No Pot of Gold at the End of the Rainbow

Can we forecast gold prices using classical time series models?

June 5, 2016

This work project was developed by Dário Soares ([m2015315@novaims.unl.pt](mailto:m2015315@novaims.unl.pt)), Francisco Lourenço ([m2015187@novaims.unl.pt](mailto:m2015187@novaims.unl.pt)), Hugo Ricardo ([m2015343@novaims.unl.pt](mailto:m2015343@novaims.unl.pt)) and Ricardo Dias ([m2015058@novaims.unl.pt](mailto:m2015058@novaims.unl.pt))

# Background

# Methodology

# Data

Describe the data...

# Load data  
gold <- read.csv("project\_gold\_gld\_daily.txt",header=F, sep=",", as.is=TRUE, skip=54, col.names = c("Date", "Value"))  
  
# Set as time-series  
gold.ts0 <- ts(gold$Value)  
  
# Remove last 6 observations  
gold.ts <- head(gold.ts0,-6)  
length(gold.ts0)

[1] 2845

length(gold.ts)

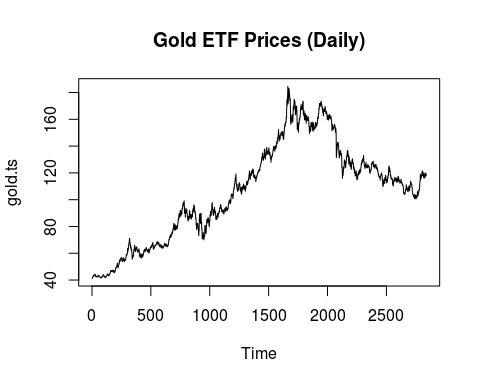
[1] 2839

# Exploratory Analysis

## Stationarity

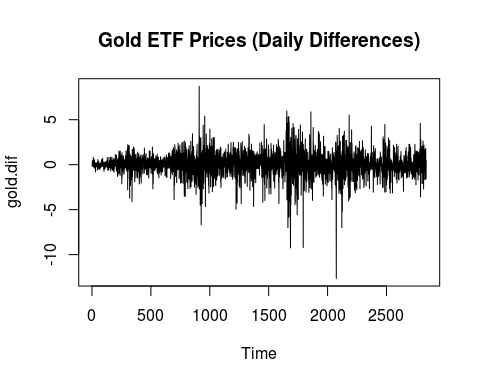
We started by plotting the original time series of Gold ETF prices in USD.

plot.ts(gold.ts, main="Gold ETF Prices (Daily)")



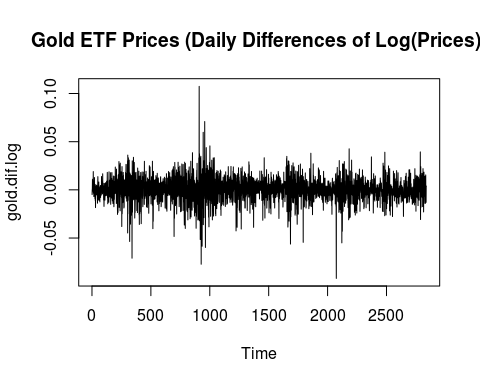
The time series *is not* weakly stationary as the level and variance change over time. Because of this we cannot apply directly the Box-Jenkins methodology, so we will evaluate if first differences are sufficient to turn the time series stationary.

gold.dif <- diff(gold.ts)  
plot.ts(gold.dif, main="Gold ETF Prices (Daily Differences)")



Although the result can be considered adequate concerning the level stationarity, issues regarding [MORE] variance remain. Therefore, we will apply a log transformation [MORE] and a subsequent first difference.

gold.dif.log <- diff(log(gold.ts))  
plot.ts(gold.dif.log, main="Gold ETF Prices (Daily Differences of Log(Prices))")



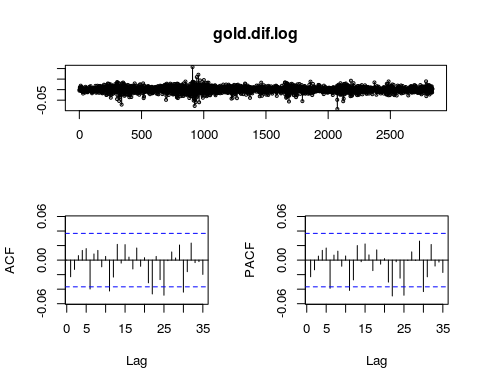
Unfortunately the log transformation did not overcome the variance issues. Later on we will try to fit a GARCH model. For now, we will follow a pragmatic approach: we will try to fit an ARIMA model nonetheless and evaluate at the end the temporal structure of the residuals.

