Analisis Acciones

Librerias

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.5.1     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.1
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(quantmod)
```

```
## Warning: package 'quantmod' was built under R version 4.4.3
```

```
## Cargando paquete requerido: xts
## Cargando paquete requerido: zoo
## 
## Adjuntando el paquete: 'zoo'
## 
## The following objects are masked from 'package:base':
## 
##     as.Date, as.Date.numeric
## 
## 
## ######################### Warning from 'xts' package ##########################
## #                                                                             #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                            #
## #                                                                             #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
## # dplyr from breaking base R's lag() function.                                #
## #                                                                             #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                             #
## ###############################################################################
## 
## Adjuntando el paquete: 'xts'
## 
## The following objects are masked from 'package:dplyr':
## 
##     first, last
## 
## Cargando paquete requerido: TTR
```

```
## Warning: package 'TTR' was built under R version 4.4.3
```

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

Referencias sobre Analisis Tecnico

Basico: https://corporatefinanceinstitute.com/resources/career-map/sell-side/capital-markets/technical-analysis/

Velas: https://bookdown.org/kochiuyu/technical-analysis-with-r-second-edition2/candle-stick-pattern.html

Dojis: https://bookdown.org/kochiuyu/technical-analysis-with-r-second-edition2/doji.html

Estrategias: https://trendspider.com/learning-center/technical-analysis-strategies/

Estrategias de Cruces de Promedios Moviles: implementar gloden cross

Estrategia MACD: implementar regla sobre cero (https://www.youtube.com/watch?v=W78Xg_pnJ1A)

Estrategia RSI: implementar regla 30-70

Avanzado (basado en el paquete "", no lo usamos): https://rpubs.com/jwcb1025/quantstrat_trading_strategy

Analisis Tecnico con R: https://bookdown.org/kochiuyu/technical-analysis-with-r-second-edition2/

Usando xts: https://rpubs.com/odenipinedo/manipulating-time-series-data-with-xts-and-zoo-in-R

Datos

Fuente: https://www.kaggle.com/datasets/jakewright/9000-tickers-of-stock-market-data-full-history?resource=download

```
# datos<-read_csv("/home/andresfaral/Downloads/all_stock_data.csv")
# dim(datos)
# tabla<-table(datos$Ticker)
# maximo<-max(tabla)
# tickers<-names(tabla[tabla==maximo])
# datos.filt<-datos %>% filter(Ticker %in% tickers)
# datos.filt
# save(datos.filt,file="stock_comp")
# Cargo datos de Empresas con la historia COMPLETA
load(file="stock_comp")
datos.filt
```

Date	Ticker	Open	High	Low	Close	Volume	Dividends	Stock Splits
<date>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1962-01-02	ED	0.000000000	0.265827556	0.261787623	0.261787623	25600	0.00000	0.00
1962-01-02	CVX	0.000000000	0.046808902	0.046069266	0.046808902	105840	0.00000	0.00
1962-01-02	GD	0.000000000	0.210032760	0.203060708	0.208289742	2648000	0.00000	0.00
1962-01-02	BP	0.000000000	0.141439331	0.139527977	0.139527977	77440	0.00000	0.00
1962-01-02	MSI	0.000000000	0.764922976	0.745253521	0.751810193	65671	0.00000	0.00
1962-01-02	HON	0.000000000	1.559642297	1.549127912	1.556137562	40740	0.00000	0.00
1962-01-02	FL	0.000000000	0.972249303	0.953805589	0.959075153	49200	0.00000	0.00
1962-01-02	GT	0.000000000	1.946900130	1.914270519	1.936023593	32000	0.00000	0.00
1962-01-02	JNJ	0.000000000	0.067765735	0.067414438	0.067765735	0	0.00000	0.00
1962-01-02	MMM	0.000000000	0.541890853	0.525952884	0.529937267	254509	0.00000	0.00

1-10 of 10,000 rows   Previous 1 2 3 4 5 6 ... 1000 Next

EDA

Summary

```
summary(datos.filt)
```

```
##       Date               Ticker               Open              High        
##  Min.   :1962-01-02   Length:458751      Min.   :  0.0000   Min.   :  0.0019  
##  1st Qu.:1977-10-14   Class :character   1st Qu.:  0.7825   1st Qu.:  1.0854  
##  Median :1993-06-08   Mode  :character   Median :  6.9453   Median :  7.2564  
##  Mean   :1993-06-11                      Mean   : 24.1942   Mean   : 24.5616  
##  3rd Qu.:2009-02-19                      3rd Qu.: 27.8408   3rd Qu.: 28.1586  
##  Max.   :2024-11-04                      Max.   :479.0400   Max.   :480.2800  
##       Low                Close              Volume             Dividends      
##  Min.   :  0.0019   Min.   :  0.0019   Min.   :        0   Min.   : 0.0000  
##  1st Qu.:  1.0637   1st Qu.:  1.0746   1st Qu.:   521442   1st Qu.: 0.0000  
##  Median :  7.0900   Median :  7.1800   Median :  1983700   Median : 0.0000  
##  Mean   : 24.0654   Mean   : 24.3183   Mean   :  4113892   Mean   : 0.0036  
##  3rd Qu.: 27.5219   3rd Qu.: 27.8440   3rd Qu.:  5301103   3rd Qu.: 0.0000  
##  Max.   :476.3700   Max.   :477.6200   Max.   :442012304   Max.   :51.0600  
##   Stock Splits    
##  Min.   :0.000000
##  1st Qu.:0.000000
##  Median :0.000000
##  Mean   :0.000686
##  3rd Qu.:0.000000
##  Max.   :4.000000
```

Tabla de tickers

```
table(datos.filt$Ticker)
```

```
##
##    AA   AEP    BA    BP   CAT   CNP   CVX   DIS   DTE    ED    FL    GD    GE
## 15819 15819 15819 15819 15819 15819 15819 15819 15819 15819 15819 15819 15819
##    GT   HON   HPQ   IBM    IP   JNJ    KO    KR   MMM    MO   MRK   MRO   MSI
## 15819 15819 15819 15819 15819 15819 15819 15819 15819 15819 15819 15819 15819
##    PG   XOM   XRX
## 15819 15819 15819
```

Usando quantmod

Fuente: https://www.quantmod.com/

Convierto una serie a xts para manejarlo con quantmod

```
# convierto 1 accion a xts
datos.filt2<-datos.filt %>% filter(Ticker=="KO") # just coke
KO.xts<- xts(datos.filt2[,c(3:7)],order.by = datos.filt2$Date)
#
class(KO.xts)
```

```
## [1] "xts" "zoo"
```

```
head(KO.xts)
```

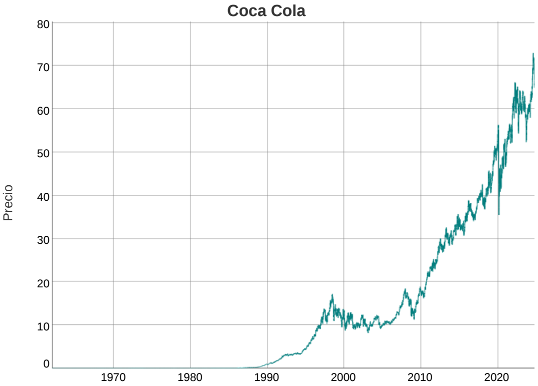
```
##                   Open        High         Low       Close  Volume
## 1962-01-02 0.004007111 0.004116209 0.004007111 0.004007111  806400
## 1962-01-03 0.003947602 0.003947602 0.003858325 0.003917833 1574400
## 1962-01-04 0.003927766 0.003977356 0.003927766 0.003947602  844800
## 1962-01-05 0.003947603 0.003997192 0.003848408 0.003858326 1420800
## 1962-01-08 0.003828574 0.003828574 0.003744263 0.003818656 2035200
## 1962-01-09 0.003818655 0.003907917 0.003788901 0.003888081  960000
```

Visualizacion usando dygraphs

```
require(dygraphs)
```

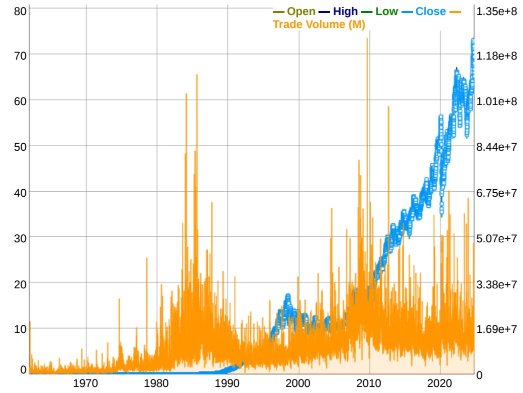
```
## Cargando paquete requerido: dygraphs
```

```
dygraph(KO.xts$Close, main = "Coca Cola") %>%
  dyAxis("y", label = "Precio") %>%
  dyOptions(stackedGraph = FALSE)
```



Visualizando Precio al cierre y Volumen

```
dygraph(KO.xts) %>%
  dySeries("Close", label = "Close", color = "#0198f9", drawPoints = TRUE, pointSize = 3, pointShape = "square")
%>%
  dySeries("Volume", label = "Trade Volume (M)", stepPlot = TRUE, fillGraph = TRUE, color = "#FF9900", axis = "y
2")
```



Traigo la accion de google de yahoo finance

