R to explore data ; plots; computation

A.Prices ; returns; log-returns ; Explore stock

Quantmod package

FIG 2: Plot of Close

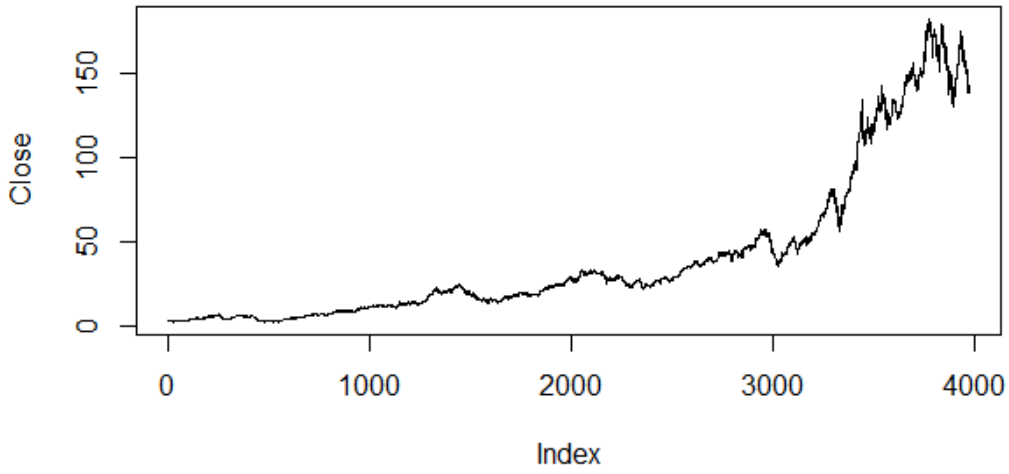
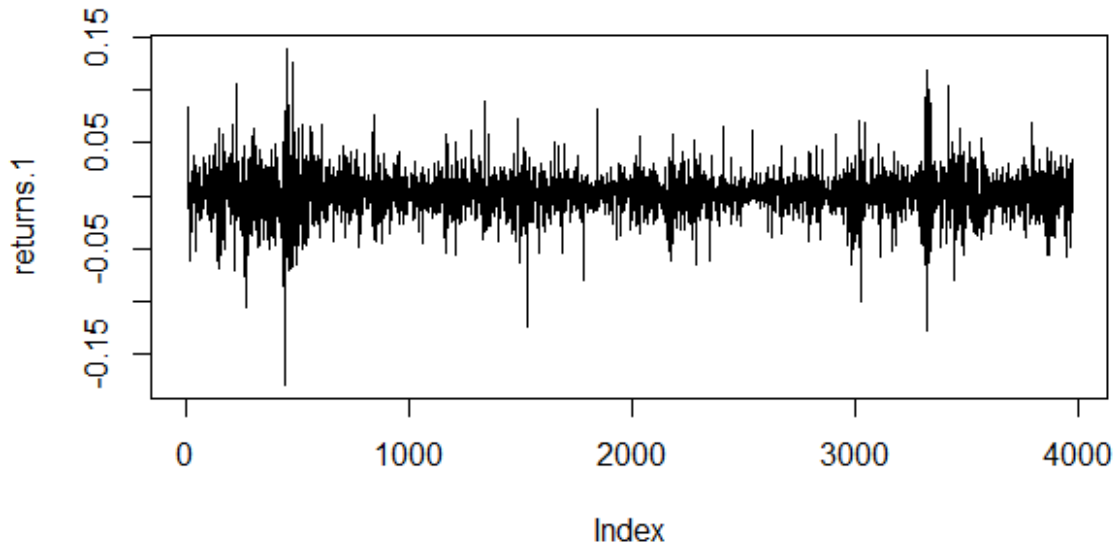


FIG 2: Plot of returns as a vector/matrix [from Delt()]



##########DOCUMENTATION

Delt {quantmod} R Documentation

Calculate Percent Change

Description

Calculate the k-period percent difference within one series, or between two series. Primarily used to calculate the percent change from one period to another of a given series, or to calculate the percent difference between two series over the full series.

