Progetto - Deep Learning on Temporal Data

Confronto ARIMA model e ARMA(1,1)-sGARCH(1,1)

Davide Zicca

9/7/2021

Motivazione e data setting

L'analisi che viene qui proposta prende spunto dalle lezioni *Deep Learning on Temporal Data* e presenta un confronto tra i modelli ARIMA e un modello ARMA(1,1)-sGARCH(1,1) che presenta dunque anche l'equazione per la varianza. I modelli GARCH sono largamente usati in letteratura per catturare gli effetti *disruptive* a cui sono spesso soggetti i ticker azionari. Ho scelto i prezzi azionari di Amazon da Giugno 2015 a Giugno 2021.

```
#librerie necessarie
library(ggplot2)
library(tidyquant)
library(timetk)
library(tseries)
library(timeSeries)
library(forecast)
library(seastests)
library(rugarch)
library(fDMA)
library(dplyr)
set.seed(29)
amazon = tq_get("AMZN",
                  from = '2015-06-01',
                  to = "2021-06-01",
                  get = "stock.prices")
```

Data Visualization, calcolo log returns e fit del MACD

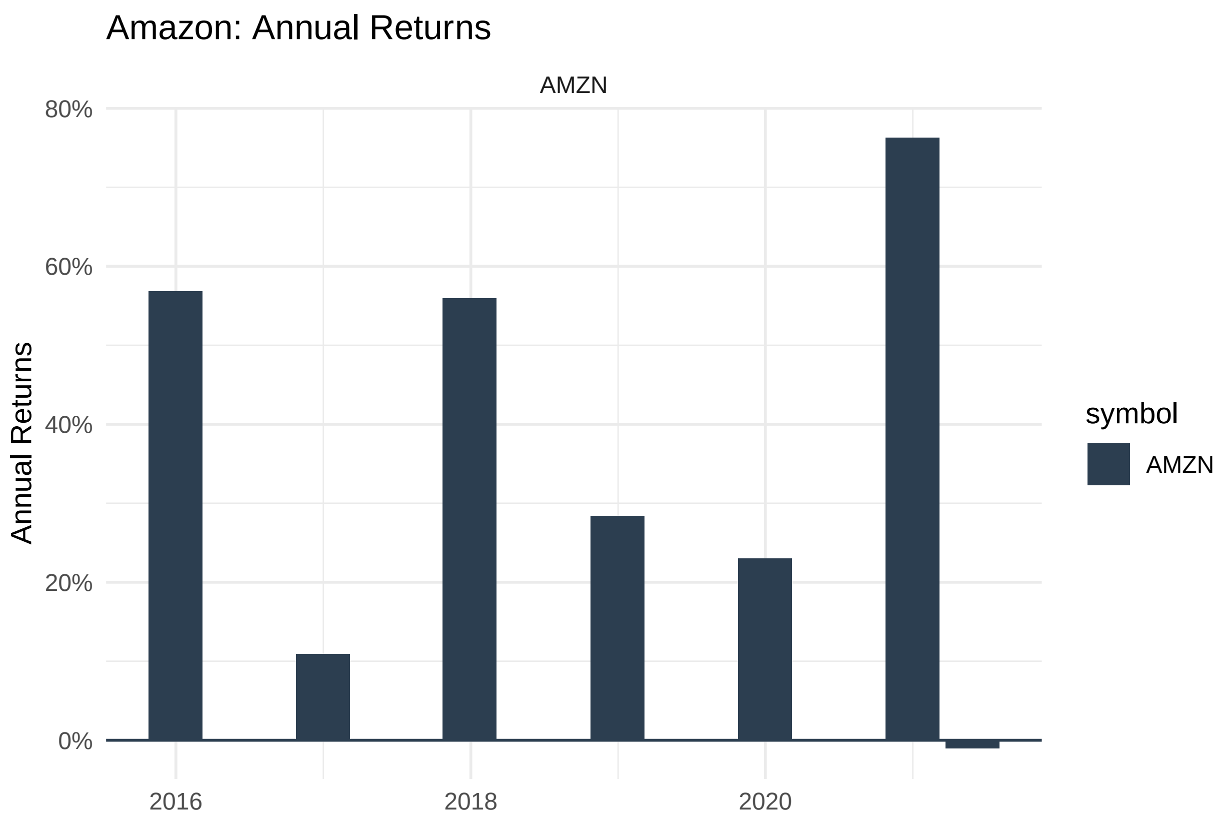
```
amazon %>%
  ggplot(aes(x = date, y = adjusted)) +
  geom_line() +
  ggtitle("Amazon") +
  labs(x = "Date", "Price") +
  scale_x_date(date_breaks = "years", date_labels = "%Y") +
  labs(x = "Data", y = "Adjusted Price") +
  theme_minimal()
```


```
AMZN_annual_returns = amazon %>%
  group_by(symbol) %>%
  tq_transmute(select     = adjusted,
               mutate_fun = periodReturn,
               period     = "yearly",
               type       = "arithmetic")
AMZN_annual_returns
```