```
getSymbols("GOOG",src="yahoo") # from google finance
```

```
## [1] "GOOG"
```

```
class(GOOG)
```

```
## [1] "xts" "zoo"
```

```
tail(GOOG)
```

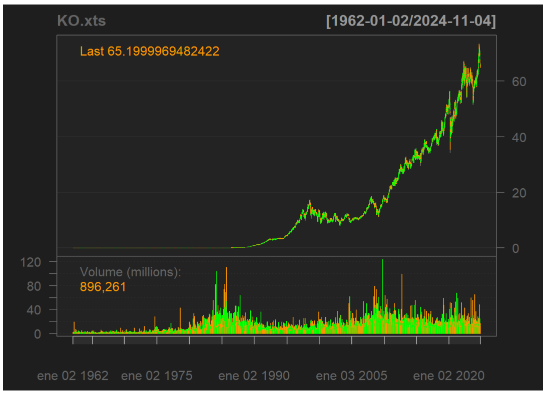
```
##            GOOG.Open GOOG.High GOOG.Low GOOG.Close GOOG.Volume GOOG.Adjusted
## 2025-04-25   167.100   168.240  163.000     163.85    35148100        163.85
## 2025-04-28   164.260   164.950  160.380     162.42    20871200        162.42
## 2025-04-29   162.045   162.680  159.390     162.06    15955200        162.06
## 2025-04-30   159.860   161.374  157.155     160.89    20639500        160.89
## 2025-05-01   162.520   163.940  160.930     162.79    21904300        162.79
## 2025-05-02   164.955   166.700  163.660     165.81    16832500        165.81
```

Charts financieros

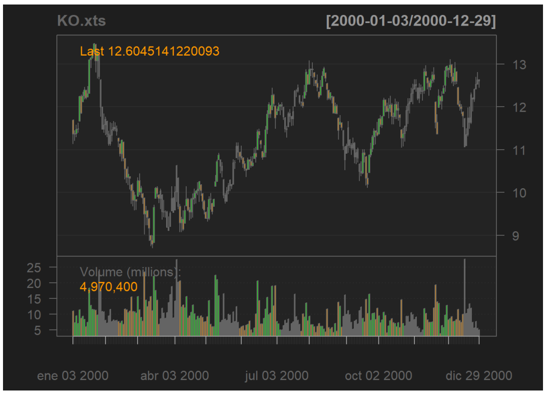
Charts financieros

Las series de OHLC

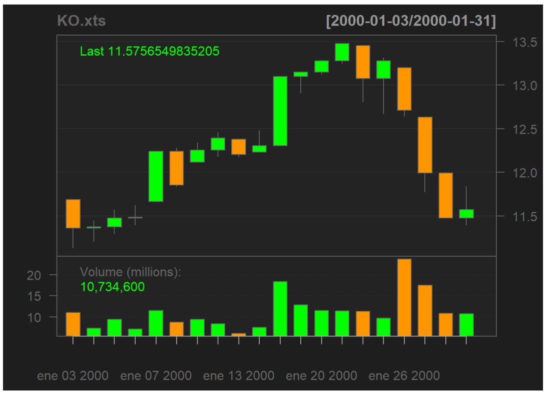
```
chartSeries(KO.xts) # toda la serie
```



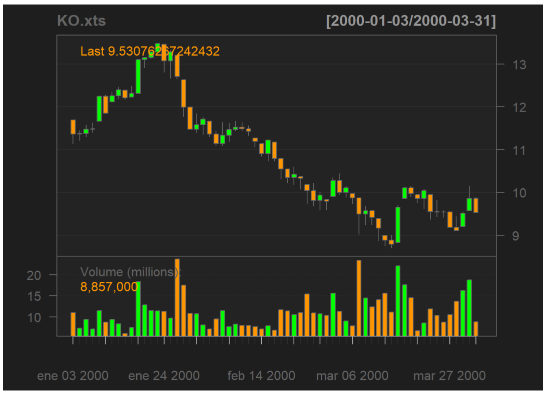
```
chartSeries(KO.xts,subset="2000") # un anio especifico
```



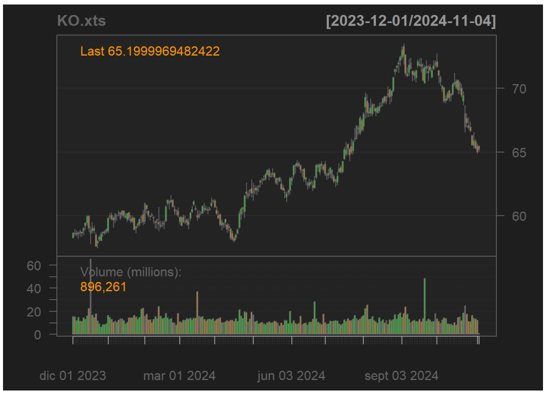
```
chartSeries(KO.xts,subset="2000-01") # un anio y mes especifico
```



```
chartSeries(KO.xts,subset="2000-01::2000-03") # un periodo
```



```
chartSeries(KO.xts,subset="last 12 months") # ultimo anio
```



Agregado de Indicadores

Agregado de indicadores

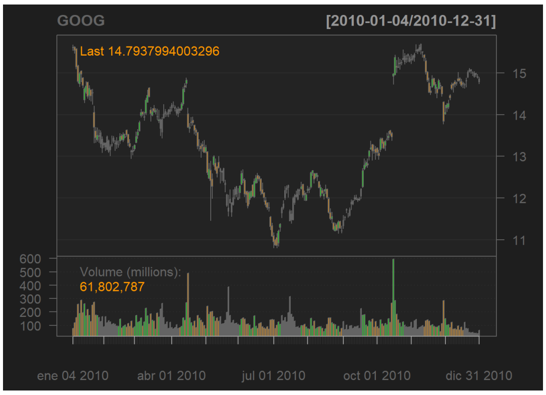
Mirar:

Moving Averages: https://en.wikipedia.org/wiki/Moving_average

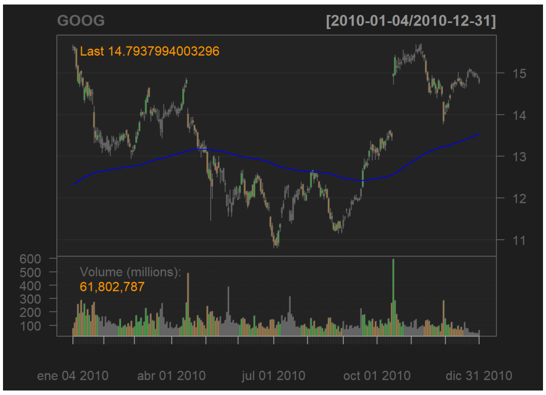
EMA: https://en.wikipedia.org/wiki/Exponential_smoothing

EMA en TTR: https://bookdown.org/kochiuyu/technical-analysis-with-r-second-edition2/exponential-moving-average-ema.html#ttr-1

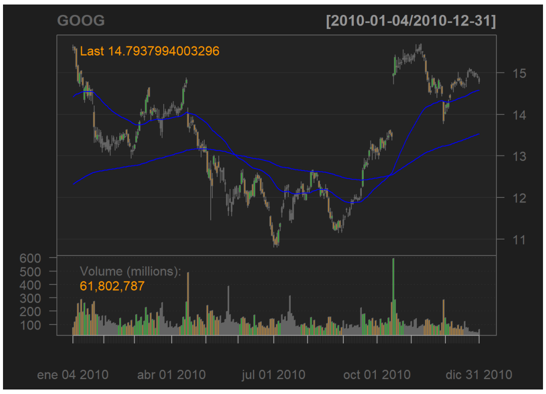
```
# por separado
chartSeries(GOOG,subset="2010")
```



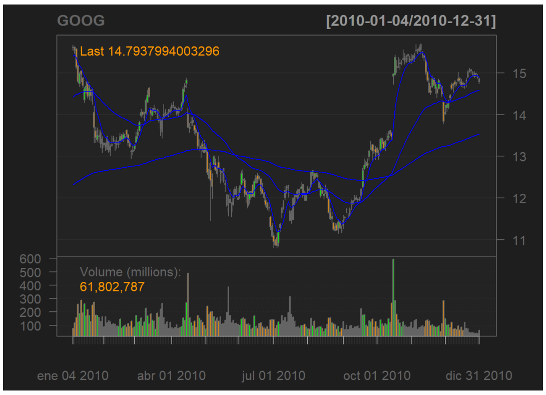
```
addEMA(200)
```



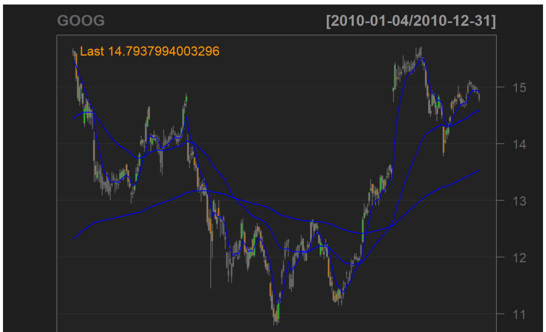
```
addEMA(50)
```

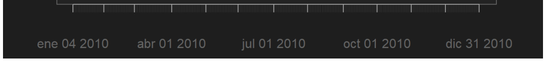


```
addEMA(7)
```



```
# todo junto
chartSeries(GOOG,subset="2010",TA="addEMA(200);addEMA(50);addEMA(7)")
```





Calculando Indicadores

```
ema7<-EMA(GOOG$GOOG.Open,7)
class(ema7)
```

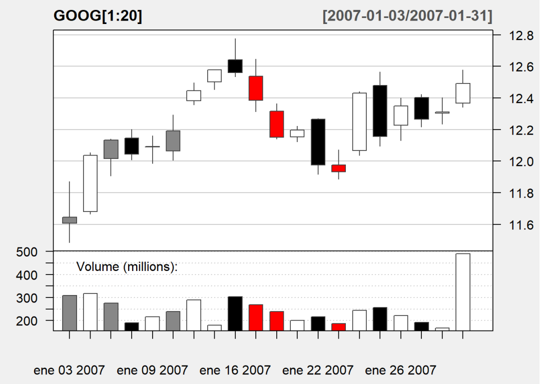
```
## [1] "xts" "zoo"
```

