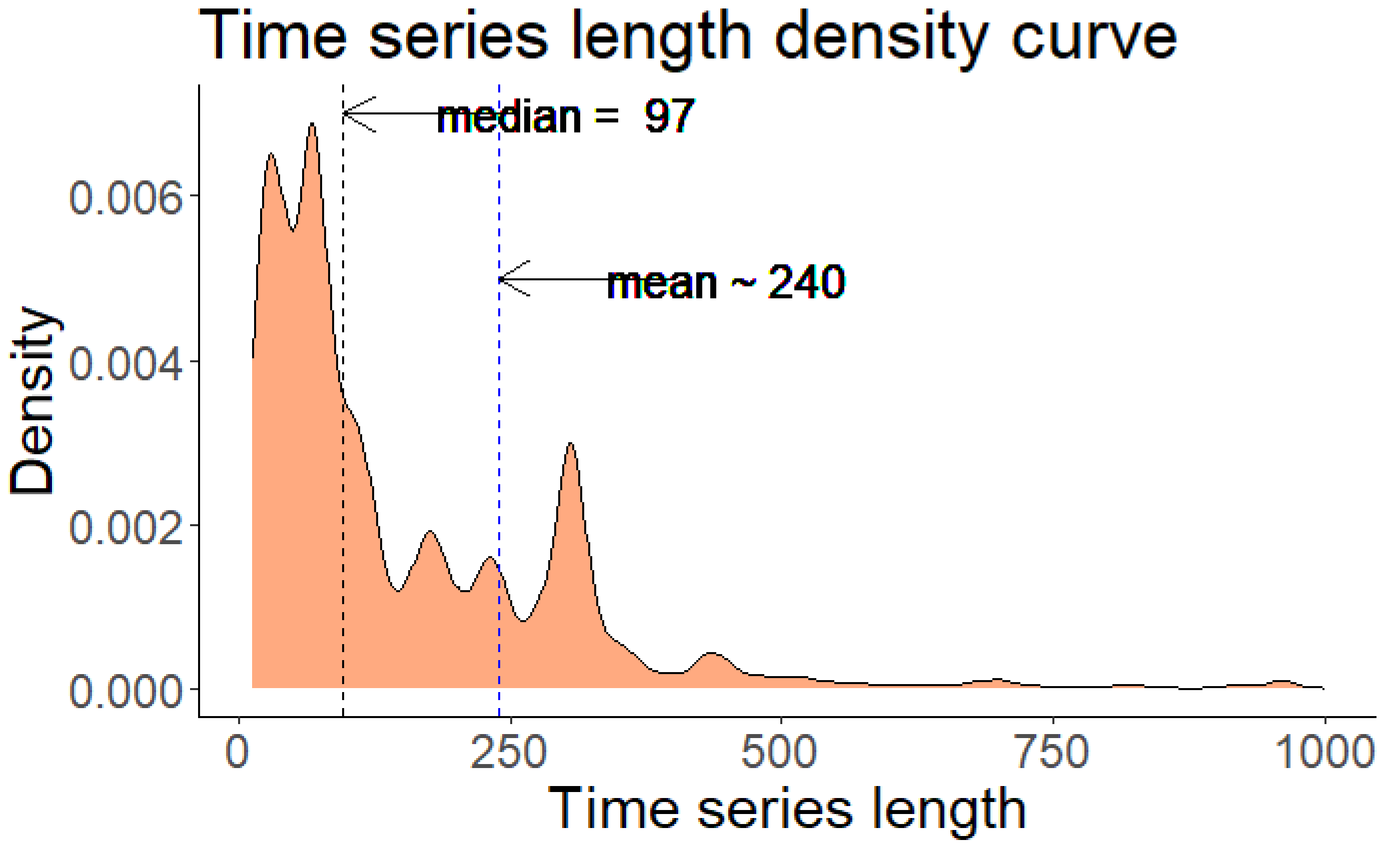
```
# Load needed packages

require(M4comp2018) # M4 data
require(ggplot2)
require(forecast)
library(opera)
```