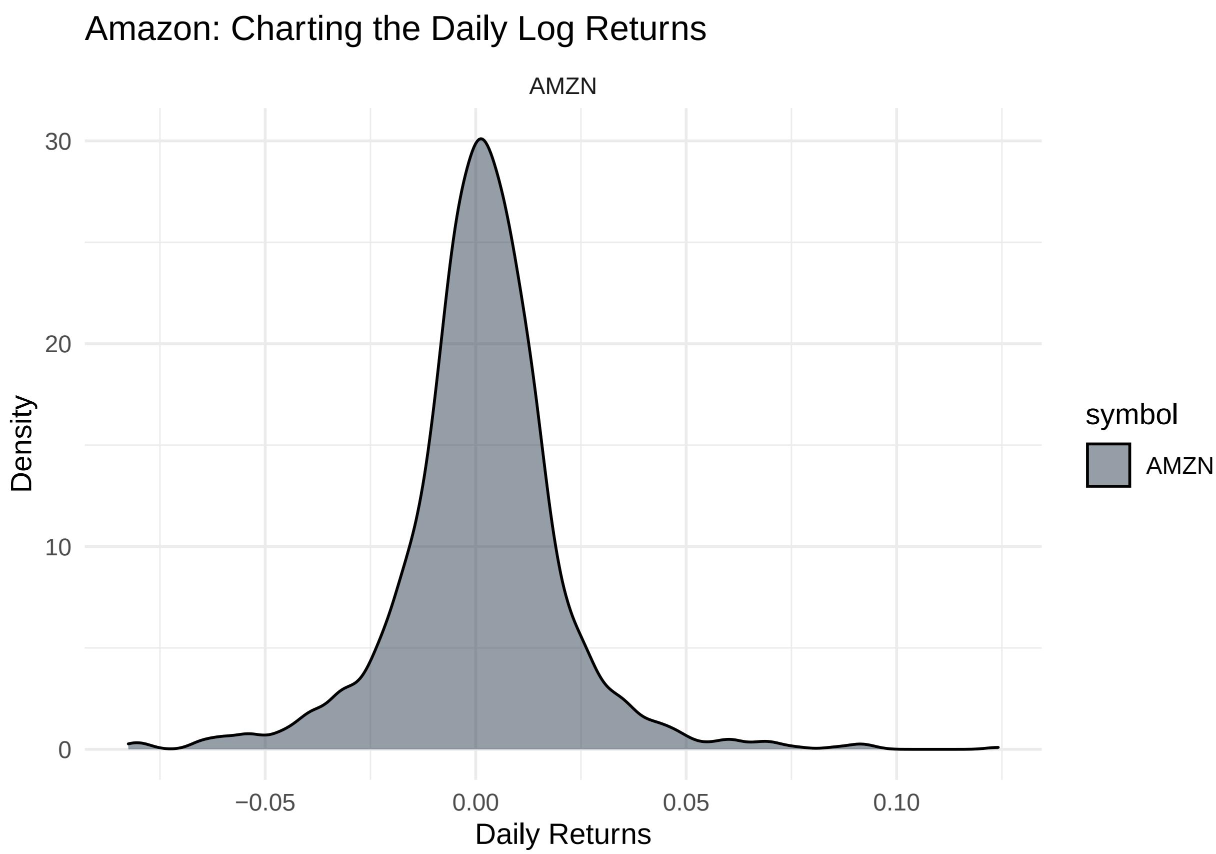
```
## # A tibble: 7 x 3
## # Groups:   symbol [1]
##   symbol date       yearly.returns
##   <chr>  <date>              <dbl>
## 1 AMZN   2015-12-31         0.568
## 2 AMZN   2016-12-30         0.109
## 3 AMZN   2017-12-29         0.560
## 4 AMZN   2018-12-31         0.284
## 5 AMZN   2019-12-31         0.230
## 6 AMZN   2020-12-31         0.763
## 7 AMZN   2021-05-28        -0.0104
```

```
AMZN_annual_returns %>%
  ggplot(aes(x = date, y = yearly.returns, fill = symbol)) +
  geom_col() +
  geom_hline(yintercept = 0, color = palette_light()[[1]]) +
  scale_y_continuous(labels = scales::percent) +
  labs(title = "Amazon: Annual Returns",
       y = "Annual Returns", x = "") +
```

```r
  facet_wrap(~ symbol, ncol = 2, scales = "free_y") +
  theme_minimal() +
  scale_fill_tq()
```




```r
AMZN_daily_log_returns = amazon %>%
  group_by(symbol) %>%
  tq_transmute(select     = adjusted,
               mutate_fun = periodReturn,
               period     = "daily",
               type       = "log",
               col_rename = "daily.returns")

AMZN_daily_log_returns %>%
  ggplot(aes(x = daily.returns, fill = symbol)) +
  geom_density(alpha = 0.5) +
  labs(title = "Amazon: Charting the Daily Log Returns",
       x = "Daily Returns", y = "Density") +
  theme_minimal() +
  scale_fill_tq() +
  facet_wrap(~ symbol, ncol = 2)
```