```
head(ema7,10)
```

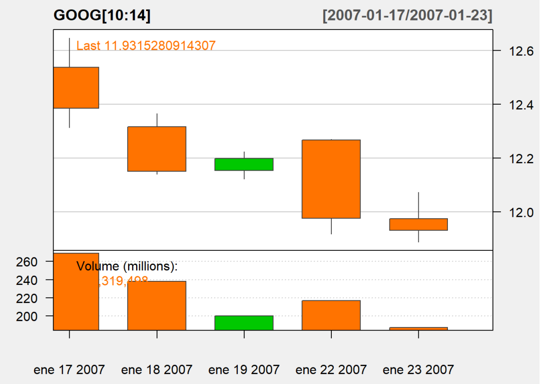
```
##                 EMA
## 2007-01-03       NA
## 2007-01-04       NA
## 2007-01-05       NA
## 2007-01-08       NA
## 2007-01-09       NA
## 2007-01-10       NA
## 2007-01-11 11.99885
## 2007-01-12 12.12486
## 2007-01-16 12.25398
## 2007-01-17 12.32493
```

Mas visualizaciones

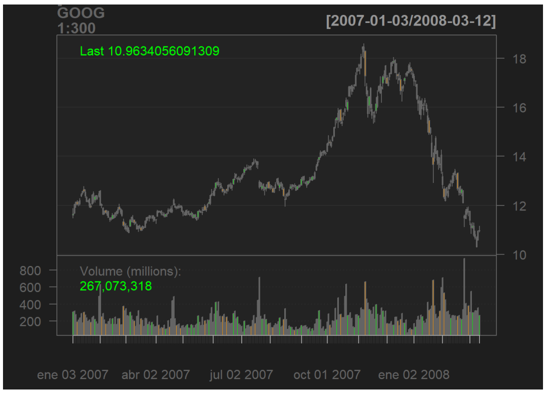
```
candleChart(GOOG[1:20],multi.col=TRUE,theme="white")
```



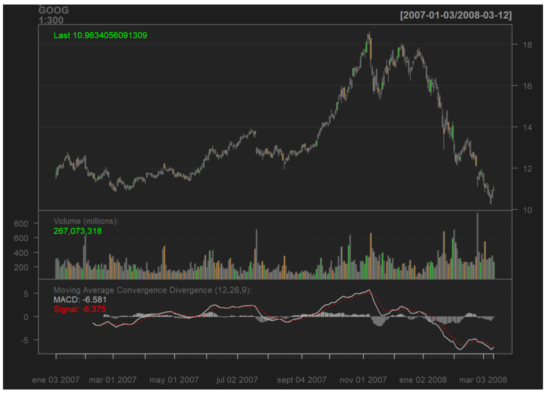
```
candleChart(GOOG[10:14],multi.col=FALSE,theme="white")
```



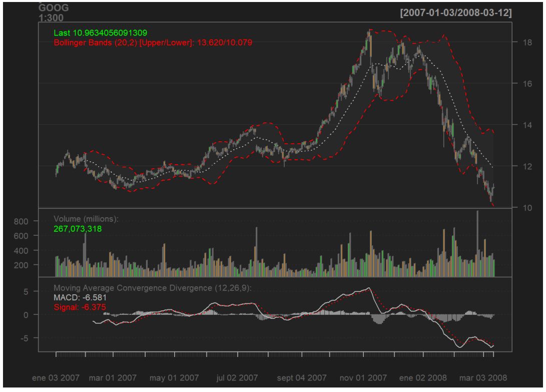
```
chartSeries(GOOG[1:300,])
```



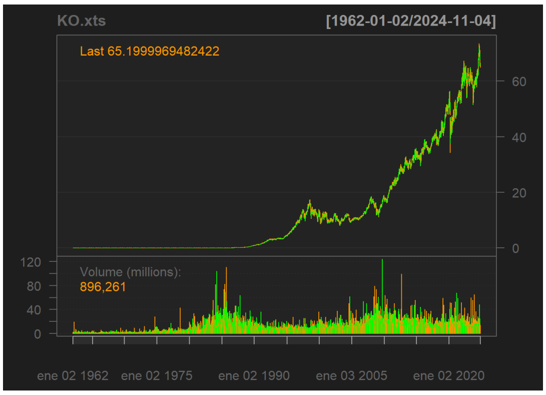
```
addMACD()
```



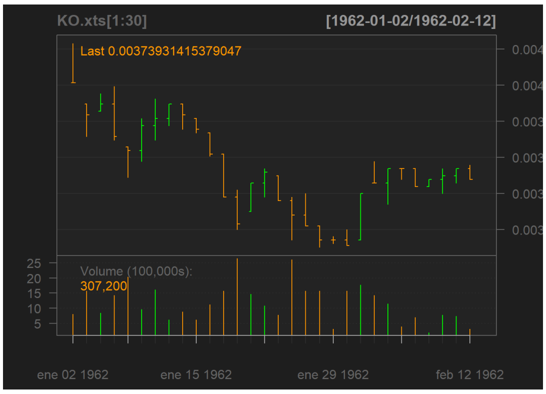
```
addBBands()
```



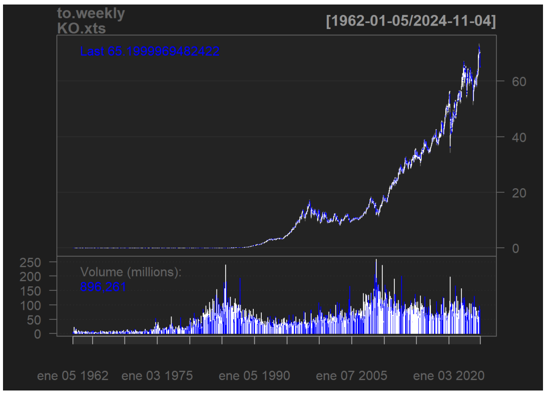
```
barChart(KO.xts)
```



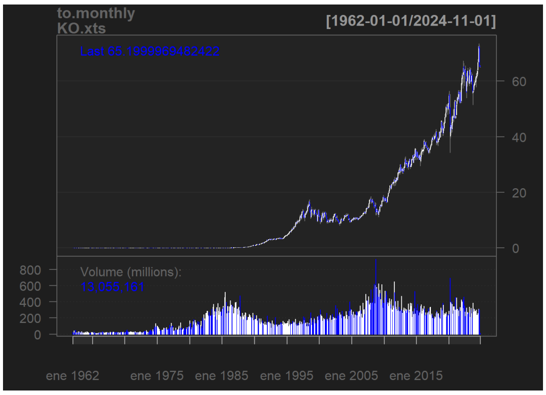
```
barChart(KO.xts[1:30])
```



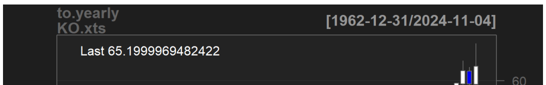
```
chartSeries(to.weekly(KO.xts),up.col='white',dn.col='blue')
```

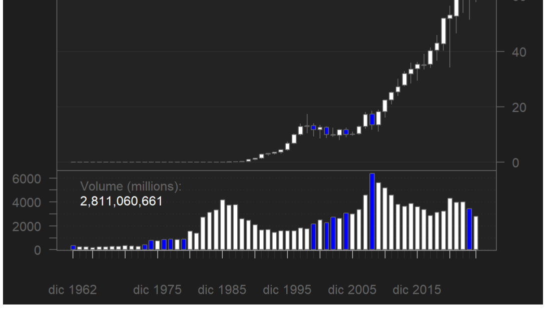


```
chartSeries(to.monthly(KO.xts),up.col='white',dn.col='blue')
```

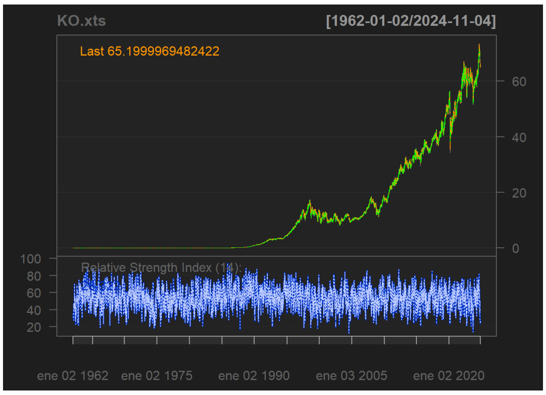


```
chartSeries(to.yearly(KO.xts),up.col='white',dn.col='blue')
```





```
chartSeries(KO.xts,TA="addRSI()")
```



Estrategias de Trading

Se compra 1 accion con la señial bulish y se vende 1 accion con la selian bearish Se empiezo siempre con una compra Se comienza con saldo 0, una compra disminuye/aumenta el saldo en el precio de compra/venta No puede haber mas ventas que compras (no se puede shortear) Al final del periodo se vende todo el stock (si hubiera) El resultado (target) es el saldo al final del

Deteccion de dojis

```
datos.filt2.doji <- datos.filt2 %>%
  mutate(
    dragonfly = ifelse(
      (abs(Close - Open) / ((Open + Close) / 2) < 0.005) &
      (abs(((Open + Close) / 2) - High) / ((((Open + Close) / 2) + High) / 2) < 0.005) &
      (abs(((Open + Close) / 2) - Low) / ((((Open + Close) / 2) + Low) / 2) > 0.01),
      1, 0
    ),
    gravestone = ifelse(
      (abs(Close - Open) / ((Open + Close) / 2) < 0.005) &
      (abs(((Open + Close) / 2) - Low) / ((((Open + Close) / 2) + Low) / 2) < 0.005) &
      (abs(((Open + Close) / 2) - High) / ((((Open + Close) / 2) + High) / 2) > 0.01),
      1, 0
    )
  )

sum(datos.filt2.doji$dragonfly)
```

```
## [1] 618
```

```
sum(datos.filt2.doji$gravestone)
```

```
## [1] 523
```

Calculo de proporciones poblacionales

```
datos.filt2.doji <- datos.filt2.doji %>%
  mutate(increase_nd = ifelse(is.na(lead(Close)), FALSE, Close < lead(Close)))

p_pob_increase <- sum(datos.filt2.doji$increase_nd)/nrow(datos.filt2.doji)

datos.filt2.doji.dragon <- datos.filt2.doji %>%
  filter(dragonfly == 1)

p_dragon_increase <- sum(datos.filt2.doji.dragon$increase_nd)/nrow(datos.filt2.doji.dragon)

datos.filt2.doji.grave <- datos.filt2.doji %>%
  filter(gravestone == 1)

p_grave_decrease <- 1 - sum(datos.filt2.doji.grave$increase_nd)/nrow(datos.filt2.doji.grave)