Hide

```


# Load M4 data
data(M4)

# Calculate the length of each time series
tseries_count <- 100000
ts_length <- vector(length = tseries_count)

for (i in 1:tseries_count) {
  ts_length[i] <- length(M4[[i]]$x)
}
```

Hide

```
# Time series length density curve

mean_val <- mean(ts_length)
median_val <- median(ts_length)
df <- data.frame(x = ts_length[ts_length<=1000])

p <- ggplot(df, aes(x=x, fill="#ffaa80")) +
  geom_density(fill="#ffaa80") +
  geom_vline(aes(xintercept=mean_val), color="blue", linetype="dashed") +
  geom_vline(aes(xintercept=median_val), color="black", linetype="dashed") +
  labs(title="Time series length density curve", x="Time series length", y = "Density") +
  theme_classic()

p <- p + geom_segment(aes(x = median_val + 160, y = 0.007, xend = median_val, yend = 0.007),
                arrow = arrow(length = unit(0.5, "cm")))
p <- p + geom_text(x=median_val + 205, y= 0.007, size = 6, label=paste("median = ", as.character(median_va
l)))
p <- p + geom_segment(aes(x = mean_val+160, y = 0.005, xend = mean_val, yend = 0.005),
                    arrow = arrow(length = unit(0.5, "cm")))
p <- p + geom_text(x=mean_val+210, y=0.005, size = 6, label=paste("mean", "~", as.character(round(mean_val,
1))))
p <- p + theme(text = element_text(size=22))
p
```



Hide

```
print(paste("Number of time series used in the curve with length less than 1000 = ", nrow(df)))
```

```
[1] "Number of time series used in the curve with length less than 1000 =  97128"
```

Hide

```
print(paste("The mean of time series length for 100,000 time series = ", mean_val))
```

```
[1] "The mean of time series length for 100,000 time series =  240.02047"
```

Hide

```
print(paste("The median of time series length for 100,000 time series = ", median_val))
```

```
[1] "The median of time series length for 100,000 time series =  97"
```

Hide

```
print(paste("Number of time series with length less than the median = ", sum(df$x<=median_val)))
```

```
[1] "Number of time series with length less than the median =  50371"
```

Hide

```
# Calculate the percentages of using the used forecasting methods in the model
plot_percentages <- function(data, models_names){

  model_count <- ncol(data[[1]])
  data_count <- length(data)
  models <- matrix(nrow = data_count, ncol = model_count)
  point_count <-  nrow(data[[1]])

  for (data_id in 1:data_count){
    for(model_id in 1:model_count){
      models[data_id,model_id] = sum(data[[data_id]][,model_id]>0)/point_count
    }
  }

  results <- data.frame(Models=models_names, Percentages=round(colSums(models)/data_count*100,0))
  print(results)
  # ggplot(data=results,aes(x=Models,y=Percentages,fill=Models)) +
  #   geom_bar(stat="identity", width=0.5) +
  #   geom_text(aes(label=paste(as.character(Percentages),"%")),size=6, nudge_y=0.5) +
  #   theme(text = element_text(size=24)) + scale_x_discrete(labels = NULL, breaks = NULL) + labs(x = "") +
  #   ylab("Percentage") + theme(legend.position="top")+ scale_fill_brewer(palette = "Spectral")
}
```

Hide

```
# load opera data then plot the figures
load("./opera_data.RData")

plot_percentages (hourly_opera, c("THIEF+ ETS forecasts","THIEF+ ARIMA forecasts","THIEF+ Naïve S forecast
s","TBATS"))
```

Models <chr>	Percentages <dbl>
THIEF+ ETS forecasts	64
THIEF+ ARIMA forecasts	69
THIEF+ Naïve S forecasts	50
TBATS	46

4 rows

Hide

```
plot_percentages (daily_opera, c("ETS","BATS","Bagged ETS"))
```

Models <chr>	Percentages <dbl>
ETS	52
BATS	56
Bagged ETS	51

3 rows

Hide

```
plot_percentages (weekly_opera, c("Regression with ARIMA errors","TBATS"))
```