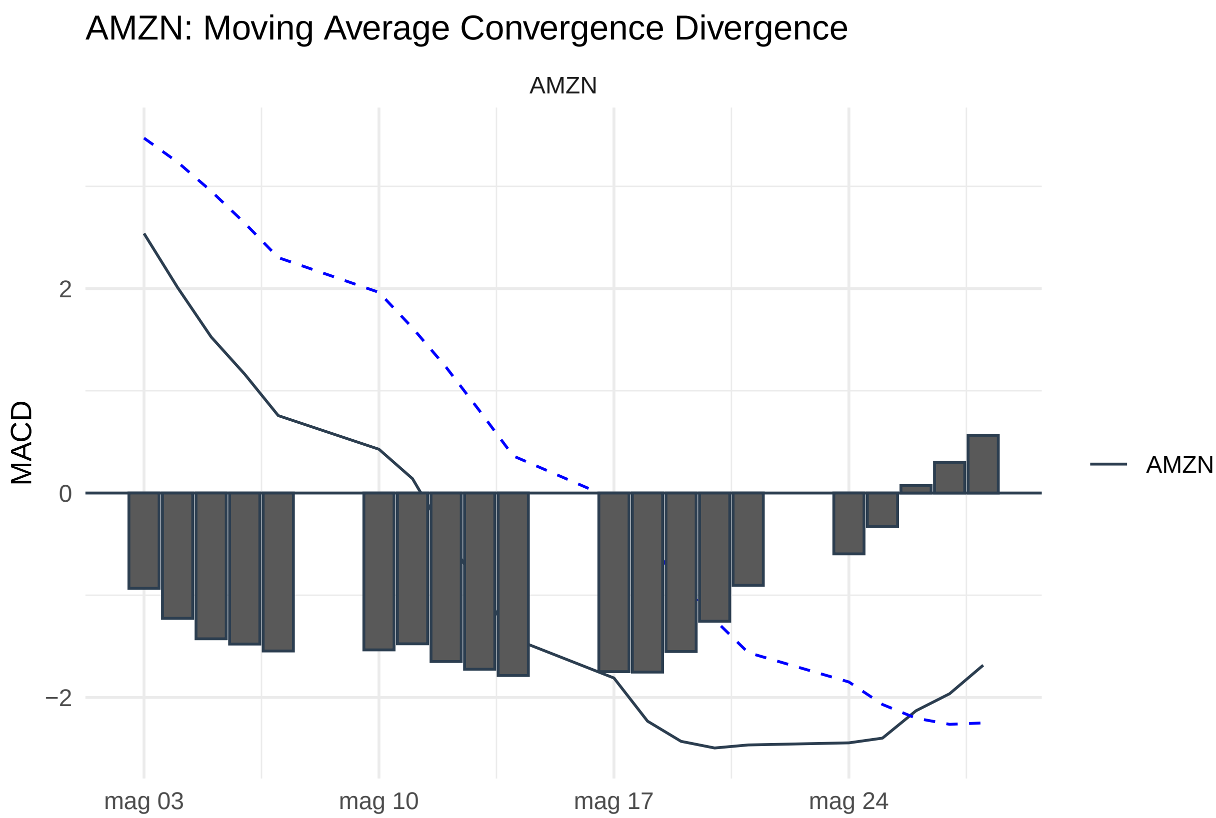
Calcolo e plot del Moving Average Convergence Divergence:

```
AMZN_macd = amazon %>%
  group_by(symbol) %>%
  tq_mutate(select     = close,
            mutate_fun = MACD,
            nFast      = 12,
            nSlow      = 26,
            nSig       = 9,
            maType     = SMA) %>%
  mutate(diff = macd - signal) %>%
  select(-(open:volume))
AMZN_macd
```

```
## # A tibble: 1,511 x 6
## # Groups:   symbol [1]
##    symbol date       adjusted  macd signal  diff
##    <chr>  <date>        <dbl> <dbl>  <dbl> <dbl>
##  1 AMZN   2015-06-01     431.    NA     NA    NA
##  2 AMZN   2015-06-02     431.    NA     NA    NA
##  3 AMZN   2015-06-03     437.    NA     NA    NA
##  4 AMZN   2015-06-04     431.    NA     NA    NA
##  5 AMZN   2015-06-05     427.    NA     NA    NA
##  6 AMZN   2015-06-08     424.    NA     NA    NA
##  7 AMZN   2015-06-09     425.    NA     NA    NA
##  8 AMZN   2015-06-10     431.    NA     NA    NA
##  9 AMZN   2015-06-11     433.    NA     NA    NA
```

```
## 10 AMZN   2015-06-12     430.     NA     NA    NA
## # ... with 1,501 more rows
```

```
AMZN_macd %>%
  filter(date >= as_date("2021-05-01")) %>%
  ggplot(aes(x = date)) +
  geom_hline(yintercept = 0, color = palette_light()[[1]]) +
  geom_line(aes(y = macd, col = symbol)) +
  geom_line(aes(y = signal), color = "blue", linetype = 2) +
  geom_bar(aes(y = diff), stat = "identity", color = palette_light()[[1]]) +
  facet_wrap(~ symbol, ncol = 2, scale = "free_y") +
  labs(title = "AMZN: Moving Average Convergence Divergence",
       y = "MACD", x = "", color = "") +
  theme_minimal() +
  scale_color_tq()
```



Test: stazionarietà e presenza ARCH effect

```
#Stationarity test
#adf.test(AMZN_daily_log_returns$daily.returns, alternative = "stationary")
# p-value = 0.01 -> si rifiuta l'ipotesi nulla
ts= ts(AMZN_daily_log_returns)[,3]

# seasonality test
isSeasonal(ts, freq=1)
```

```
## [1] FALSE
```

```r
#stationary test
adf.test(ts, alternative= "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts
## Dickey-Fuller = -11.801, Lag order = 11, p-value = 0.01
## alternative hypothesis: stationary
```

```r
#ARCH EFFECT

archtest(as.vector(AMZN_daily_log_returns$daily.returns))
```

```
##
##  Engle's LM ARCH Test
##
## data:  as.vector(AMZN_daily_log_returns$daily.returns)
## statistic = 53.451, lag = 1, p-value = 2.652e-13
## alternative hypothesis: ARCH effects of order 1 are present
```

```r
# presenza ARCH effect
```