p_pob_decrease <- 1 - p_pob_increase

p_pob_increase
```

```
## [1] 0.4928251
```

```
p_dragon_increase
```

```
## [1] 0.4110032
```

```
p_pob_decrease
```

```
## [1] 0.5071749
```

```
p_grave_decrease
```

```
## [1] 0.499044
```

Calculo de la diferencia porcentual

```
datos.filt2.doji <- datos.filt2.doji %>%
  mutate(porc_diff_nd = ifelse(is.na(lead(Close)), 0, (lead(Close) - Close)/Close))

p_mean_diff <- mean(datos.filt2.doji$porc_diff_nd)

datos.filt2.doji.dragon <- datos.filt2.doji %>%
  filter(dragonfly == 1)

mean_dragon_diff <- mean(datos.filt2.doji.dragon$porc_diff_nd)

datos.filt2.doji.grave <- datos.filt2.doji %>%
  filter(gravestone == 1)

mean_grave_diff <- mean(datos.filt2.doji.grave$porc_diff_nd)
```

Aplicandolo a dias posteriores

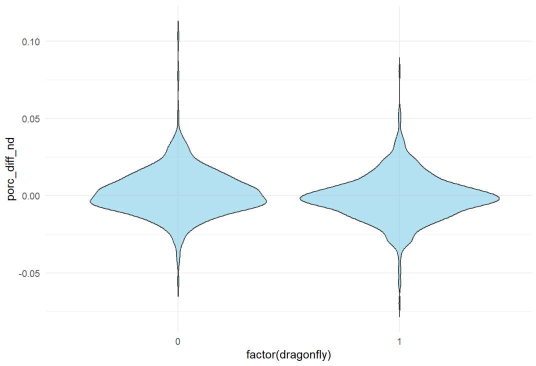
```
library(dplyr)
```

```r
calcular_dif_porcentual <- function(datos) {
  datos <- datos %>%
    mutate(
      diff_1 = (lead(Close, 1) - Close) / Close,
      diff_2 = (lead(Close, 2) - Close) / Close,
      diff_3 = (lead(Close, 3) - Close) / Close,
      diff_4 = (lead(Close, 4) - Close) / Close,
      diff_5 = (lead(Close, 5) - Close) / Close,
      diff_6 = (lead(Close, 6) - Close) / Close,
      diff_7 = (lead(Close, 7) - Close) / Close
    )

  # Función auxiliar para calcular los promedios de diferencias
  calcular_promedios <- function(df) {
    colMeans(df %>% select(starts_with("diff_")), na.rm = TRUE)
  }

  # Calcular promedios para la población total, con Gravestone y con Dragonfly
  total_mean <- calcular_promedios(datos)
  grave_mean <- calcular_promedios(datos %>% filter(gravestone == 1))
  dragon_mean <- calcular_promedios(datos %>% filter(dragonfly == 1))

  # Crear la tabla final
  tabla_resultado <- rbind(
    "Total" = total_mean,
    "Gravestone Doji" = grave_mean,
    "Dragonfly Doji" = dragon_mean
  )

  return(as.data.frame(tabla_resultado))
}

# Uso de la función
tabla_doji <- calcular_dif_porcentual(datos.filt2.doji)
print(tabla_doji)
```

```
##                        diff_1        diff_2      diff_3      diff_4      diff_5
## Total             0.0007245497  0.0014559828 0.002179687 0.002903392 0.003622175
## Gravestone Doji   0.0019373561  0.0028653237 0.002862548 0.004107481 0.004407989
## Dragonfly Doji   -0.0012537717 -0.0001856587 0.001220299 0.001642666 0.001952062
##                      diff_6      diff_7
## Total           0.004342560 0.005060004
## Gravestone Doji 0.005512973 0.007154211
## Dragonfly Doji  0.003270228 0.003611660
```

Distribucion de las diferencias porcentuales entre dojis

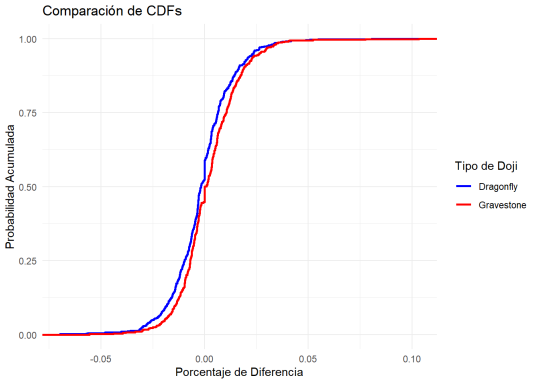
```r
datos.filt2.doji.gd <- datos.filt2.doji %>%
  filter(dragonfly == 1 | gravestone == 1)

ggplot(datos.filt2.doji.gd, aes(x = factor(dragonfly)
                                , y = porc_diff_nd, fill = dragonfly)) +
  geom_violin(fill = "skyblue", alpha = 0.6, trim = FALSE) +
  theme_minimal() +
  labs(title = "")
```



### Grafico de densidad para dojis

```r
library(ggplot2)
library(dplyr)

# Filtrar los datos
datos.filt2.doji.gd <- datos.filt2.doji %>%
  filter(dragonfly == 1 | gravestone == 1)

# Crear el gráfico de CDF
ggplot(datos.filt2.doji.gd, aes(x = porc_diff_nd, color = factor(dragonfly + 2 * gravestone))) +
  stat_ecdf(geom = "step", size = 1) +
  scale_color_manual(values = c("blue", "red"), labels = c("Dragonfly", "Gravestone")) +
  labs(title = "Comparación de CDFs",
       x = "Porcentaje de Diferencia",
       y = "Probabilidad Acumulada",
       color = "Tipo de Doji") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Ampliamos la muestra

```r
tickers <- list('JNJ','PG','KO','BP','CVX')
muestra <- datos.filt %>%
  filter(Ticker %in% tickers)

muestra.doji <- muestra %>%
  mutate(
    dragonfly = ifelse(
      (abs(Close - Open) / ((Open + Close) / 2) < 0.005) &
      (abs(((Open + Close) / 2) - High) / ((((Open + Close) / 2) + High) / 2) < 0.005) &
      (abs(((Open + Close) / 2) - Low) / ((((Open + Close) / 2) + Low) / 2) > 0.01),
      1, 0
    ),
    gravestone = ifelse(
      (abs(Close - Open) / ((Open + Close) / 2) < 0.005) &
      (abs(((Open + Close) / 2) - Low) / ((((Open + Close) / 2) + Low) / 2) < 0.005) &
      (abs(((Open + Close) / 2) - High) / ((((Open + Close) / 2) + High) / 2) > 0.01),
      1, 0
    )
  )

sum(muestra.doji$dragonfly)
```

```
## [1] 2501
```

```
sum(muestra.doji$gravestone)
```

```
## [1] 2077
```

Proporciones poblacionales para la muestra ampliada

```
muestra.doji <- muestra.doji %>%
  mutate(increase_nd = ifelse(is.na(lead(Close)), FALSE, Close < lead(Close))) %>%
  arrange(Ticker)

p_pob_increase <- sum(muestra.doji$increase_nd)/nrow(muestra.doji)

muestra.doji.dragon <- muestra.doji %>%
  filter(dragonfly == 1)

p_dragon_increase <- sum(muestra.doji.dragon$increase_nd)/nrow(muestra.doji.dragon)

muestra.doji.grave <- muestra.doji %>%
  filter(gravestone == 1)

p_grave_decrease <- 1 - sum(muestra.doji.grave$increase_nd)/nrow(muestra.doji.grave)

p_pob_decrease <- 1 - p_pob_increase

p_pob_increase
```

```
## [1] 0.5015108
```

```
p_dragon_increase
```

```
## [1] 0.5417833
```

```
p_pob_decrease
```

```
## [1] 0.4984892
```

```
p_grave_decrease
```

```
## [1] 0.4597978
```

Diferencias porcentuales para la muestra ampliada

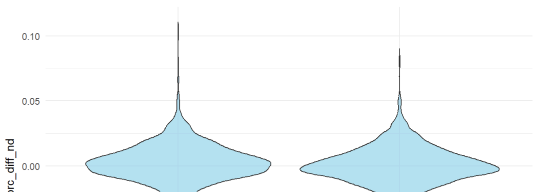
```
muestra.doji <- muestra.doji %>%
  mutate(porc_diff_nd = ifelse(is.na(lead(Close)), 0, (lead(Close) - Close)/Close))

p_mean_diff <- mean(muestra.doji$porc_diff_nd)

muestra.doji.dragon <- muestra.doji %>%
  filter(dragonfly == 1)

mean_dragon_diff <- mean(muestra.doji.dragon$porc_diff_nd)

muestra.doji.grave <- muestra.doji %>%
  filter(gravestone == 1)

mean_grave_diff <- mean(muestra.doji.grave$porc_diff_nd)
```

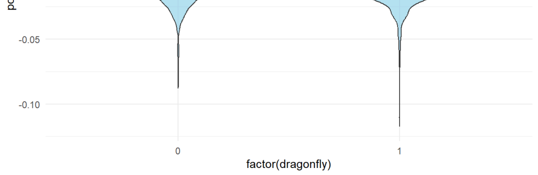
```
calcular_dif_porcentual <- function(datos) {
  datos <- datos %>%
    mutate(
      diff_1 = (lead(Close, 1) - Close) / Close,
      diff_2 = (lead(Close, 2) - Close) / Close,
      diff_3 = (lead(Close, 3) - Close) / Close,
      diff_4 = (lead(Close, 4) - Close) / Close,
      diff_5 = (lead(Close, 5) - Close) / Close,
      diff_6 = (lead(Close, 6) - Close) / Close,
      diff_7 = (lead(Close, 7) - Close) / Close
    )

  # Función auxiliar para calcular los promedios de diferencias
  calcular_promedios <- function(df) {
    colMeans(df %>% select(starts_with("diff_")), na.rm = TRUE)
  }

  # Calcular promedios para la población total, con Gravestone y con Dragonfly
  total_mean <- calcular_promedios(datos)
  grave_mean <- calcular_promedios(datos %>% filter(gravestone == 1))
  dragon_mean <- calcular_promedios(datos %>% filter(dragonfly == 1))

  # Crear la tabla final
  tabla_resultado <- rbind(
    "Total" = total_mean,
    "Gravestone Doji" = grave_mean,
    "Dragonfly Doji" = dragon_mean
  )

  return(as.data.frame(tabla_resultado))
}

# Uso de la función
tabla_acciones <- calcular_dif_porcentual(muestra.doji)
print(tabla_acciones)
```

```
##                       diff_1       diff_2      diff_3      diff_4      diff_5
## Total           5.403829e-04 0.0010864445 0.001625822 0.002163637 0.002695216
## Gravestone Doji 2.000267e-03 0.0032636286 0.004154459 0.005103181 0.005753727
## Dragonfly Doji  3.745869e-05 0.0002723454 0.001118714 0.001484956 0.002207417
##                      diff_6      diff_7
## Total           0.003227896 0.003757427
## Gravestone Doji 0.006845586 0.006844590
## Dragonfly Doji  0.002806054 0.003325673
```