Usage

Delt(x1, x2 = NULL, k = 0, type = c("arithmetic", "log"))

Arguments

x1

m x 1 vector

x2

m x 1 vector

k

change over k-periods. default k=1 when x2 is NULL.

type

type of difference. log or arithmetic (default).

Details

When called with only x1, the one period percent change of the series is returned by default. Internally this happens by copying x1 to x2. A two period difference would be specified with k=2.

################################DOCUMENATION

periodReturn {quantmod} R Documentation
Calculate Periodic Returns
Description
Given a set of prices, return periodic returns.

Usage
```
periodReturn(x,
          period='monthly',
          subset=NULL,
          type='arithmetic',
          leading=TRUE,
          ...)

dailyReturn(x, subset=NULL, type='arithmetic',
        leading=TRUE, ...)
weeklyReturn(x, subset=NULL, type='arithmetic',
        leading=TRUE, ...)
monthlyReturn(x, subset=NULL, type='arithmetic',
        leading=TRUE, ...)
quarterlyReturn(x, subset=NULL, type='arithmetic',
        leading=TRUE, ...)
annualReturn(x, subset=NULL, type='arithmetic',
        leading=TRUE, ...)
yearlyReturn(x, subset=NULL, type='arithmetic',
        leading=TRUE, ...)
allReturns(x, subset=NULL, type='arithmetic',
        leading=TRUE)
```
Arguments
x
object of state prices, or an OHLC type object

period
character string indicating time period. Valid entries are ‘daily’, ‘weekly’, ‘monthly’, ‘quarterly’, ‘yearly’. All are accessible from wrapper functions described below. Defaults to monthly returns (same as monthlyReturn)

subset
an xts/ISO8601 style subset string

type
type of returns: arithmetic (discrete) or log (continuous)

leading
should incomplete leading period returns be returned

...
passed along to to.period

Details
periodReturn is the underlying function for wrappers:

allReturns: calculate all available return periods

dailyReturn: calculate daily returns

weeklyReturn: calculate weekly returns

monthlyReturn: calculate monthly returns

quarterlyReturn: calculate quarterly returns

annualReturn: calculate annual returns

Value
Returns object of the class that was originally passed in, with the possible exception of monthly and quarterly return indicies being changed to class yearmon and yearqtr where available. This can be overridden with the indexAt argument passed in the ... to the to.period function.

By default, if subset is NULL, the full dataset will be used.

Note
Attempts are made to re-convert the resultant series to its original class, if supported by the xts package. At present, objects inheriting from the 'ts' class are returned as xts objects. This is to make the results more visually appealling and informative. All xts objects can be converted to class ts with as.ts if that is desirable.

The first and final row of returned object will have the period return to last date, i.e. this week/month/quarter/year return to date even if the start/end is not the start/end of the period. Leading period calculations can be suppressed by setting leading=FALSE.

Author(s)
Jeffrey A. Ryan

See Also
getSymbols

Examples
```
## Not run:
getSymbols('QQQQ',src='yahoo')
allReturns(QQQQ)  # returns all periods

periodReturn(QQQQ,period='yearly',subset='2003::')  # returns years 2003 to present
periodReturn(QQQQ,period='yearly',subset='2003')  # returns year 2003
```

xts- package;