Fit modelli ARMA, verifica auto-correlation e valori BIC per scelta modello

```r
n = length(ts)
n
```

```
## [1] 1511
```

```r
nV=round(n/3)  # Validation set (33% del totale)
nV
```

```
## [1] 504
```

```r
nT=n-nV  # training set - osservazioni
train=ts[c(1:nT)]
valid=ts[c((nT+1):n)]

auto_model1=auto.arima(train, ic="aic", stationary=FALSE,seasonal=FALSE)
auto_model1
```

```
## Series: train
## ARIMA(0,0,0) with non-zero mean
##
## Coefficients:
##         mean
##       0.0014
## s.e.  0.0006
##
## sigma^2 estimated as 0.0003423:  log likelihood=2589.54
## AIC=-5175.08   AICc=-5175.07   BIC=-5165.25
```

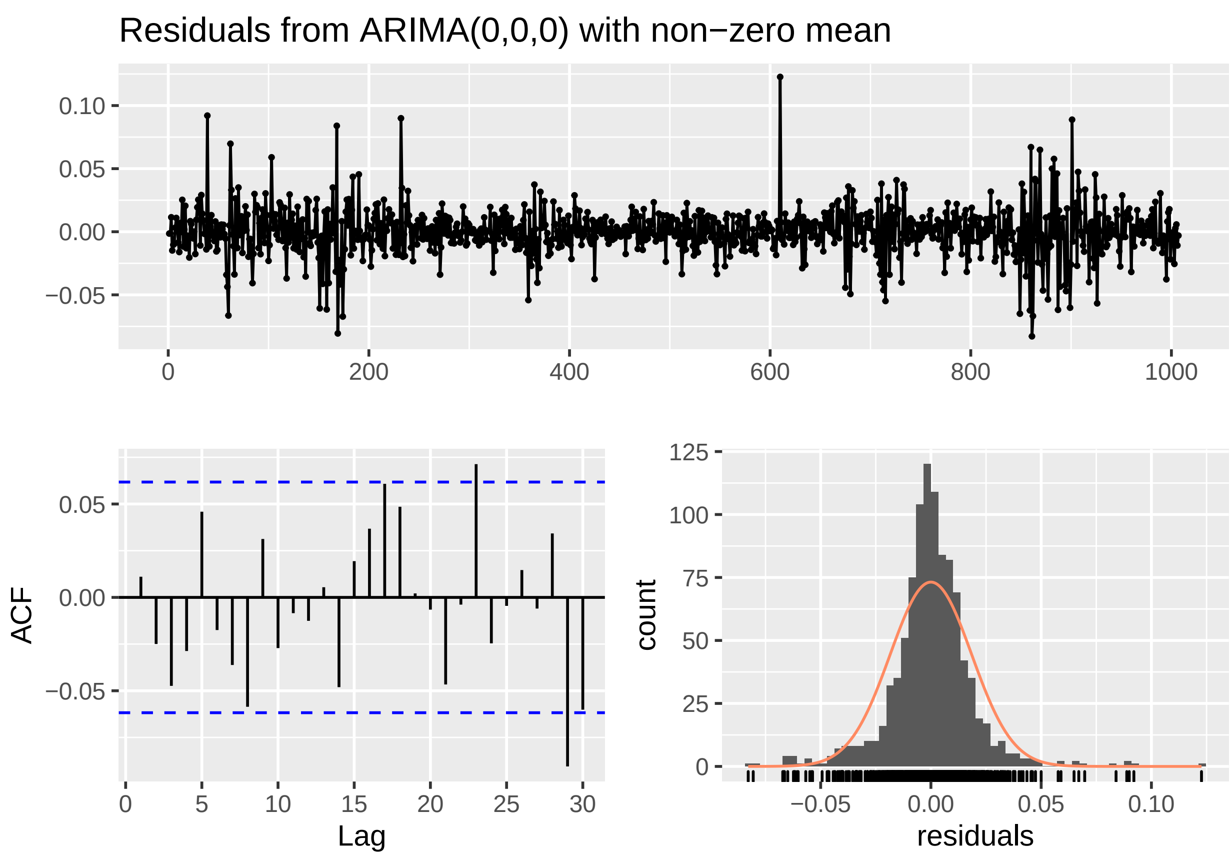
```r
res1=auto_model1$residuals

Box.test(res1, lag = 12, type = "Box-Pierce", fitdf=0)
```

```
##
##  Box-Pierce test
##
## data:  res1
## X-squared = 12.998, df = 12, p-value = 0.3692
```

```r
Box.test(res1, lag = 12, type = "Ljung-Box",  fitdf=0)
```