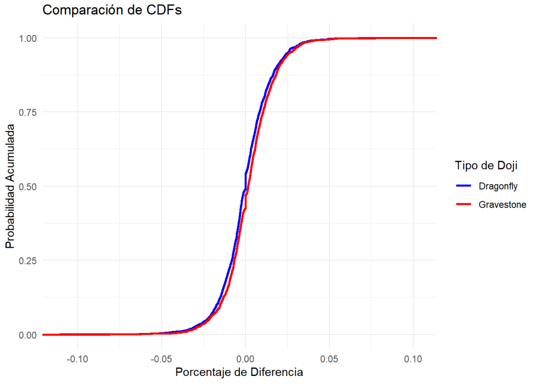
Graficos para la muestra ampliada

```
muestra.doji.gd <- muestra.doji %>%
  filter(dragonfly == 1 | gravestone == 1)

ggplot(muestra.doji.gd, aes(x = factor(dragonfly)
                              , y = porc_diff_nd, fill = dragonfly)) +
  geom_violin(fill = "skyblue", alpha = 0.6, trim = FALSE) +
  theme_minimal() +
  labs(title = "")
```





```
muestra.doji.gd <- muestra.doji %>%
  filter(dragonfly == 1 | gravestone == 1)


ggplot(muestra.doji.gd, aes(x = porc_diff_nd, color = factor(dragonfly + 2 * gravestone))) +
  stat_ecdf(geom = "step", size = 1) +
  scale_color_manual(values = c("blue", "red"), labels = c("Dragonfly", "Gravestone")) +
  labs(title = "Comparación de CDFs",
       x = "Porcentaje de Diferencia",
       y = "Probabilidad Acumulada",
       color = "Tipo de Doji") +
  theme_minimal()
```



Test de Hipotesis

Para cada tipo de Doji, compararemos la proporción observada con la proporción general de la población utilizando una prueba de proporciones.

Hipótesis para Dragonfly Doji

H0: La probabilidad de aumento después de un Dragonfly Doji es igual a la probabilidad general de aumento (pdragon = ppob) Ha: La probabilidad de aumento después de un Dragonfly Doji es diferente de la probabilidad general (pdragon != ppob)

Hipótesis para Gravestone Doji H0: La probabilidad de disminución después de un Gravestone Doji es igual a la probabilidad general de disminución (pgrave = ppob) Ha: La probabilidad de disminución después de un Gravestone Doji es diferente de la probabilidad general (pgrave != ppob)

```
# Calcular proporciones generales
p_pob_increase <- sum(muestra.doji$increase_nd) / nrow(muestra.doji)
p_pob_decrease <- 1 - p_pob_increase

# Número de observaciones en cada subconjunto
n_dragon <- nrow(muestra.doji.dragon)
n_grave <- nrow(muestra.doji.grave)

# Casos donde el precio aumentó después de un Dragonfly Doji
x_dragon <- sum(muestra.doji.dragon$increase_nd)

# Casos donde el precio disminuyó después de un Gravestone Doji
x_grave <- sum(muestra.doji.grave$increase_nd)

# Prueba de proporciones para Dragonfly Doji
test_dragon <- prop.test(x_dragon, n_dragon, p = p_pob_increase, alternative = "two.sided")

# Prueba de proporciones para Gravestone Doji
test_grave <- prop.test(n_grave - x_grave, n_grave, p = p_pob_decrease, alternative = "two.sided")

# Mostrar resultados
test_dragon
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  x_dragon out of n_dragon, null probability p_pob_increase
## X-squared = 16.065, df = 1, p-value = 6.122e-05
## alternative hypothesis: true p is not equal to 0.5015108
## 95 percent confidence interval:
##  0.5220066 0.5614305
## sample estimates:
##         p
## 0.5417833
```

```
test_grave
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  n_grave - x_grave out of n_grave, null probability p_pob_decrease
## X-squared = 12.283, df = 1, p-value = 0.0004571
## alternative hypothesis: true p is not equal to 0.4984892
## 95 percent confidence interval:
##  0.4382188 0.4815269
## sample estimates:
##         p
## 0.4597978
```

```
datos.filt.doji <- datos.filt %>%
  arrange(Ticker) %>%
  mutate(
    dragonfly = ifelse(
      (abs(Close - Open) / ((Open + Close) / 2) < 0.005) &
      (abs(((Open + Close) / 2) - High) / ((((Open + Close) / 2) + High) / 2) < 0.005) &
      (abs(((Open + Close) / 2) - Low) / ((((Open + Close) / 2) + Low) / 2) > 0.01),
      1, 0
    ),
    gravestone = ifelse(
      (abs(Close - Open) / ((Open + Close) / 2) < 0.005) &
      (abs(((Open + Close) / 2) - Low) / ((((Open + Close) / 2) + Low) / 2) < 0.005) &
      (abs(((Open + Close) / 2) - High) / ((((Open + Close) / 2) + High) / 2) > 0.01),
      1, 0
    ),
    increase_nd = ifelse(is.na(lead(Close)), FALSE, Close < lead(Close)),
    porc_diff_nd = ifelse(is.na(lead(Close)), 0, (lead(Close) - Close)/Close)
  )

datos.filt.dragon <- datos.filt.doji %>%
  filter(dragonfly == 1)

datos.filt.grave <- datos.filt.doji %>%
  filter(gravestone == 1)

# Calcular proporciones generales
p_pob_increase2 <- sum(datos.filt.doji$increase_nd) / nrow(datos.filt.doji)
p_pob_decrease2 <- 1 - p_pob_increase2

# Número de observaciones en cada subconjunto
n_dragon2 <- nrow(datos.filt.dragon)
n_grave2 <- nrow(datos.filt.grave)

# Casos donde el precio aumentó después de un Dragonfly Doji
```

```
x_dragon2 <- sum(datos.filt.dragon$increase_nd)

# Casos donde el precio disminuyó después de un Gravestone Doji
x_grave2 <- sum(datos.filt.grave$increase_nd)

# Prueba de proporciones para Dragonfly Doji
test_dragon2 <- prop.test(x_dragon2, n_dragon2, p = p_pob_increase2, alternative = "two.sided")

# Prueba de proporciones para Gravestone Doji
test_grave2 <- prop.test(n_grave2 - x_grave2, n_grave2, p = p_pob_decrease2, alternative = "two.sided")

# Mostrar resultados
test_dragon2
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  x_dragon2 out of n_dragon2, null probability p_pob_increase2
## X-squared = 98.763, df = 1, p-value < 2.2e-16
## alternative hypothesis: true p is not equal to 0.4776535
## 95 percent confidence interval:
##  0.4300000 0.4456578
## sample estimates:
##         p
## 0.4378135
```

```
test_grave2
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  n_grave2 - x_grave2 out of n_grave2, null probability p_pob_decrease2
## X-squared = 41.863, df = 1, p-value = 9.788e-11
## alternative hypothesis: true p is not equal to 0.5223465
## 95 percent confidence interval:
##  0.4865403 0.5032393
## sample estimates:
##         p
## 0.4948884
```

```
# Calcular la media poblacional general
pob_mean <- mean(datos.filt.doji$porc_diff_nd)

# Prueba t para Dragonfly Doji
test_dragon3 <- t.test(datos.filt.dragon$porc_diff_nd,
                       mu = pob_mean,
                       alternative = "two.sided")

# Prueba t para Gravestone Doji
test_grave3 <- t.test(datos.filt.grave$porc_diff_nd,
                       mu = pob_mean,
                       alternative = "two.sided")

# Mostrar resultados
test_dragon3
```

```
##
##  One Sample t-test
##
## data:  datos.filt.dragon$porc_diff_nd
## t = -8.3215, df = 15549, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0.0004795564
## 95 percent confidence interval:
##  -0.0012818762 -0.0006102639
## sample estimates:
##   mean of x
## -0.00094607
```

```
test_grave3
```

```
##
##  One Sample t-test
##
## data:  datos.filt.grave$porc_diff_nd
## t = 6.1577, df = 13889, p-value = 7.583e-10
## alternative hypothesis: true mean is not equal to 0.0004795564
## 95 percent confidence interval:
##  0.001126624 0.001730944
## sample estimates:
##   mean of x
## 0.001428784
```