Models <chr>	**Percentages** <dbl>
Regression with ARIMA errors	67
TBATS	64

2 rows

Hide

```
plot_percentages (monthly_opera, c("THIEF+ ETS forecasts","THIEF+ ETS forecasts ","BATS","TBATS","ETS"))
```

Models <chr>	**Percentages** <dbl>
THIEF+ ETS forecasts	54
THIEF+ ETS forecasts	45
BATS	53
TBATS	53
ETS	50

5 rows

Hide

```
plot_percentages (qarterly_opera, c("THIEF+ ETS forecasts","THIEF+ ETS forecasts ","BATS","TBATS","ETS"))
```

Models <chr>	**Percentages** <dbl>
THIEF+ ETS forecasts	56
THIEF+ ETS forecasts	52
BATS	60
TBATS	59
ETS	58

5 rows

Hide

```
plot_percentages (weekly_opera, c("TBATS","ETS"))
```

Models <chr>	**Percentages** <dbl>

Models	Percentages
<chr>	<dbl>
TBATS	67
ETS	64

2 rows

Hide

```
smape_cal <- function(outsample, forecasts){
  #Used to estimate sMAPE
  outsample <- as.numeric(outsample) ; forecasts<-as.numeric(forecasts)
  smape <- (abs(outsample-forecasts)*200)/(abs(outsample)+abs(forecasts))
  return(smape)
}

Daily_OPERA_1step <- function(training1, validation, training2, outofsample, fh){

  set.seed(2019)
  models <- 3
  experts_1 <- matrix(NA, nrow= fh, ncol= models)
  experts_2 <- matrix(NA, nrow= length(training2), ncol= models)
  forecasts <- matrix(NA, nrow= fh, ncol= models)

  # fit using training1 data
  fitETS1 <- ets(training1)
  fitBATS1 <- bats(training1, biasadj = FALSE)
  fitBaggedETS1 <- baggedETS(ts(as.vector(training1)))

  # forecast future values to use it with the validation data
  experts_1[,1] <- forecast(fitETS1 ,h=fh)$mean
  experts_1[,2] <- forecast(fitBATS1 ,h=fh)$mean
  experts_1[,3] <- forecast(fitBaggedETS1 ,h=fh)$mean

  # fit the selected models using training2 set = training 1 + validation set
  fitETS_1 <- ets(training2)
  fitBATS_1 <- bats(training2, biasadj = FALSE)
  fitBaggedETS_1 <- baggedETS(ts(as.vector(training2)))

  # use the fitted data as expert forecasts
  experts_2[,1] <- fitETS_1$fitted
  experts_2[,2] <- fitBATS_1$fitted
  experts_2[,3] <- fitBaggedETS_1$fitted


  # builds the aggregation rule
  MLpol0 <- mixture(model = "MLpol", loss.type = "square")

  # use opera to find the weights using the online mode with the validation set
  x_1 <- cbind(ETS = experts_1[,1], BATS = experts_1[,2], BAGGEDETS = experts_1[,3])
  MLpol <- predict(MLpol0, newexpert = x_1, newY = as.vector(validation), online = TRUE, type="all")
  OPERA <- list(MLpol)

  # use opera to find the weights using the online mode with the fitted data
  x_2 <- cbind(ETS = experts_2[,1], BATS = experts_2[,2], BAGGEDETS = experts_2[,3])
  MLpol <- predict(MLpol0, newexpert = x_2, newY = as.vector(training2), online = TRUE, type="all")


  # forecast h points
  fitETS2 <- ets(outofsample , model= fitETS_1, use.initial.values=TRUE)
  forecasts[,1] <- fitted(fitETS2)

  fitBATS2 <- bats(outofsample, biasadj = FALSE, model= fitBATS_1)
  forecasts[,2] <- fitted(fitBATS2)

  fBaggedETS <- matrix(nrow = length(outofsample), ncol = 100)
```

```
  for (i in 1:100) {
    fitx <- ets(outofsample , model = fitBaggedETS_1$models[[i]], use.initial.values=TRUE)
    onestepx <- fitted(fitx)
    fBaggedETS[,i] <- onestepx
  }
  forecasts[,3] <- rowMeans(fBaggedETS)

  # calculate final forecasts by multiplying the obtained forecasts* weights
  finalforecasts_1 <- finalforecasts_2 <- numeric(fh)
  mean_val <- median_val <- numeric(fh)
  for(j in 1:fh){
    finalforecasts_1[j]<-round(sum(OPERA[[1]]$weights[j,] * forecasts[j,]),4)

    median_val[j] <-  median(forecasts[j,])
    mean_val[j] <-  mean(forecasts[j,])
  }
  finalforecasts_1 = as.numeric(finalforecasts_1)

  # offline mode
  finalforecasts_2<- as.numeric(predict(MLpol0, newexperts = forecasts, online = FALSE, type = 'response'))

  outofsample = as.numeric(outofsample)


  print (paste("RMSE with OPERA = ", round(accuracy(finalforecasts_1,outofsample)[2],6)))
  print (paste("RMSE with OPERA offline = ", round(accuracy(finalforecasts_2,outofsample)[2],6)))
  print (paste("RMSE with average = ", round(accuracy(mean_val,outofsample)[2],6)))
  print (paste("RMSE with median = ", round(accuracy(median_val,outofsample)[2],6)))

  return(list(f1 = finalforecasts_1, f2 = finalforecasts_2))
}
```