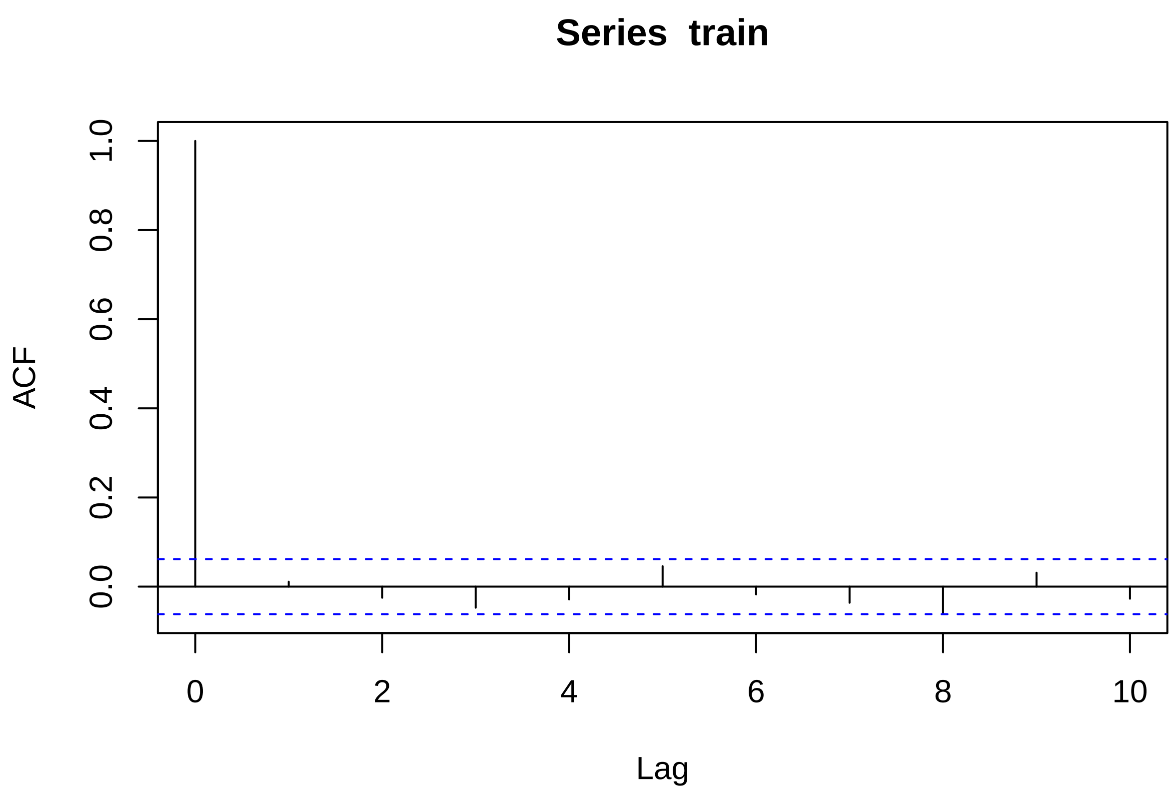
```
##
##  Box-Ljung test
##
## data:  res1
## X-squared = 13.104, df = 12, p-value = 0.3616
```

```r
checkresiduals(auto_model1, include.mean=FALSE, plot=TRUE)
```

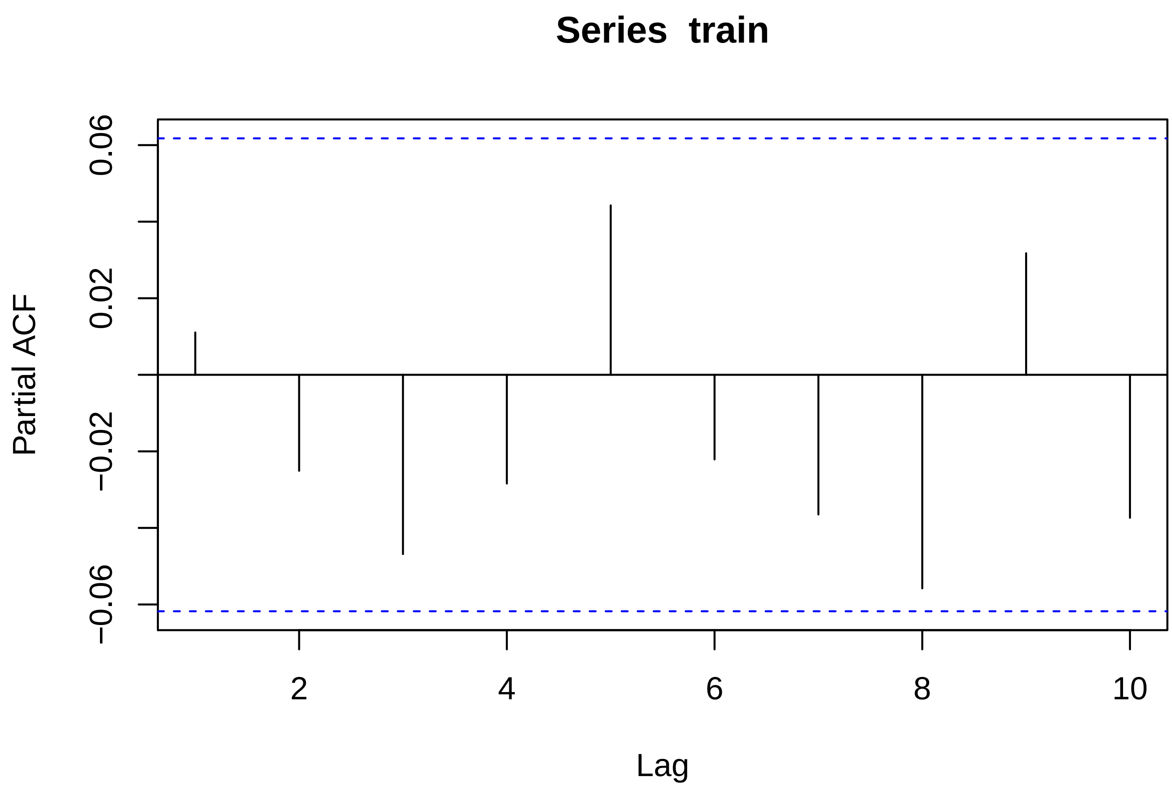



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,0) with non-zero mean
## Q* = 12.87, df = 9, p-value = 0.1686
##
## Model df: 1.   Total lags used: 10
```

```r
acf(train, lag=10)
```




```r
pacf(train, lag=10)
```




```
arima_model=arima(train, order=c(1,0,1) )
arima_model
```

```
##
## Call:
## arima(x = train, order = c(1, 0, 1))
##
## Coefficients:
##          ar1     ma1  intercept
##       0.0051  0.0062     0.0014
## s.e.  1.2736  1.3051     0.0006
##
## sigma^2 estimated as 0.0003419:  log likelihood = 2589.6,  aic = -5171.2
```