## Seasonality

Theoretically seasonality should not be presented in a financial time series as it would mean an arbritrage edge.

## ARIMA

tsdisplay(gold.dif.log)



Acf(gold.dif.log, plot = FALSE)

Autocorrelations of series 'gold.dif.log', by lag  
  
 0 1 2 3 4 5 6 7 8 9   
 1.000 -0.023 -0.013 0.006 0.013 0.016 -0.040 0.008 0.013 -0.009   
 10 11 12 13 14 15 16 17 18 19   
 0.005 -0.042 -0.023 0.022 -0.004 0.021 0.004 -0.012 0.017 -0.009   
 20 21 22 23 24 25 26 27 28 29   
 0.003 -0.031 -0.047 0.005 -0.027 -0.048 -0.001 0.011 0.003 0.021   
 30 31 32 33 34   
-0.044 -0.016 0.023 -0.003 -0.002

Pacf(gold.dif.log, plot = FALSE)

Partial autocorrelations of series 'x', by lag  
  
 0 1 2 3 4 5 6 7 8 9   
-0.023 -0.013 0.006 0.013 0.017 -0.039 0.007 0.012 -0.008 0.006   
 10 11 12 13 14 15 16 17 18 19   
-0.042 -0.027 0.020 -0.002 0.022 0.007 -0.014 0.014 -0.006 0.002   
 20 21 22 23 24 25 26 27 28 29   
-0.030 -0.049 -0.002 -0.025 -0.048 -0.001 0.011 0.000 0.026 -0.043   
 30 31 32 33   
-0.023 0.021 -0.008 -0.003

gold.log <- log(gold.ts)  
auto.arima(gold.log)

Series: gold.log   
ARIMA(0,1,0) with drift   
  
Coefficients:  
 drift  
 4e-04  
s.e. 2e-04  
  
sigma^2 estimated as 0.0001574: log likelihood=8399.02  
AIC=-16794.04 AICc=-16794.04 BIC=-16782.14

# ARIMA(0,1,0) with drift   
# Random walk

Box.test(gold.dif.log, lag = 20, type = "Ljung-Box")

##   
## Box-Ljung test  
##   
## data: gold.dif.log  
## X-squared = 19.676, df = 20, p-value = 0.4783

p-value = 0.4783 There is no well defined temporal structure in the transformed data(gold.dif.log) The auto.arima points to a random walk. The Ljung-Box test applied to the transformed data does not reject the null, with a p-value of 0.4783. Given these assertions we will fit a linear model to the transformed data with only the intercept in order to impose a zero mean series. Then we will analyse the ACF and PACF of the linear model residuals and decide for a GARCH case.

fitlin <- Arima(gold.dif.log, order = c(0,0,0))  
summary(fitlin)

## Series: gold.dif.log   
## ARIMA(0,0,0) with non-zero mean   
##   
## Coefficients:  
## intercept  
## 4e-04  
## s.e. 2e-04  
##   
## sigma^2 estimated as 0.0001574: log likelihood=8399.02  
## AIC=-16794.04 AICc=-16794.04 BIC=-16782.14  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set -2.301834e-15 0.01254511 0.008865689 -Inf Inf 0.6658542  
## ACF1  
## Training set -0.0228644

The intercept is statisticaly significant for a 5% error type I. We will try to fit a Arch/Garch as it follows

log(yt) - log(yt-1) = beta0 + epsilon

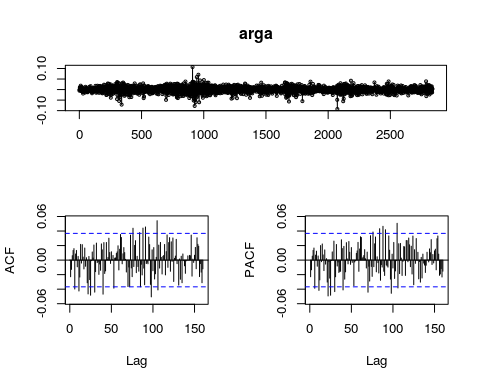
or

Yt/yt-1 = e^(beta0 + epsilon)

