

# Crypto News!

Enrico Pirani Max

## Importazione delle librerie necessarie

In primo luogo, per eseguire il nostro processo di creazione di strategie di trading, dobbiamo importare le librerie necessarie nel nostro ambiente. In tutto questo processo, utilizzeremo alcune delle librerie finanziarie più popolari in R, ovvero Quantmod, TTR e Performance Analytics. Utilizzando la funzione library in R, possiamo importare i nostri pacchetti richiesti.

```
y <- rnorm(100)
x <- rnorm(100)
m <- lm(y ~ x)
summary(x)

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.7067 -0.3848  0.2961  0.2732  0.9606  3.2715

summary(m)

##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.1737 -0.5614  0.0973  0.7656  2.1140
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.230017   0.109606  -2.099   0.0384 *
## x           -0.006153   0.100508  -0.061   0.9513
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.061 on 98 degrees of freedom
## Multiple R-squared:  3.825e-05, Adjusted R-squared:  -0.01017
## F-statistic: 0.003748 on 1 and 98 DF,  p-value: 0.9513

library(quantmod)

## Loading required package: xts

## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##      as.Date, as.Date.numeric

## Loading required package: TTR

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

library(PerformanceAnalytics)

##
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':
##
##      legend

library(TTR)
```

### *Passaggio 2: Estrazione dei dati da Yahoo e Plotting di base*

durante tutto il nostro processo, lavoreremo con i dati del prezzo delle cryptovalute Bitcoin, Ethereum, Binance, Cardano e XRP. Estraiamo i dati di queste valute da Yahoo in R.

```
getSymbols("BTC-USD", src = "yahoo", from = "2019-01-01")

## [1] "BTC-USD"

getSymbols("ETH-USD", src = "yahoo", from = "2019-01-01")

## [1] "ETH-USD"

getSymbols("BNB-USD", src = "yahoo", from = "2019-01-01")

## [1] "BNB-USD"

getSymbols("ADA-USD", src = "yahoo", from = "2019-01-01")

## [1] "ADA-USD"

getSymbols("XRP-USD", src = "yahoo", from = "2019-01-01")

## [1] "XRP-USD"
```

```
getSymbols("SOL-USD", src = "yahoo", from = "2020-01-01")
```

```
## [1] "SOL-USD"
```

Ora facciamo un po' di visualizzazione dei nostri dati estratti! Il seguente codice produce un grafico a barre finanziario dei prezzi delle azioni insieme al volume.

```
barChart(`BTC-USD`, theme = chartTheme("black"))
```

```
barChart(`BNB-USD`, theme = chartTheme("black"))
```

```
barChart(`ETH-USD`, theme = chartTheme("black"))
```

```
barChart(`ADA-USD`, theme = chartTheme("black"))
```

```
barChart(`XRP-USD`, theme = chartTheme("black"))
```

```
barChart(`SOL-USD`, theme = chartTheme("black"))
```

### *Creazione di indicatori tecnici*

Ci sono molti indicatori tecnici utilizzati per l'analisi finanziaria ma, per la nostra analisi, utilizzeremo e creeremo sei dei più famosi indicatori tecnici, ovvero: Media mobile semplice (SMA), Parabolic Stop And Reverse (SAR), Indice del canale delle materie prime (CCI), Tasso di variazione (ROC), Indice del momento stocastico (SMI) e infine Williams %R. Facciamolo!. Nella nuova dimensione di questi

Media mobile semplice (SMA):

L'intervallo di tempo standard che prenderemo è di 20 giorni SMA e 50 giorni SMA. Ma non ci sono restrizioni nell'uso di qualsiasi intervallo di tempo.

Il seguente codice calcolerà la SMA di tre aziende per 20 giorni e 50 giorni insieme ad un grafico:

```
BTC <- `BTC-USD`
```

```
ETH <- `ETH-USD`
```

```
BNB <- `BNB-USD`
```

```
ADA <- `ADA-USD`
```

```
XRP <- `XRP-USD`
```

## # 1. BTC-USD

```
sma50_btc <- SMA(BTC$`BTC-USD.Close`, n = 50)
sma100_btc <- SMA(BTC$`BTC-USD.Close`, n = 100)
lineChart(`BTC-USD`, theme = chartTheme("black"))
```

```
addSMA(n = 50, col = "blue")
```

```
addSMA(n = 100, col = "orange")
legend("left",
  col = c("green", "blue", "orange"),
  legend = c("BTC-USD", "SMA50", "SMA100"), lty = 1, bty = "n",
  text.col = "white", cex = 0.8
)
```

## # 2. ETH-USD

```
sma50_btc <- SMA(ETH$`ETH-USD.Close`, n = 50)
sma100_btc <- SMA(ETH$`ETH-USD.Close`, n = 100)
lineChart(`ETH-USD`, theme = chartTheme("black"))
```

```
addSMA(n = 50, col = "blue")
```

```
addSMA(n = 100, col = "orange")
legend("left",
  col = c("green", "blue", "orange"),
  legend = c("ETH-USD", "SMA50", "SMA100"), lty = 1, bty = "n",
  text.col = "white", cex = 0.8
)
```

```
sma50_btc <- SMA(ADA$`ADA-USD.Close`, n = 50)
sma100_btc <- SMA(ADA$`ADA-USD.Close`, n = 100)
lineChart(`ADA-USD`, theme = chartTheme("black"))
```

```
addSMA(n = 50, col = "blue")
```

```
addSMA(n = 100, col = "orange")
legend("left",
  col = c("green", "blue", "orange"),
  legend = c("ADA-USD", "SMA50", "SMA100"), lty = 1, bty = "n",
  text.col = "white", cex = 0.8
)
```

```
sma50_btc <- SMA(BNB$`BNB-USD.Close`, n = 50)
sma100_btc <- SMA(BNB$`BNB-USD.Close`, n = 100)
lineChart(`BNB-USD`, theme = chartTheme("black"))
```

```
addSMA(n = 50, col = "blue")
```

```
addSMA(n = 100, col = "orange")
legend("left",
  col = c("green", "blue", "orange"),
  legend = c("BNB-USD", "SMA50", "SMA100"), lty = 1, bty = "n",
  text.col = "white", cex = 0.8
)
```

```
sma50_btc <- SMA(XRP$`XRP-USD.Close`, n = 50)
sma100_btc <- SMA(XRP$`XRP-USD.Close`, n = 100)
lineChart(`XRP-USD`, theme = chartTheme("black"))
```

```
addSMA(n = 50, col = "blue")
```

```
addSMA(n = 100, col = "orange")
legend("left",
  col = c("green", "blue", "orange"),
  legend = c("BNB-USD", "SMA50", "SMA100"), lty = 1, bty = "n",
  text.col = "white", cex = 0.8
)
```