Golden y Death Cross

```
# Load necessary library
library(xts)

# Example data (assuming df is your data frame)
datos.filt2$Date <- as.Date(datos.filt2$Date)  # Convert Date column to Date type

# Convert to xts object (excluding non-numeric columns)
df_xts <- xts(datos.filt2[, -c(1,2)], order.by = datos.filt2$Date)

# View the xts object
print(df_xts)
```

```
##                   Open        High         Low       Close   Volume
## 1962-01-02  0.004007111  0.004116209  0.004007111  0.004007111   806400
## 1962-01-03  0.003947602  0.003947602  0.003858325  0.003917833  1574400
## 1962-01-04  0.003927766  0.003977356  0.003927766  0.003947602   844800
## 1962-01-05  0.003947603  0.003997192  0.003848408  0.003858326  1420800
## 1962-01-08  0.003828574  0.003828574  0.003744263  0.003818656  2035200
## 1962-01-09  0.003818655  0.003907917  0.003788901  0.003888081   960000
## 1962-01-10  0.003888078  0.003962470  0.003848406  0.003907914  1612800
## 1962-01-11  0.003907915  0.003947602  0.003888079  0.003947602   614400
## 1962-01-12  0.003947602  0.003947602  0.003878161  0.003917833   883200
## 1962-01-15  0.003907919  0.003907919  0.003868248  0.003878166   614400
##         ...
## 2024-10-22 69.000000000 69.750000000 68.680000305 69.449996948 18603300
## 2024-10-23 66.989997864 68.699996948 66.580001831 68.010002136 24655200
## 2024-10-24 67.650001526 68.040000916 66.949996948 67.300003052 17568800
## 2024-10-25 67.069999695 67.699996948 66.790000916 66.919998169 11138100
## 2024-10-28 66.959999084 67.400001526 66.599998474 66.669998169 10761400
## 2024-10-29 66.290000916 66.339996338 65.519996643 65.559997559 16525900
## 2024-10-30 65.510002136 66.540000916 65.319999695 65.919998169 14177800
## 2024-10-31 65.809997559 65.989997864 65.260002136 65.309997559 13383700
## 2024-11-01 65.470001221 65.660003662 64.889999390 65.010002136 12158900
## 2024-11-04 65.470001221 65.419998169 65.019996643 65.199996948   896261
##            Dividends Stock.Splits
## 1962-01-02         0            0
## 1962-01-03         0            0
## 1962-01-04         0            0
## 1962-01-05         0            0
## 1962-01-08         0            0
## 1962-01-09         0            0
## 1962-01-10         0            0
## 1962-01-11         0            0
## 1962-01-12         0            0
## 1962-01-15         0            0
##         ...
## 2024-10-22         0            0
## 2024-10-23         0            0
## 2024-10-24         0            0
## 2024-10-25         0            0
## 2024-10-28         0            0
## 2024-10-29         0            0
## 2024-10-30         0            0
## 2024-10-31         0            0
## 2024-11-01         0            0
## 2024-11-04         0            0
```

Calculamos los EMAs

```
macd50 <- EMA(df_xts$Close, 50)

macd200 <- EMA(df_xts$Close, 200)

macd50
```

```
##                 EMA
## 1962-01-02       NA
## 1962-01-03       NA
## 1962-01-04       NA
## 1962-01-05       NA
## 1962-01-08       NA
## 1962-01-09       NA
## 1962-01-10       NA
## 1962-01-11       NA
## 1962-01-12       NA
## 1962-01-15       NA
##        ...
## 2024-10-22 69.83285
## 2024-10-23 69.76137
## 2024-10-24 69.66484
## 2024-10-25 69.55720
## 2024-10-28 69.44398
## 2024-10-29 69.29167
## 2024-10-30 69.15944
## 2024-10-31 69.00849
## 2024-11-01 68.85168
## 2024-11-04 68.70848
```

```
macd200
```

```
##                 EMA
## 1962-01-02       NA
## 1962-01-03       NA
## 1962-01-04       NA
## 1962-01-05       NA
## 1962-01-08       NA
## 1962-01-09       NA
## 1962-01-10       NA
## 1962-01-11       NA
## 1962-01-12       NA
## 1962-01-15       NA
##        ...
## 2024-10-22 65.43913
## 2024-10-23 65.46471
## 2024-10-24 65.48297
## 2024-10-25 65.49727
## 2024-10-28 65.50894
## 2024-10-29 65.50945
## 2024-10-30 65.51353
## 2024-10-31 65.51151
## 2024-11-01 65.50652
## 2024-11-04 65.50347
```

```
#resta de los dos
```

Identificamos las cruces

```
library(xts)

macddf <- data.frame(
  Date = index(macd50),
  ema50 = coredata(macd50),
  ema200 = coredata(macd200)
)

macddf <- macddf %>%
  mutate(ema50may = EMA - EMA.1 > 0,
         cross = ifelse(ema50may != lead(ema50may),ifelse(ema50may == TRUE, -1, 1) , 0)) %>%
  left_join(datos.filt2, by = "Date")

macddf_cross <- macddf %>%
  filter(cross != 0)

golden <- count(macddf_cross, cross == 1)
```

Simulacion de estrategias

Simulamos una estrategia de trading basada en los resultados

```
macddf_valid <- macddf %>%
  filter(!is.na(Close), !is.na(cross))

initial_state <- list(
  cash = 10000,
  stocks = 0,
  saldo = 10000
)

# Acumulador paso a paso
sim_list <- accumulate(
  1:nrow(macddf_valid),
  .init = initial_state,
  .f = function(state, i) {
    row <- macddf_valid[i, ]

    new_state <- state

    if (!is.na(row$cross)) {
      if (row$cross == 1) {
        # Golden Cross → Comprar con todo el cash
        new_state$stocks <- state$cash / row$Close
        new_state$cash <- 0
      } else if (row$cross == -1) {
        # Death Cross → Vender todo
        new_state$cash <- state$stocks * row$Close
        new_state$stocks <- 0
      }
    }

    # Recalcular el saldo total
    new_state$saldo <- new_state$cash + new_state$stocks * row$Close
    return(new_state)
  }
)

# Convertir lista de estados a data frame
sim_df <- bind_rows(sim_list[-1])  # sacar .init
macddf_sim <- bind_cols(macddf_valid, sim_df)
```

Simulamos una estrategia de trading azarosa

```
n <- nrow(macddf_valid)
macddf_valid$azar <- 0

# Dividir en bloques
bloques <- floor(n / 75)

# Elegir una posición aleatoria dentro de cada bloque
idx_1 <- sapply(0:(75 - 1), function(i) {
  bloque_inicio <- i * bloques + 1
  bloque_fin <- min((i + 1) * bloques, n)
  sample(bloque_inicio:bloque_fin, 1)
})

# Asignar los 1
macddf_valid$azar[idx_1] <- 1
```

```
sim_list_aux <- accumulate(
  1:nrow(macddf_valid),
  .init = initial_state,
  .f = function(state, i) {
    row <- macddf_valid[i, ]

    new_state <- state
    if (!is.na(row$azar)) {
      if (row$azar == 1) {
        if(state$stocks == 0){
          new_state$stocks <- state$cash / row$Close
          new_state$cash <- 0
        }
        else if(state$stocks != 0){
          new_state$cash <- state$stocks * row$Close
          new_state$stocks <- 0
        }
      }
    }

    # Recalcular el saldo total
    new_state$saldo <- new_state$cash + new_state$stocks * row$Close
    return(new_state)
  }
)

# Convertir lista de estados a data frame
sim_df_aux <- bind_rows(sim_list_aux[-1])  # sacar .init
macddf_sim_azar <- bind_cols(macddf_valid, sim_df_aux)
```

Ahora lo repetimos 100 veces

```
library(purrr)
library(dplyr)

# Lista para guardar los resultados finales de saldo
resultados <- numeric(100)

# Loop de 100 repeticiones
for (k in 1:100) {
  # Paso 1: generar señales azarosas distribuidas
  n <- nrow(macddf_valid)
  macddf_valid$azar <- 0
  bloques <- floor(n / 75)
  idx_1 <- sapply(0:(75 - 1), function(i) {
    bloque_inicio <- i * bloques + 1
    bloque_fin <- min((i + 1) * bloques, n)
    sample(bloque_inicio:bloque_fin, 1)
  })
  macddf_valid$azar[idx_1] <- 1

  # Paso 2: correr simulación con accumulate
  initial_state <- list(stocks = 0, cash = 10000, saldo = 10000)

  sim_list_aux <- accumulate(
    1:nrow(macddf_valid),
    .init = initial_state,
    .f = function(state, i) {
      row <- macddf_valid[i, ]
      new_state <- state

      if (!is.na(row$azar) && row$azar == 1) {
        if (state$stocks == 0) {
          new_state$stocks <- state$cash / row$Close
          new_state$cash <- 0
        } else {
          new_state$cash <- state$cash + state$stocks * row$Close
          new_state$stocks <- 0
        }
      }

      new_state$saldo <- new_state$cash + new_state$stocks * row$Close
      return(new_state)
    }
  )

  sim_df_aux <- bind_rows(sim_list_aux[-1])

  # Paso 3: guardar el último saldo
  resultados[k] <- sim_df_aux$saldo[nrow(sim_df_aux)]
}

# Ver resultados
resultados
```

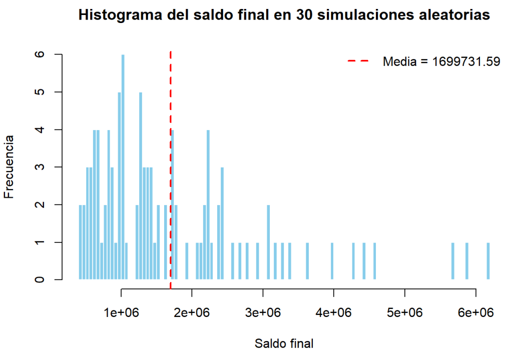
```
##   [1] 2231717.9 1321279.8 2136515.9 1001355.6 1370528.3 1284325.2  839868.4
##   [8]  656175.0 1411912.4  615999.4 1201015.9 1345927.2 1713289.0 2374761.4
##  [15] 1436412.4  977130.8 2229860.7 1628870.3 1273127.9 1272261.5  402383.4
##  [22] 1302586.3 3154412.6 2560289.7 1720913.8 1020490.4 1015069.1  573834.6
##  [29] 1292821.1  611586.5  692259.1 1291779.9  471001.9  702870.6 1241727.7
##  [36] 4563375.3 2084722.7 2750914.5 1398957.2 1542212.4  644248.4  525701.7
##  [43] 1030481.5 2660553.4 2216854.5 2359747.8  842656.8 1709107.0 3258598.6
##  [50] 3051651.0 1413559.3  773745.6  564526.3  791044.4  831840.2 1009079.1
##  [57] 5859656.2 2448100.7 1375773.5 4259682.9 2444768.3  953425.5  478376.5
##  [64] 1766506.3  989558.9 1724299.6  974077.7  907554.4  688384.4 3994344.5
##  [71] 2169963.7 1081479.9 3079851.2 1529976.8 3360796.7  825319.4  514672.8
##  [78] 2267320.9 1486433.6  959744.1 1931361.5  513803.2 6155151.4 1638433.4
##  [85] 1751122.7  679615.3 4421403.0  874616.3 2405794.2  872783.6  431493.4
##  [92] 1005330.0  609242.5 2930304.8 5694825.5  580211.1 2197063.1  894937.4
##  [99] 3639205.7 2206453.0
```

```
summary(resultados)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  402383  865252 1333604 1699732 2220106 6155151
```