Hide

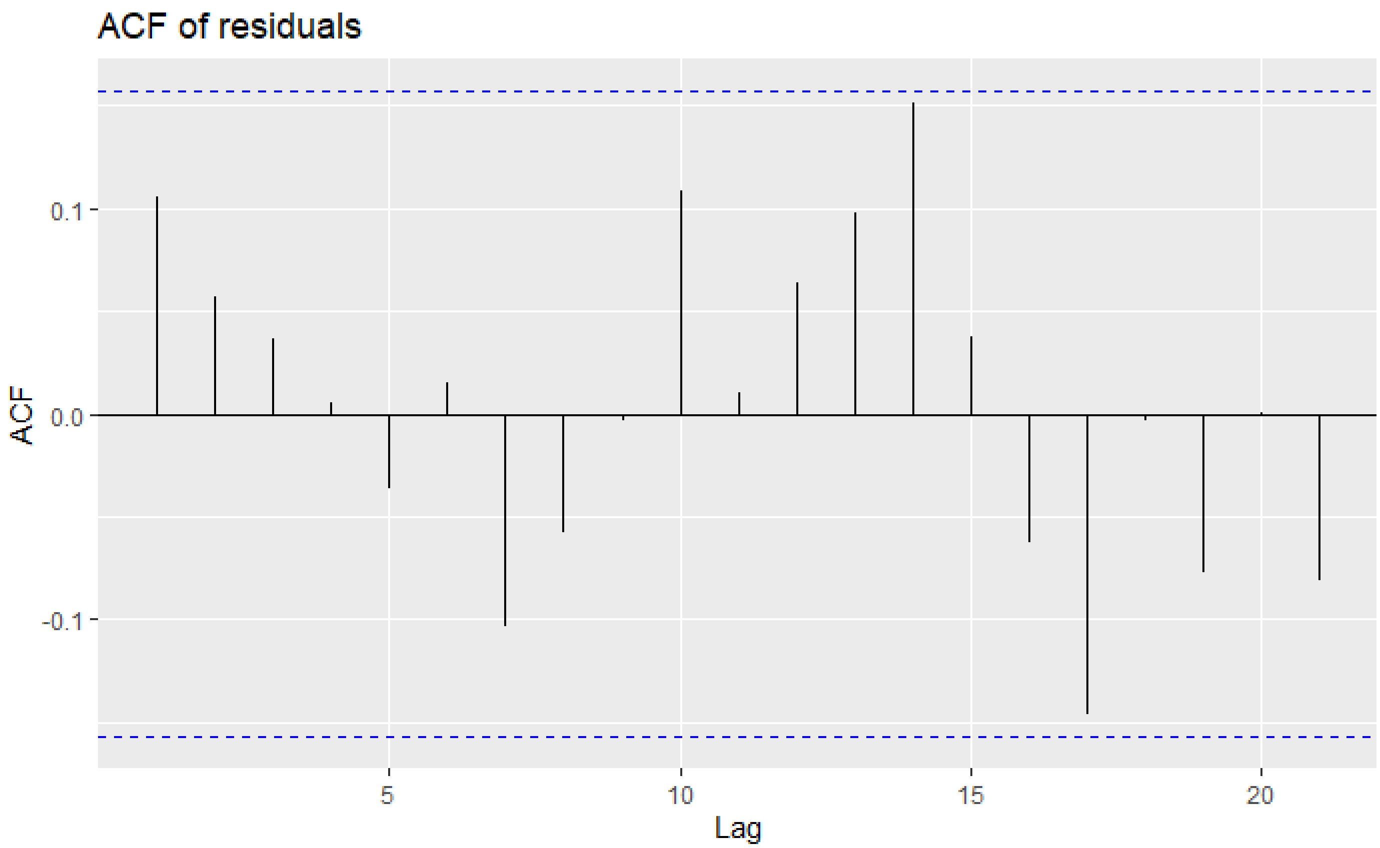
```
# one-step ahead forecasts for EUR/USD time series

load("./eur.RData")

ts <- ts(ts, frequency = 5)

training1 <- window(ts, end= c(94,4))
validation <- window(ts, start=c(94,5), end= c(125,5))

training2 <- window(ts, end= c(125,5))
outofsample <- window(ts, start= c(126,1))
fh <- 156

f = Daily_OPERA_1step(training1, validation, training2, outofsample, fh)
```

```
You should provide observations to train non trivial model
```

```
[1] "RMSE with OPERA =  0.002923"
[1] "RMSE with OPERA offline =  0.002962"
[1] "RMSE with average =  0.002962"
[1] "RMSE with median =  0.002953"
```