FIGURE 3: plot returns, use xts -object

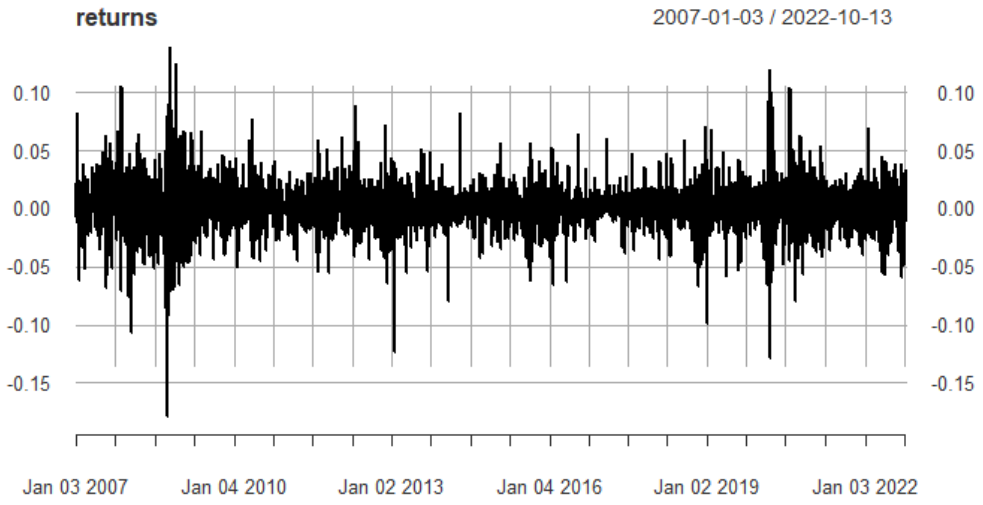
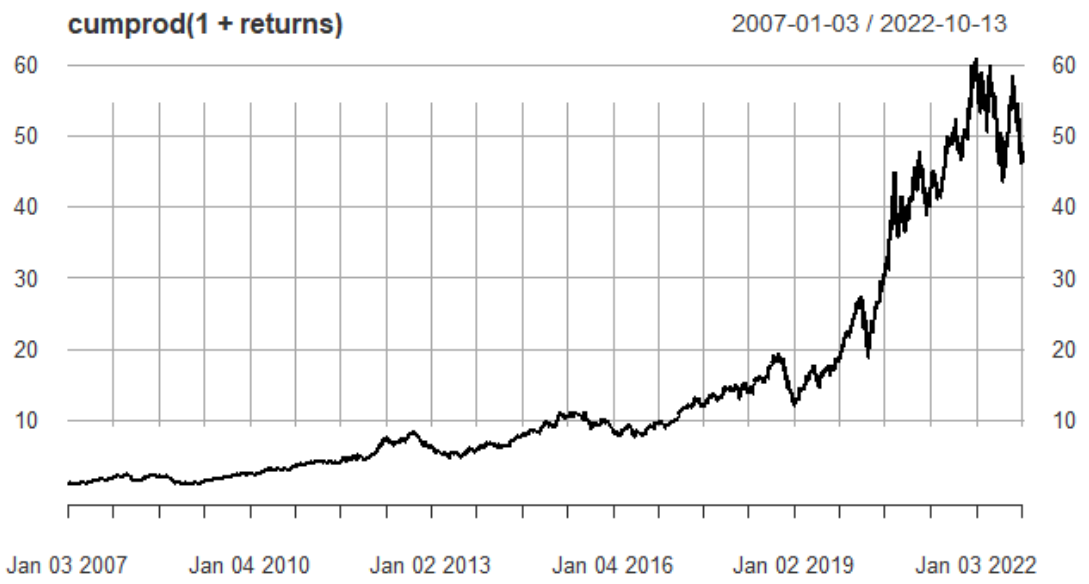


Fig 4: cumprod() of returns

> plot(cumprod(1+returns), type="l")



##FIG 5: log-return; Sum of log-returns

```
> logreturn.AAPL <- log(1+ returns)
> plot(cumprod(1+returns), type="l")
> lines(cumsum(logreturn.AAPL), col="red")
```