```
BIC(auto_model1)
```

```
## [1] -5165.248
```

```
BIC(arima_model)
```

```
## [1] -5151.545
```

Modello ARMA(1,1)-sGARCH(1,1): fit, risultati e plot

```
# GARCH MODEL assumendo una distribuzione Normale
#sGARCH
library(rugarch)
```

```r
s_garchMod = ugarchspec(mean.model = list(armaOrder = c(1, 1), include.mean = TRUE
),
variance.model = list(model = 'sGARCH',
                      garchOrder = c(1, 1)),
distribution.model = "norm")

s_garchFit = ugarchfit(spec=s_garchMod, data=ts)
s_garchFit
```

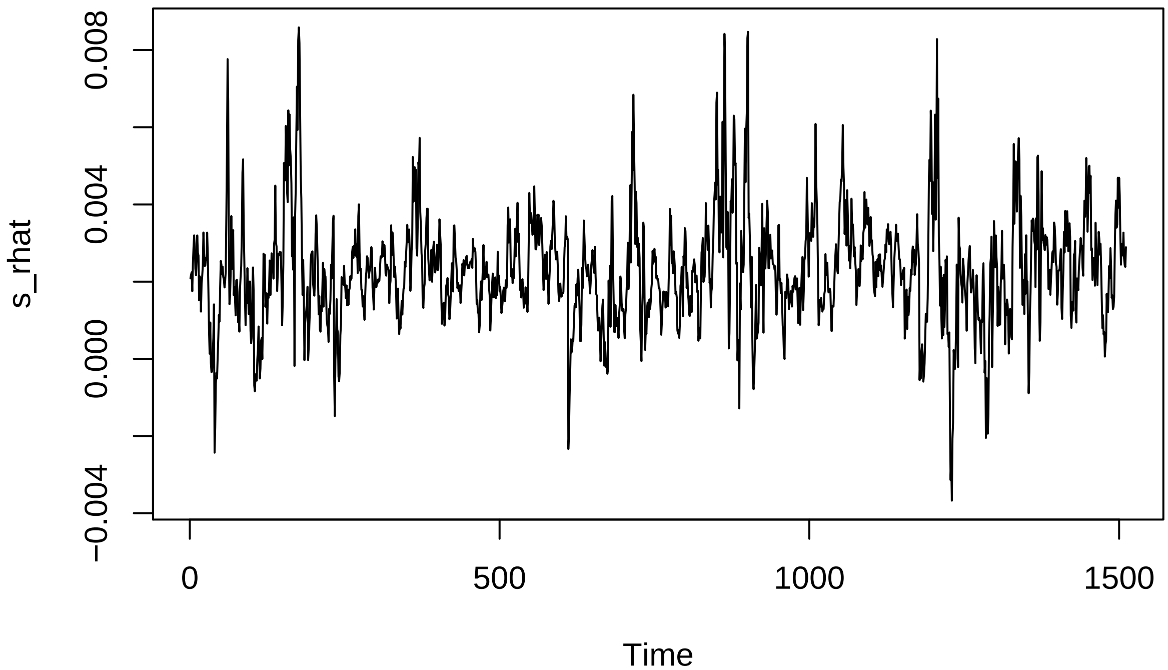
```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(1,0,1)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.002084    0.000315   6.6101        0
## ar1     0.797653    0.114210   6.9841        0
## ma1    -0.840888    0.101013  -8.3246        0
## omega   0.000027    0.000005   5.3200        0
## alpha1  0.181082    0.028111   6.4417        0
## beta1   0.754320    0.030798  24.4923        0
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.002084    0.000405   5.1413  0.00000
## ar1     0.797653    0.127935   6.2348  0.00000
## ma1    -0.840888    0.111760  -7.5241  0.00000
## omega   0.000027    0.000011   2.5527  0.01069
## alpha1  0.181082    0.035858   5.0500  0.00000
## beta1   0.754320    0.043618  17.2939  0.00000
##
## LogLikelihood : 3999.049
##
## Information Criteria
## ------------------------------------
##
## Akaike       -5.2853
## Bayes        -5.2642
## Shibata      -5.2853
## Hannan-Quinn -5.2774
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic  p-value
## Lag[1]                      3.267 0.070701
```