Hide

```
res1 <- outofsample - f[[1]]
ggAcf(res1) + ggtitle("ACF of residuals")
```



Hide

```
Box.test(res1,lag=10, fitdf=0, type="Lj")
```

```

	Box-Ljung test

data:  res1
X-squared = 7.0884, df = 10, p-value = 0.7171
```

Hide

```r
Daily_OPERA_hstep <- function(training1, validation, training2, outofsample, original, fh){

  set.seed(2019)
  models <- 3
  experts_1 <- matrix(NA, nrow= fh, ncol= models)
  experts_2 <- matrix(NA, nrow= length(training2), ncol= models)
  forecasts <- matrix(NA, nrow= fh, ncol= models)

  # fit using training1 data
  fitETS1 <- ets(training1)
  fitBATS1 <- bats(training1, biasadj = FALSE)
  fitBaggedETS1 <- baggedETS(ts(as.vector(training1)))

  # forecast future values to use it with the validation data
  experts_1[,1] <- forecast(fitETS1 ,h=fh)$mean
  experts_1[,2] <- forecast(fitBATS1 ,h=fh)$mean
  experts_1[,3] <- forecast(fitBaggedETS1 ,h=fh)$mean

  # fit the selected models using training2 set = training 1 + validation set
  fitETS_1 <- ets(training2)
  fitBATS_1 <- bats(training2, biasadj = FALSE)
  fitBaggedETS_1 <- baggedETS(ts(as.vector(training2)))

  # use the fitted data as expert forecasts
  experts_2[,1] <- fitETS_1$fitted
  experts_2[,2] <- fitBATS_1$fitted
  experts_2[,3] <- fitBaggedETS_1$fitted


  # builds the aggregation rule
  MLpol0 <- mixture(model = "MLpol", loss.type = "square")

  # use opera to find the weights using the online mode with the validation set
  x_1 <- cbind(ETS = experts_1[,1], BATS = experts_1[,2], BAGGEDETS = experts_1[,3])
  MLpol <- predict(MLpol0, newexpert = x_1, newY = as.vector(validation), online = TRUE, type="all")
  OPERA <- list(MLpol)

  # use opera to find the weights using the online mode with the fitted data
  x_2 <- cbind(ETS = experts_2[,1], BATS = experts_2[,2], BAGGEDETS = experts_2[,3])
  MLpol <- predict(MLpol0, newexpert = x_2, newY = as.vector(training2), online = TRUE, type="all")


  # forecast h points
  forecasts[,1] <- forecast(fitETS_1 ,h=fh)$mean
  forecasts[,2] <- forecast(fitBATS_1 ,h=fh)$mean
  forecasts[,3] <- forecast(fitBaggedETS_1 ,h=fh)$mean


  # calculate final forecasts by multiplying the obtained forecasts* weights
  finalforecasts_1 <- finalforecasts_2 <- numeric(fh)
  mean_val <- median_val <- numeric(fh)
  for(j in 1:fh){
    finalforecasts_1[j]<-round(sum(OPERA[[1]]$weights[j,] * forecasts[j,]),4)

    median_val[j] <-  median(forecasts[j,])
    mean_val[j] <-  mean(forecasts[j,])
  }
```

```
  # offline mode
  finalforecasts_2<- predict(MLpol0, newexperts = forecasts, online = FALSE, type = 'response')

  finalforecasts_1 = as.numeric(finalforecasts_1)
  finalforecasts_2 = as.numeric(finalforecasts_2)

  finalforecasts_1 = original[1:14] + finalforecasts_1
  finalforecasts_2 = original[1:14] + finalforecasts_2
  mean_val = original[1:14] + mean_val
  median_val = original[1:14] + median_val

  print (paste("SMAPE with OPERA = ", round(mean(smape_cal(original[2:15], finalforecasts_1)),6)))
  print (paste("SMAPE with OPERA offline = ", round(mean(smape_cal(original[2:15], finalforecasts_2)),6)))
  print (paste("SMAPE with average = ", round(mean(smape_cal(original[2:15], mean_val)),6)))
  print (paste("SMAPE with median = ", round(mean(smape_cal(original[2:15], median_val)),6)))

  return(list(f1 = finalforecasts_1, f2 = finalforecasts_2))
}
```

Hide

```
# multi-step ahead forecasts for Bitcoin time series

load("./bitcoin.RData")

print ("1 year data")
```

```
[1] "1 year data"
```