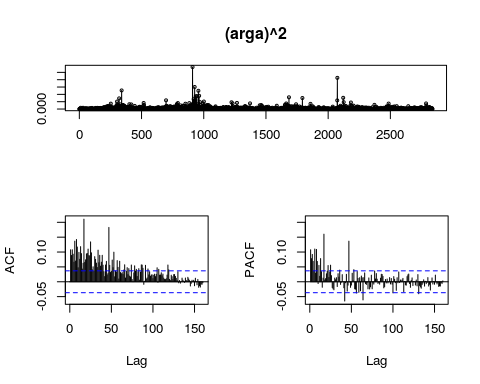
(nota: epsilon = arga)

# ARCH-GARCH

arga <- fitlin$residuals  
tsdisplay(arga, lag.max = 160)



tsdisplay((arga)\*\*2, lag.max = 160)



From the analysis of the ACF and PACF of the squares of the linear model residuals (arga) we conclude that there is a case for a changing variance over time. As the pattern is conducive to a ARMA model we will do a simulation with multiple GARCH models to assist us on the fit decision.

|  |  |  |
| --- | --- | --- |
| Model | AIC | Significance |
| GARCH | 12 | 3 |
| GARCH(1,1) | -6.096633 | YES |
| GARCH(1,0) | -5.944167 | ? |
| GARCH(2,0) | -5.971479 | ? |
| GARCH(3,0) | -5.996618 | ? |
| GARCH(4,0) | -6.014606 | ? |
| GARCH(5,0) | -6.025685 | ? |
| GARCH(6,0) | -6.050757 | ? |
| GARCH(2,1) | -6.095755 | ? |
| GARCH(3,1) | -6.094847 | ? |
| GARCH(4,1) | -6.093946 | ? |
| GARCH(5,1) | -6.093174 | ? |
| GARCH(6,1) | -6.092338 | ? |
| GARCH(1,2) | -6.095783 | ? |
| GARCH(1,3) | -6.094890 | ? |
| GARCH(1,4) | -6.094029 | ? |
| GARCH(1,5) | -6.093382 | ? |
| GARCH(1,6) | -6.092544 | ? |
| GARCH(2,2) | -6.096449 | ? |
| GARCH(2,3) | -6.095632 | ? |
| GARCH(2,4) | -6.094740 | ? |
| GARCH(2,5) | -6.094125 | ? |
| GARCH(2,6) | -6.093348 | ? |
| GARCH(3,2) | -6.095585 | ? |
| GARCH(3,3) | -6.094197 | ? |
| GARCH(3,4) | -6.093292 | ? |
| GARCH(3,5) | -6.093531 | ? |
| GARCH(3,6) | -6.092720 | ? |
| GARCH(4,2) | -6.094991 | ? |
| GARCH(4,3) | -6.093292 | ? |
| GARCH(4,4) | -6.093847 | ? |
| GARCH(4,5) | -6.093599 | ? |
| GARCH(4,6) | -6.092924 | ? |
| GARCH(5,2) | -6.094174 | ? |
| GARCH(5,3) | -6.092448 | ? |
| GARCH(5,4) | -6.093027 | ? |
| GARCH(5,5) | -6.093050 | ? |
| GARCH(5,6) | -6.092220 | ? |
| GARCH(6,2) | -6.093563 | ? |
| GARCH(6,3) | -6.093281 | ? |
| GARCH(6,4) | -6.093205 | ? |
| GARCH(6,5) | -6.093205 | ? |
| GARCH(6,6) | -6.093019 | ? |

From this simulation we concluded the est model was a garch(1,1), according to the parsimony principle

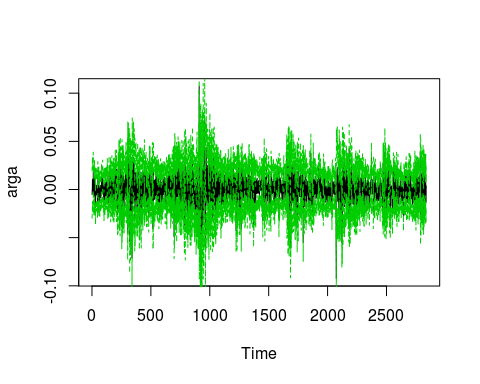
FALTA: especificação do GARCH

gfit <- garchFit(~garch(1,1), arga, trace=F)  
summary(gfit)

