TP3 - Analyse financière

Sonia Tieo

29/10/2020

Installation de la librairie quantmode

```
install.packages("quantmod")
```

```
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
```

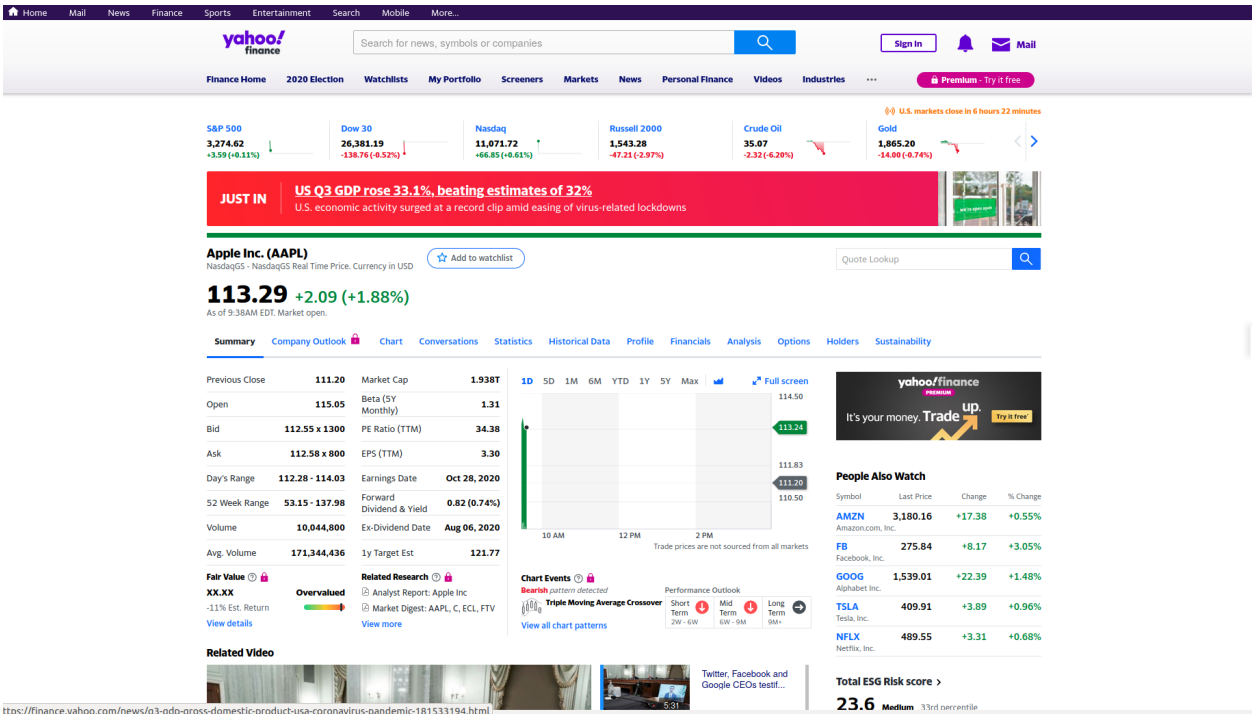
```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
## Loading required package: TTR
```

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

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

Téléchargement des données financières à partir de YAHOO



Q: Inspecter la fonction getSymbols avec *help()* et inspecter le dataframe crée *df*, commentez les colonnes:

```
df <- data.frame(getSymbols("AAPL", auto.assign = F))
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
head(df)
```

```
##            AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
## 2007-01-03  3.081786  3.092143 2.925000   2.992857  1238319600      2.586245
## 2007-01-04  3.001786  3.069643 2.993572   3.059286   847260400      2.643649
## 2007-01-05  3.063214  3.078571 3.014286   3.037500   834741600      2.624823
## 2007-01-08  3.070000  3.090357 3.045714   3.052500   797106800      2.637785
## 2007-01-09  3.087500  3.320714 3.041071   3.306072  3349298400      2.856907
## 2007-01-10  3.383929  3.492857 3.337500   3.464286  2952880000      2.993625
```

```
tail(df)
```

```
##            AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
## 2020-10-29    112.37    116.93   112.20      115.32   146129200        115.32
## 2020-10-30    111.06    111.99   107.72      108.86   190272600        108.86
## 2020-11-02    109.11    110.68   107.32      108.77   122866900        108.77
## 2020-11-03    109.66    111.49   108.73      110.44   107624400        110.44
## 2020-11-04    114.14    115.59   112.35      114.95   138235500        114.95
```

```
## 2020-11-05    117.95    119.62   116.87     119.03   125734400        119.03
```

Q. Rappel: pour s'échauffer avec les dataframes:
1. Sélectionner les 3ères lignes.
2. Sélectionner toutes les lignes et les colonnes 1 et 4.
3.Sélectionner toutes les colonnes pour la ligne "2015-12-31"
4. Quels sont les noms des colonnes ?
5. Renommer les colonnes comme ceci: Open, High, Low, Close, Volume, Adjusted
6.Renommer df par AAPL

```
df[1:3,]
```

```
##            AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
## 2007-01-03  3.081786  3.092143 2.925000   2.992857  1238319600      2.586245
## 2007-01-04  3.001786  3.069643 2.993572   3.059286   847260400      2.643649
## 2007-01-05  3.063214  3.078571 3.014286   3.037500   834741600      2.624823
```

```
#df[,c(1,4)]
df["2015-12-31" ,]
```

```
##            AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
## 2015-12-31   26.7525   26.7575   26.205     26.315   163649200      24.42216
```

```
colnames(df)
```

```
## [1] "AAPL.Open"     "AAPL.High"     "AAPL.Low"      "AAPL.Close"
## [5] "AAPL.Volume"   "AAPL.Adjusted"
```

```
colnames(df) <- c("Open", "High", "Low", "Close", "Volume", "Adjusted")
AAPL <- df
```