```
## Lag[2*(p+q)+(p+q)-1][5]     4.934 0.004145
## Lag[4*(p+q)+(p+q)-1][9]     5.617 0.324344
## d.o.f=2
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.3363  0.5620
## Lag[2*(p+q)+(p+q)-1][5]    0.6483  0.9326
## Lag[4*(p+q)+(p+q)-1][9]    1.0435  0.9844
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]   0.06644 0.500 2.000  0.7966
## ARCH Lag[5]   0.30912 1.440 1.667  0.9378
## ARCH Lag[7]   0.44368 2.315 1.543  0.9832
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  0.6034
## Individual Statistics:
## mu     0.14755
## ar1    0.03782
## ma1    0.04302
## omega  0.05536
## alpha1 0.05712
## beta1  0.06437
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.49 1.68 2.12
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                     t-value    prob sig
## Sign Bias          1.992939 0.04645  **
## Negative Sign Bias 0.594396 0.55234
## Positive Sign Bias 0.000715 0.99943
## Joint Effect       5.879244 0.11764
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20      90.5    2.709e-11
## 2    30     104.9    1.567e-10
## 3    40     109.5    1.313e-08
## 4    50     121.3    4.648e-08
##
##
## Elapsed time : 0.433579
```

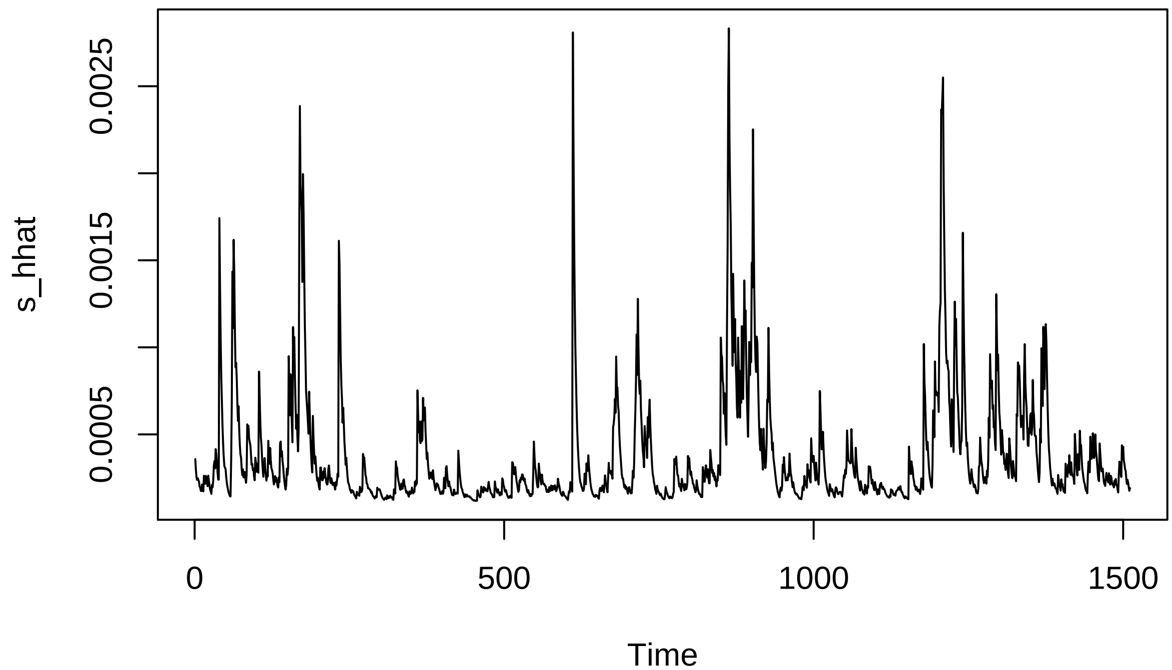
```r
## Risultati del modello
coef(s_garchFit)
```

```
##            mu           ar1           ma1         omega        alpha1
##  2.083982e-03  7.976531e-01 -8.408882e-01  2.717585e-05  1.810823e-01
##         beta1
##  7.543195e-01
```

```r
s_rhat = s_garchFit@fit$fitted.values
plot.ts(s_rhat)
```




```r
s_hhat = ts(s_garchFit@fit$sigma^2)
plot.ts(s_hhat)
```




```
fit.val     = coef(s_garchFit)
fit.sd      = diag(vcov(s_garchFit))
true.val = s_garchFit@fit$tval

fit.conf.lb = fit.val + qnorm(0.025) * fit.sd
fit.conf.ub = fit.val + qnorm(0.975) * fit.sd
print(fit.val)
```

```
##            mu           ar1           ma1         omega        alpha1
##  2.083982e-03  7.976531e-01 -8.408882e-01  2.717585e-05  1.810823e-01
##         beta1
##  7.543195e-01
```

```
print(fit.sd)
```

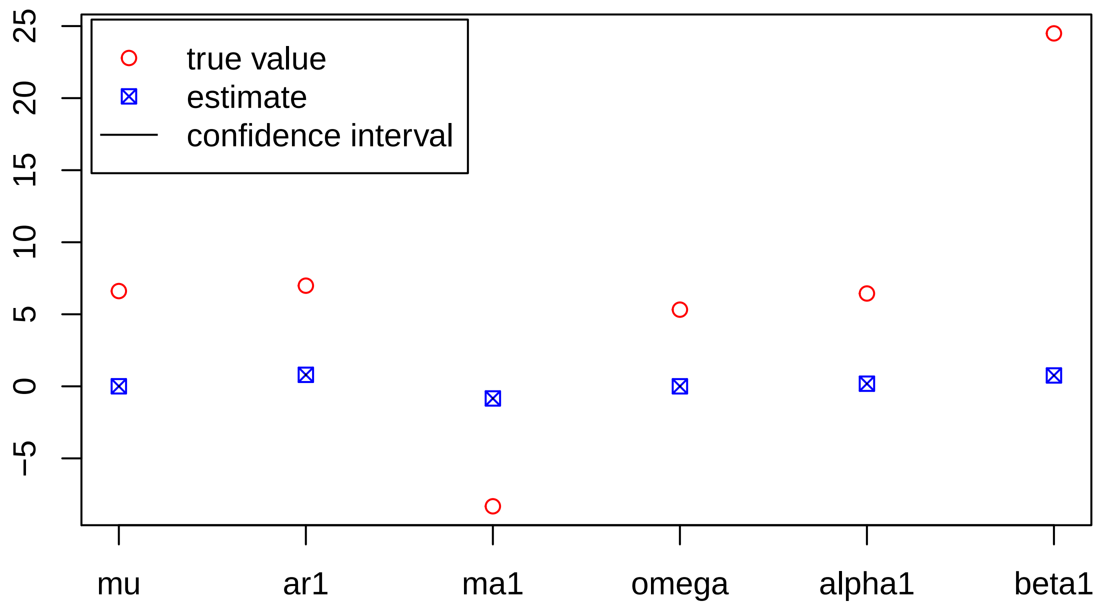
```
## [1] 9.939810e-08 1.304401e-02 1.020362e-02 2.609369e-11 7.902315e-04
## [6] 9.485306e-04
```