##   
## Title:  
## GARCH Modelling   
##   
## Call:  
## garchFit(formula = ~garch(1, 1), data = arga, trace = F)   
##   
## Mean and Variance Equation:  
## data ~ garch(1, 1)  
## <environment: 0x419b330>  
## [data = arga]  
##   
## Conditional Distribution:  
## norm   
##   
## Coefficient(s):  
## mu omega alpha1 beta1   
## -1.8078e-15 1.9279e-06 5.7395e-02 9.3103e-01   
##   
## Std. Errors:  
## based on Hessian   
##   
## Error Analysis:  
## Estimate Std. Error t value Pr(>|t|)   
## mu -1.808e-15 1.976e-04 0.000 1   
## omega 1.928e-06 4.891e-07 3.942 8.08e-05 \*\*\*  
## alpha1 5.739e-02 7.288e-03 7.875 3.33e-15 \*\*\*  
## beta1 9.310e-01 8.606e-03 108.183 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Log Likelihood:  
## 8655.123 normalized: 3.049726   
##   
## Description:  
## Sun Jun 5 17:43:25 2016 by user: dario   
##   
##   
## Standardised Residuals Tests:  
## Statistic p-Value   
## Jarque-Bera Test R Chi^2 748.1844 0   
## Shapiro-Wilk Test R W 0.9771714 0   
## Ljung-Box Test R Q(10) 5.21801 0.8761469  
## Ljung-Box Test R Q(15) 11.53687 0.7136916  
## Ljung-Box Test R Q(20) 12.36139 0.9030757  
## Ljung-Box Test R^2 Q(10) 11.668 0.3078906  
## Ljung-Box Test R^2 Q(15) 16.75008 0.3340229  
## Ljung-Box Test R^2 Q(20) 20.89322 0.4034416  
## LM Arch Test R TR^2 13.70953 0.3196408  
##   
## Information Criterion Statistics:  
## AIC BIC SIC HQIC   
## -6.096633 -6.088246 -6.096637 -6.093608

The tests results were good excepted for normality. The fit wasn't able to recover and capture the full extreme values dynamics. Anyway we pragmatically will go to forecasting.

u <- gfit@sigma.t  
argaplusu <- arga+1.96\*u  
argaminusu <- arga-1.96\*u  
plot(arga)  
lines(argaplusu, lty= 2,col = 3)  
lines(argaminusu, lty= 2,col = 3)



x <- predict(gfit, n.ahead=6)  
str(x)

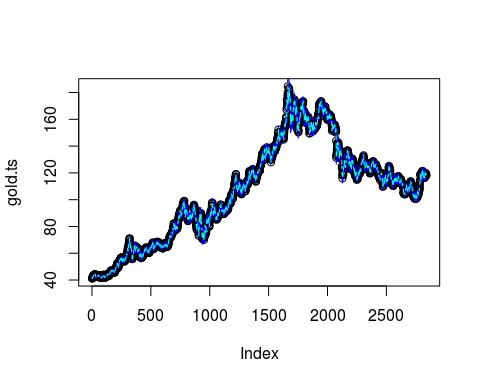
## 'data.frame': 6 obs. of 3 variables:  
## $ meanForecast : num -1.81e-15 -1.81e-15 -1.81e-15 -1.81e-15 -1.81e-15 ...  
## $ meanError : num 0.0105 0.0105 0.0105 0.0106 0.0106 ...  
## $ standardDeviation: num 0.0105 0.0105 0.0105 0.0106 0.0106 ...

x

## meanForecast meanError standardDeviation  
## 1 -1.807752e-15 0.01048150 0.01048150  
## 2 -1.807752e-15 0.01051277 0.01051277  
## 3 -1.807752e-15 0.01054360 0.01054360  
## 4 -1.807752e-15 0.01057397 0.01057397  
## 5 -1.807752e-15 0.01060391 0.01060391  
## 6 -1.807752e-15 0.01063343 0.01063343

argaforecast <- x$meanForecast  
argaforecastplusu <- argaforecast + 1.96\*(x$meanError)  
argaforecastminusu <- argaforecast - 1.96\*(x$meanError)