Indice del Canale delle Materie Prime (CCI):

Per calcolare il CCI, dobbiamo considerare i prezzi giornalieri di massimo, minimo e chiusura delle aziende insieme a un periodo di

tempo specificato e un valore costante. In questo passaggio, prenderemo 20 giorni come periodo di tempo e 0,015 come valore costante.

Il seguente codice calcolerà il CCI delle aziende insieme a un grafico:

```
# 1.BTC
cci_BTC <- CCI(HLC(BTC), n = 20, c = 0.015)
barChart(BTC, theme = "black")
```

```
addCCI(n = 20, c = 0.015)
```

```
# 2.ETH
cci_eth <- CCI(HLC(ETH), n = 20, c = 0.015)
barChart(ETH, theme = "black")
```

```
addCCI(n = 20, c = 0.015)
```

```
# 3.BNB
cci_BNB <- CCI(HLC(BNB), n = 20, c = 0.015)
barChart(BNB, theme = "black")
```

```
addCCI(n = 20, c = 0.015)
```

```
# 4. XRP
cci_XRP <- CCI(HLC(XRP), n = 20, c = 0.015)
barChart(XRP, theme = "black")
```

```
addCCI(n = 20, c = 0.015)
```

```
# 5.ADA
cci_ADA <- CCI(HLC(ADA), n = 20, c = 0.015)
barChart(ADA, theme = "black")
```

```
addCCI(n = 20, c = 0.015)
```

Tasso di variazione (ROC)

Per calcolare il ROC, dobbiamo considerare un intervallo di tempo specificato e non ci sono restrizioni nell'utilizzare qualsiasi periodo. In questo passaggio, prenderemo 25 giorni come periodo di tempo.

Il codice seguente calcolerà il ROC delle aziende insieme a un grafico:

```
# 1. BTC
```

```
roc_btc <- ROC(BTC$`BTC-USD.Close`, n = 25)
barChart(BTC, theme = "black")
```

```
addROC(n = 25)
legend("left",
  col = "red", legend = "ROC(25)", lty = 1, bty = "n",
  text.col = "white", cex = 0.8
)
```

```
roc_ETH <- ROC(ETH$`ETH-USD.Close`, n = 25)
barChart(ETH, theme = "black")
```

```
addROC(n = 25)
legend("left",
  col = "red", legend = "ROC(25)", lty = 1, bty = "n",
  text.col = "white", cex = 0.8
)
```

```
roc_BNB <- ROC(BNB$`BNB-USD.Close`, n = 25)
barChart(BNB, theme = "black")
```

```
addROC(n = 25)
legend("left",
  col = "red", legend = "ROC(25)", lty = 1, bty = "n",
  text.col = "white", cex = 0.8
)
```

```
roc_XRP <- ROC(XRP$`XRP-USD.Close`, n = 25)
barChart(XRP, theme = "black")
```

```
addROC(n = 25)
legend("left",
      col = "red", legend = "ROC(25)", lty = 1, bty = "n",
      text.col = "white", cex = 0.8
)
```

```
roc_ADA <- ROC(ADA$`ADA-USD.Close`, n = 25)
barChart(ADA, theme = "black")
```

```
addROC(n = 25)
legend("left",
      col = "red", legend = "ROC(25)", lty = 1, bty = "n",
      text.col = "white", cex = 0.8
)
```

```
# Set the number of successes (3 heads)
x <- 3
```

```
# Set the number of trials (5 coin flips)
size <- 5
```

```
# Set the probability of success (0.5 for a fair coin)
prob <- 0.5
```

```
# Calculate the probability using the dbinom function
probability <- dbinom(x, size, prob)
```

```
# Print the result
print(probability)
```

```
## [1] 0.3125
```



*Generate random data following a normal Distribution*

```
data <- rnorm(1000, mean = 0, sd = 1)
# Plot a histogram of the data
hist(data, breaks = 30, col = "skyblue", main = "Normal Distribution", xlab = "Values", ylab = "Frequency")
# Add a curve representing the probability density function of the normal distribution
curve(dnorm(x, mean = mean(data), sd = sd(data)), add = TRUE, col = "red")
```

In this example, we first generate a sample of random data following a normal distribution using the `rnorm()` function. We then create a histogram plot of the data using the `hist()` function, which visualizes the distribution of the data.

To overlay a curve representing the probability density function of the normal distribution on the histogram, we use the `curve()` function with the `dnorm()` function to calculate the probability density at each point.

This simple R Markdown code provides a visual and practical demonstration of the normal distribution, allowing readers to understand and appreciate this fundamental concept in statistics. “