Grafico de la distribucion de los resultados de la estrategia azarosa

```
hist(resultados,
     breaks = 100,
     col = "skyblue",
     border = "white",
     main = "Histograma del saldo final en 30 simulaciones aleatorias",
     xlab = "Saldo final",
     ylab = "Frecuencia")

# Línea del promedio
abline(v = mean(resultados), col = "red", lwd = 2, lty = 2)
legend("topright", legend = paste("Media =", round(mean(resultados), 2)),
       col = "red", lwd = 2, lty = 2, bty = "n")
```



Test de Hipotesis para las estrategias

```
saldo_estrategia <- macddf_sim$saldo[nrow(macddf_sim)]
t.test(resultados, mu = saldo_estrategia, alternative = "less")
```

```
##
```

```
##  One Sample t-test
##
## data:  resultados
## t = -284.41, df = 99, p-value < 2.2e-16
## alternative hypothesis: true mean is less than 36013831
## 95 percent confidence interval:
##      -Inf 1900055
## sample estimates:
## mean of x
##   1699732
```

Aplicandolo a todos los tickers

```
library(dplyr)
library(TTR)  # o library(quantmod), ya que EMA viene de ahí también
library(purrr)

# Asegurarse de que los datos estén ordenados por Ticker y Date
datos.filt <- datos.filt %>%
  arrange(Ticker, Date)

# Función para calcular EMA(50) y EMA(200)
calcular_emas <- function(df) {
  df$EMA_50 <- EMA(df$Close, n = 50)
  df$EMA_200 <- EMA(df$Close, n = 200)
  return(df)
}

# Aplicar la función a cada grupo (ticker)
datos.ema <- datos.filt %>%
  group_by(Ticker) %>%
  group_modify(~calcular_emas(.x)) %>%
  ungroup()

datos.ema
```

Ticker <chr>	Date <date>	Open <dbl>	High <dbl>	Low <dbl>	Close <dbl>	Volume <dbl>	Dividends <dbl>	Stock Splits <dbl>	EMA_50 <dbl>
AA	1962-01-02	0.000000	1.550548	1.541704	1.541704	55930	0.00000	0.0	NA
AA	1962-01-03	1.541703	1.565285	1.538755	1.565285	74906	0.00000	0.0	NA
AA	1962-01-04	1.565285	1.577076	1.565285	1.565285	80899	0.00000	0.0	NA
AA	1962-01-05	1.565284	1.574128	1.559389	1.562337	70911	0.00000	0.0	NA
AA	1962-01-08	1.556442	1.556442	1.497485	1.509277	93883	0.00000	0.0	NA
AA	1962-01-09	1.509278	1.532861	1.503383	1.503383	64919	0.00000	0.0	NA
AA	1962-01-10	1.503382	1.509277	1.497486	1.506329	34956	0.00000	0.0	NA
AA	1962-01-11	1.500434	1.500434	1.491590	1.497486	27965	0.00000	0.0	NA
AA	1962-01-12	1.497487	1.500434	1.462113	1.462113	26966	0.00000	0.0	NA
AA	1962-01-15	1.450321	1.450321	1.432634	1.432634	64919	0.00000	0.0	NA

1-10 of 10,000 rows | 1-10 of 11 columns Previous 1 2 3 4 5 6 ... 1000 Next

```
emadf <- datos.ema %>%
  select(Date, Ticker, Close, EMA_50, EMA_200) %>%
  filter(!is.na(EMA_50), !is.na(EMA_200)) %>%
  arrange(Ticker, Date) %>%
  group_by(Ticker) %>%
  mutate(
    ema50may = EMA_50 > EMA_200,
    cross = ifelse(ema50may != lag(ema50may),
                   ifelse(ema50may == TRUE, 1, -1),
                   0),
    # Agregar NA si el ticker cambia respecto al anterior
    cross = ifelse(Ticker != lag(Ticker), NA, cross)
  ) %>%
  ungroup()

emadf
```

Date <date>	Ticker <chr>	Close <dbl>	EMA_50 <dbl>	EMA_200 <dbl>	ema50may <lgl>	cross <dbl>
1962-10-15	AA	1.275179	1.335335	1.396979	FALSE	NA
1962-10-16	AA	1.272186	1.332858	1.395737	FALSE	0
1962-10-17	AA	1.251232	1.329657	1.394299	FALSE	0
1962-10-18	AA	1.230278	1.325760	1.392667	FALSE	0
1962-10-19	AA	1.209324	1.321194	1.390843	FALSE	0
1962-10-22	AA	1.185377	1.315868	1.388798	FALSE	0
1962-10-23	AA	1.191364	1.310985	1.386834	FALSE	0
1962-10-24	AA	1.245245	1.308407	1.385425	FALSE	0
1962-10-25	AA	1.281165	1.307339	1.384388	FALSE	0
1962-10-26	AA	1.257218	1.305373	1.383122	FALSE	0

1-10 of 10,000 rows Previous 1 2 3 4 5 6 ... 1000 Next

```
simular_estrategia <- function(df) {
  df <- df %>%
    filter(!is.na(Close), !is.na(cross)) %>%
    arrange(Date)

  if (nrow(df) == 0) return(NULL)  # evitar errores

  initial_state <- list(cash = 10000, stocks = 0, saldo = 10000)

  sim_list <- purrr::accumulate(
    1:nrow(df),
    .init = initial_state,
    .f = function(state, i) {
      row <- df[i, ]
      new_state <- state

      if (!is.na(row$cross)) {
        if (row$cross == 1) {
          # Golden Cross: Comprar
          new_state$stocks <- state$cash / row$Close
          new_state$cash <- 0
        } else if (row$cross == -1) {
          # Death Cross: Vender
          new_state$cash <- state$cash + state$stocks * row$Close
          new_state$stocks <- 0
        }
      }

      new_state$saldo <- new_state$cash + new_state$stocks * row$Close
      return(new_state)
    }
  )

  sim_df <- bind_rows(sim_list[-1])  # quitar estado inicial
  df_result <- bind_cols(df, sim_df)
  return(df_result)
}
```

```
# Dividir el dataframe por ticker
tickers_split <- emadf %>%
  group_by(Ticker) %>%
  group_split()

# Aplicar la simulación a cada ticker
resultados_por_ticker <- lapply(tickers_split, simular_estrategia)

# Unir todos los resultados en un único dataframe
resultados_finales <- bind_rows(resultados_por_ticker)
```

```
saldo_final_por_ticker <- resultados_finales %>%
```

```r
  group_by(Ticker) %>%
  filter(!is.na(saldo)) %>%
  slice_tail(n = 1) %>%
  select(Ticker, Date, saldo)

print(saldo_final_por_ticker)
```

```
## # A tibble: 29 × 3
## # Groups:   Ticker [29]
##    Ticker Date           saldo
##    <chr>  <date>         <dbl>
##  1 AA     2024-11-04    39778.
##  2 AEP    2024-11-04   350985.
##  3 BA     2024-11-04 15523962.
##  4 BP     2024-11-04   257503.
##  5 CAT    2024-11-04  1220865.
##  6 CNP    2024-11-04   160527.
##  7 CVX    2024-11-04  6380868.
##  8 DIS    2024-11-04  2182131.
##  9 DTE    2024-11-04  1083483.
## 10 ED     2024-11-04   885418.
## # ℹ 19 more rows
```

```r
simular_estrategia_azar <- function(df, n_senales = 75) {
  df <- df %>%
    filter(!is.na(Close)) %>%
    arrange(Date)

  n <- nrow(df)
  if (n < n_senales) return(NULL)

  # Inicializar columna de azar
  df$azar <- 0

  # Crear señales aleatorias distribuidas
  bloques <- floor(n / n_senales)
  idx_1 <- sapply(0:(n_senales - 1), function(i) {
    bloque_inicio <- i * bloques + 1
    bloque_fin <- min((i + 1) * bloques, n)
    sample(bloque_inicio:bloque_fin, 1)
  })
  df$azar[idx_1] <- 1

  df_ultimas_filas <- df %>%
    slice_tail(n = 1) %>%
    ungroup()

  df <- df %>%
    filter(azar == 1)


  df <- bind_rows(df, df_ultimas_filas) %>%
    distinct(Date, .keep_all = TRUE) %>%
    arrange(Date)

  n <- nrow(df)

  # Simulación
  initial_state <- list(cash = 10000, stocks = 0, saldo = 10000)

  sim_list <- purrr::accumulate(
    1:n,
    .init = initial_state,
    .f = function(state, i) {
      row <- df[i, ]
      new_state <- state

      if (state$stocks == 0) {
        new_state$stocks <- state$cash / row$Close
        new_state$cash <- 0
      } else {
        new_state$cash <- state$cash + state$stocks * row$Close
        new_state$stocks <- 0
      }

      new_state$saldo <- new_state$cash + new_state$stocks * row$Close
      return(new_state)
    }
  )

  sim_df <- bind_rows(sim_list[-1])
  df_result <- bind_cols(df, sim_df)
  return(df_result)
}
```

```r
library(progress)

n_repeticiones <- 100
lista_saldos_finales <- list()

# Crear barra de progreso
pb <- progress_bar$new(
  format = "[:bar] :percent | Iteración :current/:total | Tiempo restante: :eta",
  total = n_repeticiones,
  clear = FALSE,
  width = 60
)

for (i in 1:n_repeticiones) {
  pb$tick()  # Actualizar barra

  # Simular estrategia para todos los tickers
  resultados_iteracion <- lapply(tickers_split, simular_estrategia_azar) %>%
    bind_rows()

  # Extraer saldos finales
  lista_saldos_finales[[i]] <- resultados_iteracion %>%
    group_by(Ticker) %>%
    filter(!is.na(saldo)) %>%
    slice_tail(n = 1) %>%
    select(Ticker, Date, saldo)

}

resultados_saldos_finales_azar <- bind_rows(lista_saldos_finales)
resultados_saldos_finales_azar
```