# voltar aos valores iniciais  
# intercet no modelo fitlin fitlin$coef  
  
argaI<- arga + fitlin$coef  
argaplusuI<-argaplusu + fitlin$coef  
argaminusuI<-argaminusu + fitlin$coef  
  
argaforecastI <- argaforecast + fitlin$coef  
argaforecastplusuI<- argaforecastplusu + fitlin$coef  
argaforecastminusuI<- argaforecastminusu + fitlin$coef  
  
gold.ts.1 <- head(gold.ts,-1)  
gold.ts.arga<- (exp(argaI))\*gold.ts.1  
gold.ts.argaplusu<- (exp(argaplusuI))\*gold.ts.1  
gold.ts.argaminusu<- (exp(argaminusuI))\*gold.ts.1  
  
  
gold.ts.argaforecast1<- (exp(argaforecastI[1]))\*gold.ts[length(gold.ts)]  
gold.ts.argaforecast2<- (exp(argaforecastI[2]))\*gold.ts.argaforecast1  
gold.ts.argaforecast3<- (exp(argaforecastI[3]))\*gold.ts.argaforecast2  
gold.ts.argaforecast4<- (exp(argaforecastI[4]))\*gold.ts.argaforecast3  
gold.ts.argaforecast5<- (exp(argaforecastI[5]))\*gold.ts.argaforecast4  
gold.ts.argaforecast6<- (exp(argaforecastI[6]))\*gold.ts.argaforecast5  
gold.ts.argaforecast <-c(gold.ts.argaforecast1,gold.ts.argaforecast2,gold.ts.argaforecast3,gold.ts.argaforecast4,gold.ts.argaforecast5,gold.ts.argaforecast6)  
  
  
  
gold.ts.argaforecastplusu1<- (exp(argaforecastplusuI[1]))\*gold.ts[length(gold.ts)]  
gold.ts.argaforecastplusu2<- (exp(argaforecastplusuI[2]))\*gold.ts.argaforecastplusu1  
gold.ts.argaforecastplusu3<- (exp(argaforecastplusuI[3]))\*gold.ts.argaforecastplusu2  
gold.ts.argaforecastplusu4<- (exp(argaforecastplusuI[4]))\*gold.ts.argaforecastplusu3  
gold.ts.argaforecastplusu5<- (exp(argaforecastplusuI[5]))\*gold.ts.argaforecastplusu4  
gold.ts.argaforecastplusu6<- (exp(argaforecastplusuI[6]))\*gold.ts.argaforecastplusu5  
gold.ts.argaforecastplusu <- c(gold.ts.argaforecastplusu1, gold.ts.argaforecastplusu2, gold.ts.argaforecastplusu3, gold.ts.argaforecastplusu4, gold.ts.argaforecastplusu5, gold.ts.argaforecastplusu6)  
  
  
  
gold.ts.argaforecastminusu1<- (exp(argaforecastminusuI[1]))\*gold.ts[length(gold.ts)]  
gold.ts.argaforecastminusu2<- (exp(argaforecastminusuI[2]))\*gold.ts.argaforecastminusu1  
gold.ts.argaforecastminusu3<- (exp(argaforecastminusuI[3]))\*gold.ts.argaforecastminusu2  
gold.ts.argaforecastminusu4<- (exp(argaforecastminusuI[4]))\*gold.ts.argaforecastminusu3  
gold.ts.argaforecastminusu5<- (exp(argaforecastminusuI[5]))\*gold.ts.argaforecastminusu4  
gold.ts.argaforecastminusu6<- (exp(argaforecastminusuI[6]))\*gold.ts.argaforecastminusu5  
gold.ts.argaforecastminusu <- c(gold.ts.argaforecastminusu1, gold.ts.argaforecastminusu2, gold.ts.argaforecastminusu3, gold.ts.argaforecastminusu4, gold.ts.argaforecastminusu5, gold.ts.argaforecastminusu6)  
  
  
plot(gold.ts)  
lines(gold.ts.arga, lty= 2,col = 3)  
lines(gold.ts.argaplusu, lty= 2,col = 4)  
lines(gold.ts.argaminusu, lty= 2,col = 4)  
lines(c(gold.ts, gold.ts.argaforecast), lty= 2, col = 5)



# Exponential Smoothing Forecast

Em baixo tentámos modelar a nossa séries e fazer previsões recorrendo ao alisamento exponencial. Logo à partida, decidimos não utilizar o método de alisamento exponencial simples, uma vez que a nossa série é caracterizada por ter tendência e não sazonalidade.