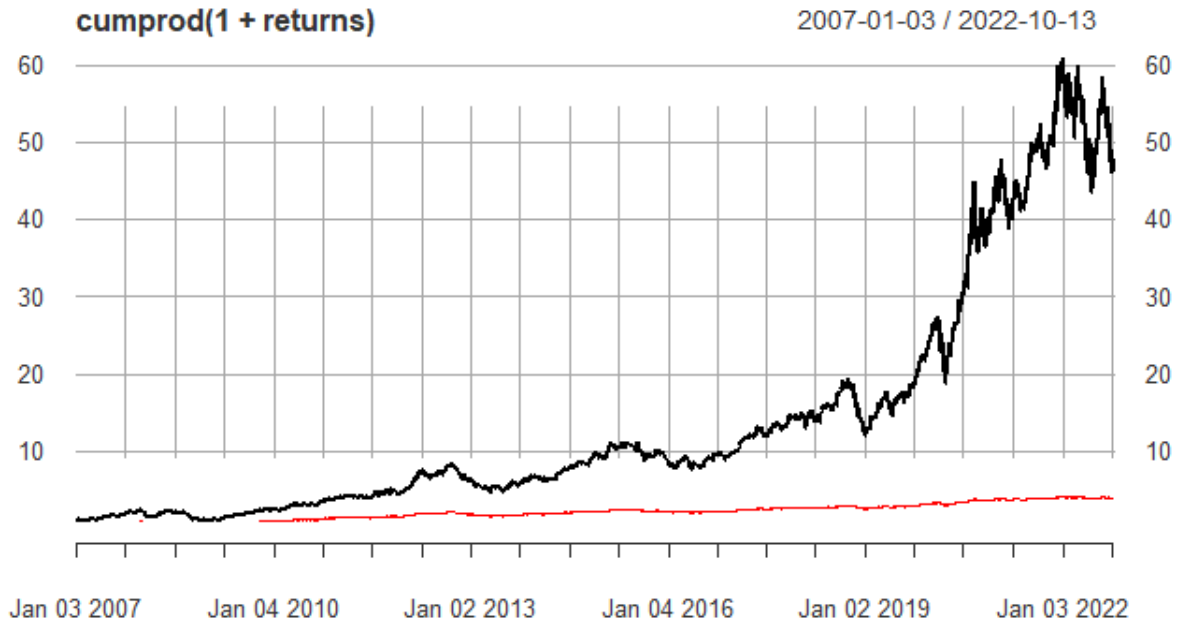
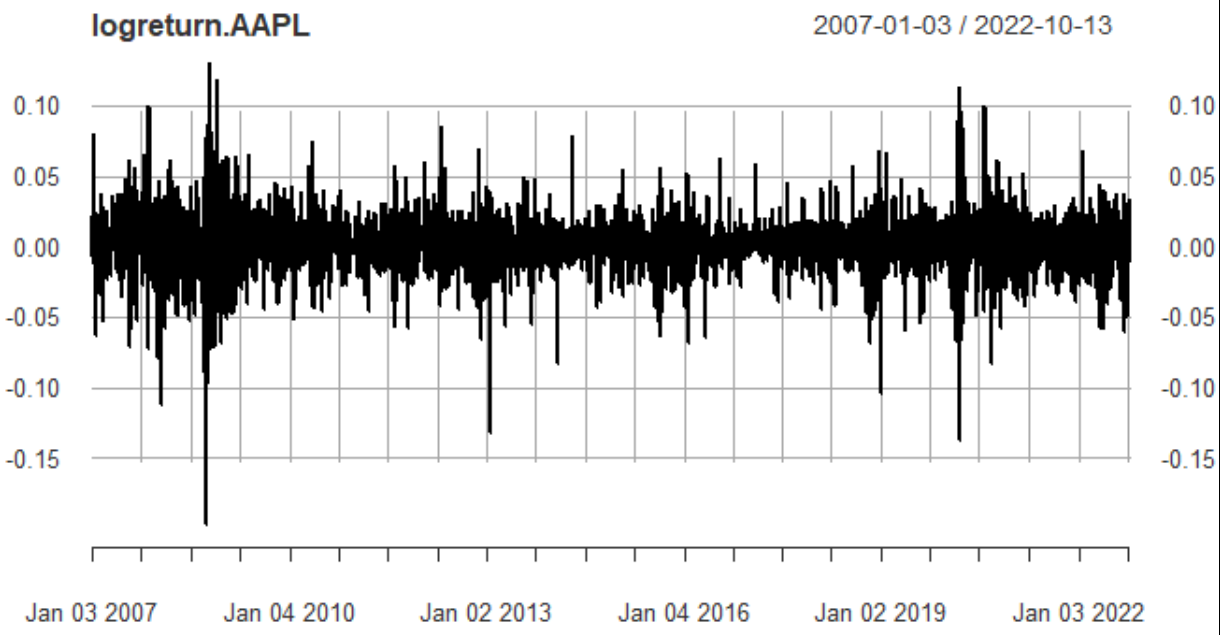