Ticker <chr>	Date <date>	saldo <dbl>
AA	2024-11-04	30501.859
AEP	2024-11-04	151449.184
BA	2024-11-04	180492.461
BP	2024-11-04	67470.796
CAT	2024-11-04	182522.581
CNP	2024-11-04	250955.506
CVX	2024-11-04	791013.650
DIS	2024-11-04	204242.766
DTE	2024-11-04	302753.923
ED	2024-11-04	162911.978

1-10 of 2,900 rows   Previous 1 2 3 4 5 6 ... 290 Next

Test de Hipotesis Estrategias

```r
# Paso 1: Calcular el promedio de cada ticker en las simulaciones
resultado_promedios <- resultados_saldos_finales_azar %>%
  group_by(Ticker) %>%
  summarise(saldo_azar = mean(saldo))

# Paso 2: Unir con el saldo de la estrategia
comparacion <- saldo_final_por_ticker %>%
  select(Ticker, saldo_estrategia = saldo) %>%
  inner_join(resultado_promedios, by = "Ticker")

# Paso 3: Test t pareado (una cola)
t.test(comparacion$saldo_estrategia, comparacion$saldo_azar,
       paired = TRUE,
       alternative = "greater")
```

```
##
##  Paired t-test
##
## data:  comparacion$saldo_estrategia and comparacion$saldo_azar
## t = 2.522, df = 28, p-value = 0.008817
## alternative hypothesis: true mean difference is greater than 0
## 95 percent confidence interval:
##  923637.5      Inf
## sample estimates:
## mean difference
##         2837656
```

Optimizacion de EMAs

```
calcular_saldo_final_ema <- function(df=datos.filt, ema1, ema2) {

  calcular_emas <- function(df, ema1, ema2) {
    df %>%
      mutate(
        EMA1 = EMA(Close, n = ema1),
        EMA2 = EMA(Close, n = ema2)
      )
  }

  # Uso:
  df <- df %>%
    group_by(Ticker) %>%
    calcular_emas(ema1, ema2) %>%
    ungroup()

  df <- df %>%
    select(Date, Ticker, Close, EMA1, EMA2) %>%
    filter(!is.na(EMA1), !is.na(EMA2)) %>%
    arrange(Ticker, Date) %>%
    group_by(Ticker) %>%
    mutate(
      ema1may = EMA1 > EMA2,
      cross = ifelse(ema1may != lag(ema1may),
                     ifelse(ema1may == TRUE, 1, -1),
                     0),
      # Agregar NA si el ticker cambia respecto al anterior
      cross = ifelse(Ticker != lag(Ticker), NA, cross)
    ) %>%
    ungroup()

  df_ultimas_filas <- df %>%
    group_by(Ticker) %>%
    slice_tail(n = 1) %>%
    ungroup()

  df <- df %>%
    filter(cross != 0)

  df <- bind_rows(df, df_ultimas_filas) %>%
    distinct(Ticker, Date, .keep_all = TRUE) %>%
    arrange(Ticker, Date)

  df_split <- df %>%
    group_by(Ticker) %>%
    group_split()

  df_result <- lapply(df_split, simular_estrategia) %>%
    bind_rows() %>%
    group_by(Ticker) %>%
    filter(!is.na(saldo)) %>%
    slice_tail(n = 1)

  return(mean(df_result$saldo))

}
```

```
library(progress)
library(reshape2)
```

```
##
## Adjuntando el paquete: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
saldo_base <- mean(saldo_final_por_ticker$saldo)

ema1 <- seq(10, 250, by = 10)
ema2 <- seq(150, 400, by = 10)

# Crear matriz vacía
matriz_resultados <- matrix(
  nrow = length(ema1),
  ncol = length(ema2),
  dimnames = list(paste0("EMA1_", ema1), paste0("EMA2_", ema2))
)

pb <- progress_bar$new(
  format = "[:bar] :percent | Fila: :current/:total | Tiempo: :elapsedfull",
  total = length(ema1) * length(ema2),  # Total de iteraciones
  clear = FALSE,
  width = 60
)

# Llenar la matriz con mapply
for (i in seq_along(ema1)) {
  for (j in seq_along(ema2)) {
    if (ema1[i] >= ema2[j]) {
      matriz_resultados[i, j] <- NA
    } else {
      matriz_resultados[i, j] <- calcular_saldo_final_ema(datos.filt, ema1[i], ema2[j])/saldo_base
    }


    pb$tick()
  }
}
```

```
library(plotly)
```

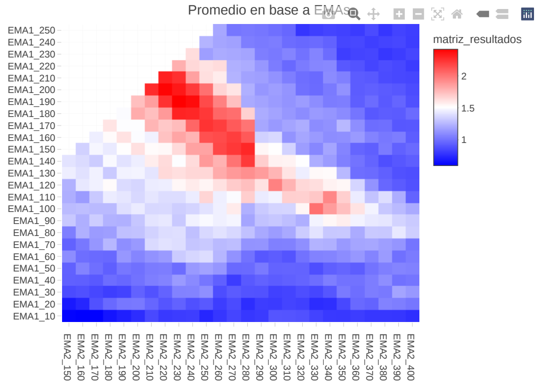
```
##
## Adjuntando el paquete: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
plot_ly(
  z = ~matriz_resultados,
  x = colnames(matriz_resultados),
  y = rownames(matriz_resultados),
  colorscale = list(
    c(0, "blue"),     # Mínimo: azul
    c(0.5, "white"), # Medio: blanco (ajusta el 0.5 según tus datos)
    c(1, "red")       # Máximo: rojo
  ),
  type = "heatmap"
) %>%
  layout(title = "Promedio en base a EMAs")
```

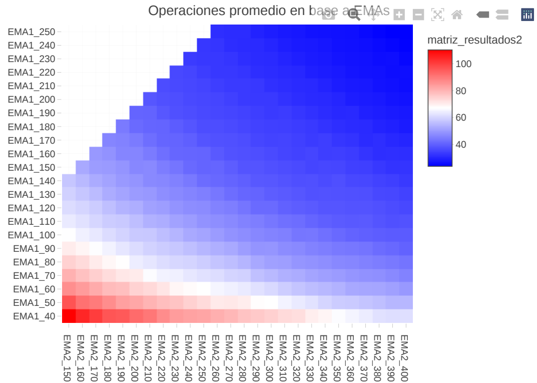


Matriz de cantidad de operaciones

```
calcular_operaciones_ema <- function(df=datos.filt, ema1, ema2) {

  calcular_emas <- function(df, ema1, ema2) {
    df %>%
      mutate(
        EMA1 = EMA(Close, n = ema1),
        EMA2 = EMA(Close, n = ema2)
      )
  }

  # Uso:
  df <- df %>%
    group_by(Ticker) %>%
    calcular_emas(ema1, ema2) %>%
    ungroup()

  df <- df %>%
    select(Date, Ticker, Close, EMA1, EMA2) %>%
    filter(!is.na(EMA1), !is.na(EMA2)) %>%
    arrange(Ticker, Date) %>%
    group_by(Ticker) %>%
    mutate(
      ema1may = EMA1 > EMA2,
      cross = ifelse(ema1may != lag(ema1may),
                     ifelse(ema1may == TRUE, 1, -1),
                     0),
      # Agregar NA si el ticker cambia respecto al anterior
      cross = ifelse(Ticker != lag(Ticker), NA, cross)
    ) %>%
    ungroup()

  df_ultimas_filas <- df %>%
    group_by(Ticker) %>%
    slice_tail(n = 1) %>%
    ungroup()

  df <- df %>%
    filter(cross != 0)

  return(nrow(df)/29)
}
```

```
ema1 <- seq(40, 250, by = 10)
ema2 <- seq(150, 400, by = 10)

# Crear matriz vacía
matriz_resultados2 <- matrix(
  nrow = length(ema1),
  ncol = length(ema2),
  dimnames = list(paste0("EMA1_", ema1), paste0("EMA2_", ema2))
)

pb <- progress_bar$new(
  format = "[:bar] :percent | Fila: :current/:total | Tiempo: :elapsedfull",
  total = length(ema1) * length(ema2),  # Total de iteraciones
  clear = FALSE,
  width = 60
)

# Llenar la matriz con mapply
for (i in seq_along(ema1)) {
  for (j in seq_along(ema2)) {
    if (ema1[i] >= ema2[j]) {
      matriz_resultados2[i, j] <- NA
    } else {
      matriz_resultados2[i, j] <- calcular_operaciones_ema(datos.filt, ema1[i], ema2[j])
    }

    pb$tick()
  }
}
```

```
plot_ly(
  z = ~matriz_resultados2,
  x = colnames(matriz_resultados2),
  y = rownames(matriz_resultados2),
  colorscale = list(
    c(0, "blue"),     # Mínimo: azul
    c(0.5, "white"), # Medio: blanco (ajusta el 0.5 según tus datos)
    c(1, "red")       # Máximo: rojo
  ),
  type = "heatmap"
) %>%
  layout(title = "Operaciones promedio en base a EMAs")
```



Conclusion

Los resultados del análisis sugieren que, en términos estadísticos, los patrones Doji no representan un indicador confiable para anticipar cambios significativos en la tendencia del precio de las acciones. Las pruebas de hipótesis realizadas no mostraron diferencias consistentes en los rendimientos posteriores a la aparición de estos patrones, lo que indica que su valor predictivo es limitado cuando se consideran de forma aislada.

En contraste, las estrategias basadas en cruces de medias móviles —particularmente el “golden cross” y el “death cross”— mostraron mayor efectividad, con diferencias estadísticamente significativas en los rendimientos observados tras la señal. Esto refuerza la idea de que los cruces de promedios móviles pueden ser herramientas más robustas dentro del análisis técnico, al ofrecer señales que reflejan mejor la dinámica de largo plazo del mercado.

En resumen, mientras que los Dojis pueden tener valor como señales complementarias dentro de un contexto más amplio, los cruces de medias móviles demostraron ser indicadores más consistentes y útiles para tomar decisiones informadas en el análisis de acciones.