Os métodos de alisamento mais indicados serão portanto o Holt's Linear Trend, Holt's Exponential, Damped Linear Trend e, Damped Exponential Trend.

fit1<-holt(gold.ts, h=6)  
fit2<-holt(gold.ts, initial='optimal', exponential=TRUE, h=6)  
fit3<-holt(gold.ts, initial='optimal', damped=TRUE, h=6)  
fit4<-holt(gold.ts, initial='optimal', exponential=TRUE, damped=TRUE, h=6)

#AIC Results  
AIC(fit1$model)

## [1] 24299.55

AIC(fit2$model)

## [1] 23814.21

AIC(fit3$model)

## [1] 24302.38

AIC(fit4$model)

## [1] 23817.22

#GRAPHICS (seria melhor se focasse o periodo de 2015,5 a 2016,5 - n consegui)  
plot(gold.ts, ylab="Gold Prices (corrigir titulo)", xlab="Month",   
 fcol="black", plot.conf=FALSE)

## Warning in plot.window(...): "fcol" is not a graphical parameter

## Warning in plot.window(...): "plot.conf" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "fcol" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "plot.conf" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "fcol" is not  
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "plot.conf" is  
## not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "fcol" is not  
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "plot.conf" is  
## not a graphical parameter

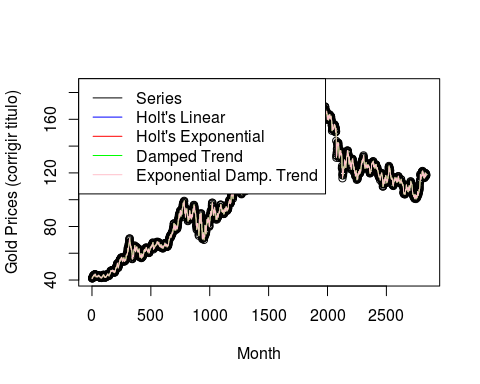
## Warning in box(...): "fcol" is not a graphical parameter

## Warning in box(...): "plot.conf" is not a graphical parameter

## Warning in title(...): "fcol" is not a graphical parameter

## Warning in title(...): "plot.conf" is not a graphical parameter

lines(fitted(fit1), col="blue")  
lines(fitted(fit2), col="red")  
lines(fitted(fit3), col="green")  
lines(fitted(fit4), col="pink")  
lines(fit1$mean, col="blue")  
lines(fit2$mean, col="red")  
lines(fit3$mean, col="green")  
lines(fit4$mean, col="pink")  
legend("topleft", lty=1, col=c("black","blue","red","green","pink"),  
 c("Series", "Holt's Linear", "Holt's Exponential","Damped Trend","Exponential Damp. Trend"))



#Forecasts (para calcular Erro de previsão dos modelos)  
fit1$mean

## Time Series:  
## Start = 2840   
## End = 2845   
## Frequency = 1   
## [1] 118.7192 118.7463 118.7734 118.8005 118.8276 118.8547

fit2$mean

## Time Series:  
## Start = 2840   
## End = 2845   
## Frequency = 1   
## [1] 118.7050 118.7188 118.7327 118.7465 118.7604 118.7742

fit3$mean

## Time Series:  
## Start = 2840   
## End = 2845   
## Frequency = 1   
## [1] 118.6908 118.6909 118.6909 118.6909 118.6909 118.6909

fit4$mean

## Time Series:  
## Start = 2840   
## End = 2845   
## Frequency = 1   
## [1] 118.6904 118.6905 118.6906 118.6906 118.6907 118.6907

Segundo o AIC o melhor modelo para a nossa série é o Holt's Exponential. Este modelo é mais adequado para séries com tendência e sem sazonalidade como é o caso da nossa. Resta comprovar se o erro de previsão do modelo Holt's Exponential é menor do que o do modelo Damped Exponential Trend cujo AIC foi o segundo mais baixo.