FIG 5: NOTE: log-return has negative vales



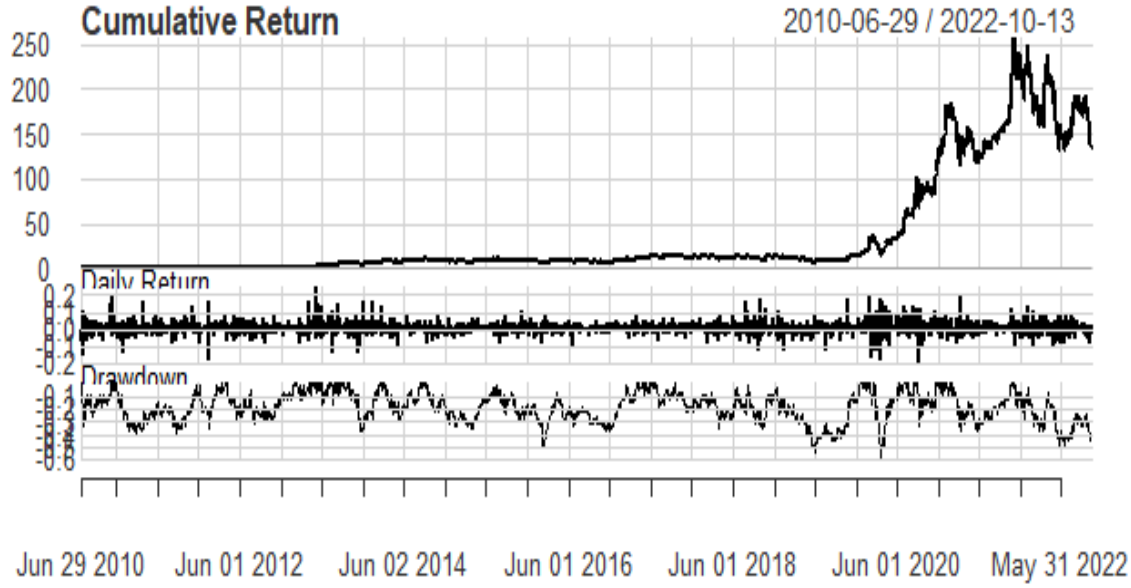
NOTE: As return is small (close to 0), returns and log-returns are almost equal.
ALMOST AS FIGURE 3

PerformAnalytics package

```
> library(PerformanceAnalytics)
## Return:  dailyReturn()
TSLA.Return<- dailyReturn(Cl(as.xts(TSLA)))
charts.PerformanceSummary(TSLA.Return, main="TESLA performance")


> returns.TSLA<- dailyReturn(Cl(as.xts(TSLA)))
> plot(cumprod(1+returns.TSLA), col="red")
```

TESLA performance



B. Portfolio; Portfolio returns

B1. Calculate Portfolio Expected Return and Risk

```
############  Calculate Portfolio expected return and risk;
## Suppose weights of  AAPL and TSLA   is   0.7,  0.7

## portfolio mean

> wts<- c(.3,.7)
> means.ret <- c( mean(combine[  ,3]),  mean(combine[ , 4]))
> port.mean <- t(wts)%*% means.ret
> port.mean
          [,1]
[1,] 0.001869691


## portfolio risk  : use covariance matrix of returns

> covariance<- cov( as.matrix(cbind(combine[  , 3], combine[  , 4])))    #note  combine is xts-
object



> class(combine)
[1] "xts" "zoo"
> port.risk <- t(wts) %*% covariance  %*% wts
> port.risk
           [,1]
[1,] 0.0007498948




################################  MARKOWITZ THEORY

port.ret <- c()
for (i in 1:41){


  port.ret[i] <- wts[i, ] %*% means.ret

}



port.risk <- c()
for (i in 1:41) {

  port.risk[i]<- t(wts[i,  ])  %*% covariance %*%  wts[i,   ]
```

```
}
head(port.risk)


> plot(port.risk, port.ret, pch=16, type="b")
```