Hide

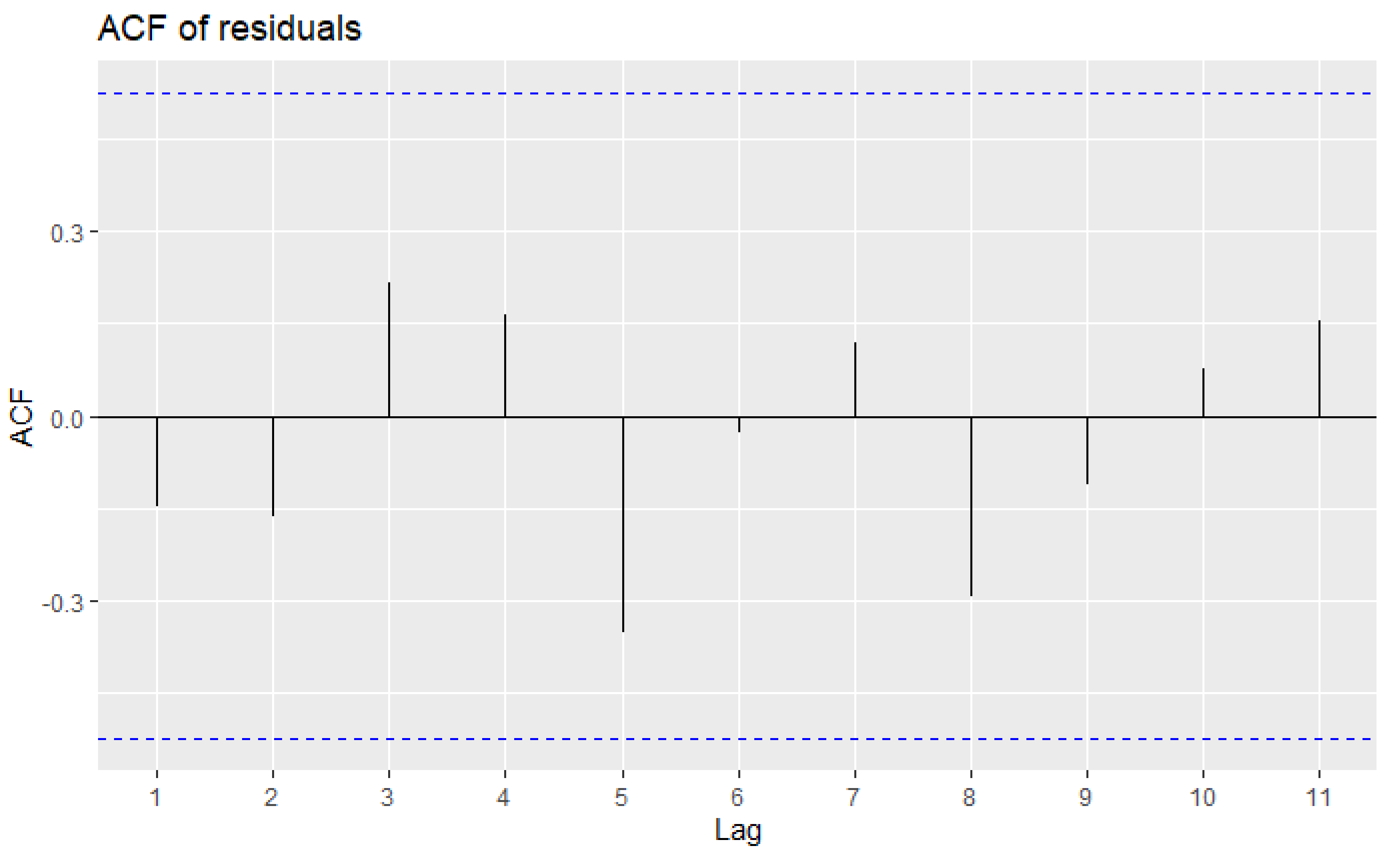
```
#original = as.numeric(c(tsClose_5years$insample,tsClose_5years$outsample[1:13]))

#original = as.numeric(c(tsClose_5years$insample[1813],tsClose_5years$outsample))
#diff_data = diff(ts(c(tsClose_5years$insample,tsClose_5years$outsample), frequency = 7))

#training1 <- window(diff_data, end = c(257, 7))
#validation <- window(diff_data, start=c(258, 1), end = c(259, 7))

#training2 <- window(diff_data, end = c(259, 7))
#outofsample <- window(diff_data, start = c(260, 1))

original = as.numeric(c(Close_1year_insample[352],out_of_sample))
diff_data = diff(ts(c(Close_1year_insample,out_of_sample), frequency = 7))

training1 <- window(diff_data, end = c(49, 2))
validation <- window(diff_data, start=c(49, 3), end = c(51, 2))

training2 <- window(diff_data, end = c(51, 2))
outofsample <- window(diff_data, start = c(51, 3))

fh <- 14


f <- Daily_OPERA_hstep(training1, validation, training2, outofsample, original, fh)
```

```
You should provide observations to train non trivial model
```

```
[1] "SMAPE with OPERA =  2.334444"
[1] "SMAPE with OPERA offline =  2.365618"
[1] "SMAPE with average =  2.365618"
[1] "SMAPE with median =  2.298935"
```

Hide

```
res1 <- original[2:15] - f[[1]]
ggAcf(res1) + ggtitle("ACF of residuals")
```



Hide

```
Box.test(res1,lag=10, fitdf=0, type="Lj")
```

```
	Box-Ljung test

data:  res1
X-squared = 10.142, df = 10, p-value = 0.4282
```