```
print(true.val)
```

```
##        mu       ar1       ma1     omega    alpha1     beta1
##  6.610052  6.984067 -8.324558  5.320047  6.441676 24.492306
```

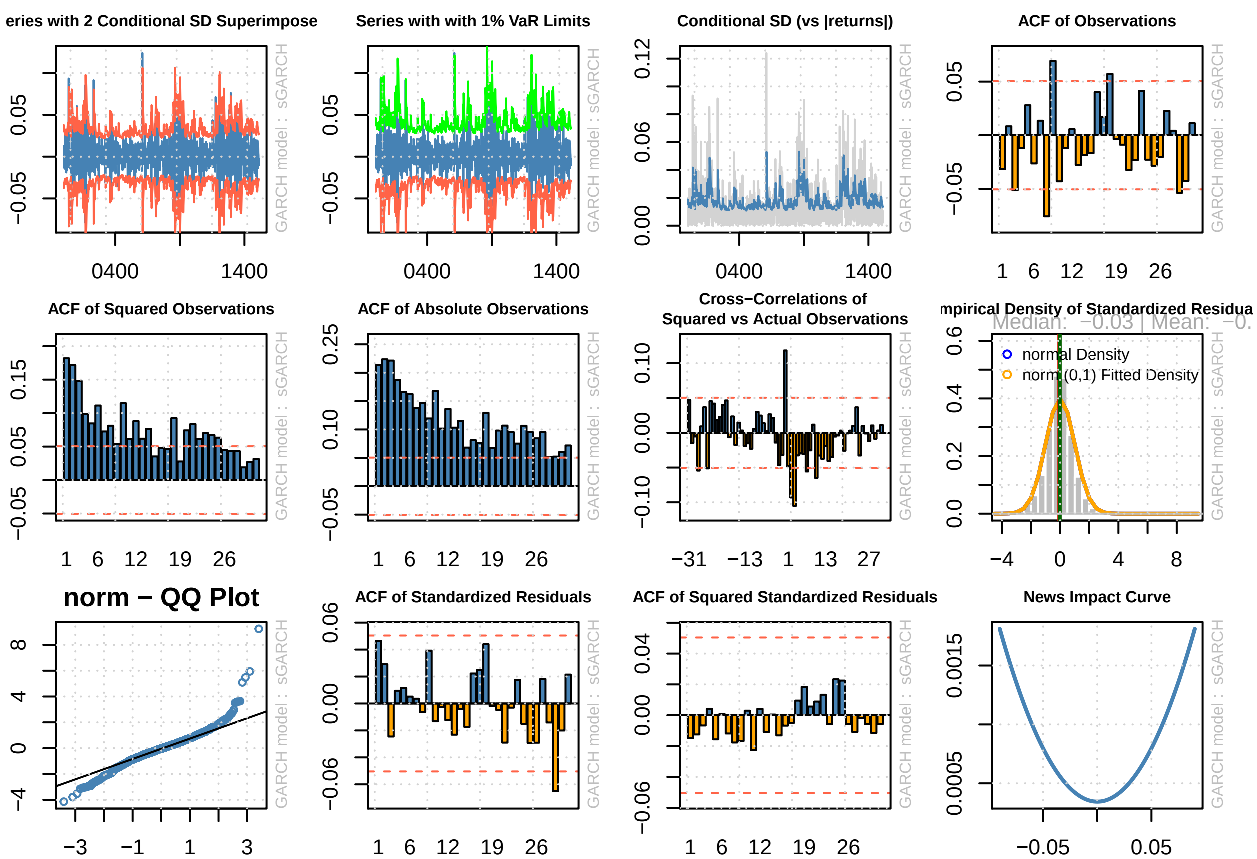
```
plot(true.val, pch = 1, col = "red",
     ylim = range(c(fit.conf.lb, fit.conf.ub, true.val)),
     xlab = "", ylab = "", axes = FALSE)
box(); axis(1, at = 1:length(fit.val), labels = names(fit.val)); axis(2)
points(coef(s_garchFit), col = "blue", pch = 7)
```

```r
for (i in 1:length(fit.val)) {
  lines(c(i,i), c(fit.conf.lb[i], fit.conf.ub[i]))
}
legend( "topleft", legend = c("true value", "estimate", "confidence interval"),
        col = c("red", "blue", 1), pch = c(1, 7, NA), lty = c(NA, NA, 1), inset = 0.01)
```




```r
par(mfrow=c(2, 3))
par(mar = c(2, 2, 2, 2))
plot(s_garchFit,which="all")
```

```
##
## please wait...calculating quantiles...
```



Confronto tra modelli

```r
# confronto tra modelli
infocriteria(s_garchFit)[2] #BIC= -5.264179
```

```
## [1] -5.264179
```

```r
BIC(auto_model1) # = -5165.248 -> è il valore più basso e quindi quello da
```

```
## [1] -5165.248
```

```r
                  # scegliere. auto_model1 è il modello da scegliere
BIC(arima_model) # = -5151.545
```

```
## [1] -5151.545
```

Conclusione

Dopo una attenta analisi dei valori BIC, viene scelto il modello \textbf{auto_model1} poichè mostra il valore più basso di BIC. Tuttavia, il modello ARMA(1,1)-sGARCH(1,1), grazie alla presenza dell'equazione della varianza è in grado di fornire informazioni maggiori rispetto ai modelli ARMA che presentano la sola equazione della media.