Q. A l'aide de la fonction *Cl()* fournie par *quantmod* , extraire le prix de fermeture de toute la période. Puis réprésenter le avec la fonction *plot()* en ajoutant l'argument *type='l'*

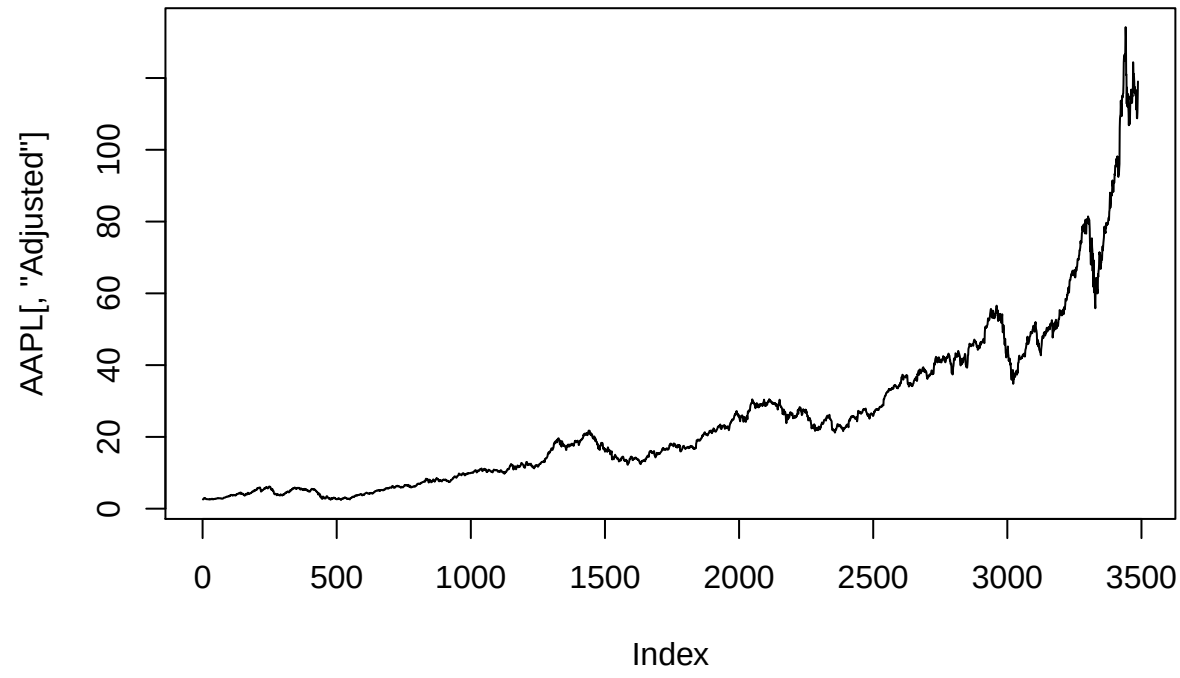
```
plot(Cl(AAPL) , type = 'l')
```

Q. Faire la même chose avec les prix ajustés (Adjusted prices) avec 2 méthodes différentes. 1. Trouver la

fonction qui permet de récupérer les prix ajustés.
2 Le faire en selectionnant la colonne

```
plot(Ad(AAPL) , type = 'l')
```


```
plot(AAPL[,"Adjusted"] , type = 'l')
```



Q. Combien y a t'il de valeurs de "prix ajustés" ?

```
length(Ad(AAPL))
```

```
## [1] 3487
```

Q. Calculer le rendement journalier sur les prix ajustés (en pourcentage) entre :
-"Jour 1" et le "Jour 2"
-"Jour 2" et le "Jour 3"

```
#J1 et J2
100*((Ad(AAPL)[2] - Ad(AAPL)[1])/ Ad(AAPL)[1])
```

```
## [1] 2.219589
```

```
#J2 et J3
100*((Ad(AAPL)[3] - Ad(AAPL)[2])/ Ad(AAPL)[3])
```

```
## [1] -0.7172293
```

Q. Faire une boucle, pour récupérer les rendements journaliers (en pourcentage) des 10èrs jours:

```
AAPL <- data.frame(getSymbols(Symbols = "AAPL", auto.assign = F))
for (jour in 1:10){
  print(jour)
  res <-100*((Ad(AAPL)[jour+1] - Ad(AAPL)[jour])/ Ad(AAPL)[jour])
  print(res)
}
```

```
## [1] 1
## [1] 2.219589
## [1] 2
## [1] -0.7121218
## [1] 3
## [1] 0.4938238
## [1] 4
## [1] 8.307045
## [1] 5
## [1] 4.785525
## [1] 6
## [1] -1.237129
## [1] 7
## [1] -1.231723
## [1] 8
## [1] 2.621009
## [1] 9
## [1] -2.214228
## [1] 10
## [1] -6.192727
```

Q: Heureusement que dans la librarire *quantmod* on peut obtenir un rendement sur la période choisie (journalier, hebdomadaire, mensuel, annuel...). Commentez:

```
#dailyReturn(Ad(as.xts(AAPL)))
#daily return ne s'applique par sur un dataframe mais uniquement sur les objets type "xts"
head(dailyReturn(Ad(as.xts(AAPL))))
```

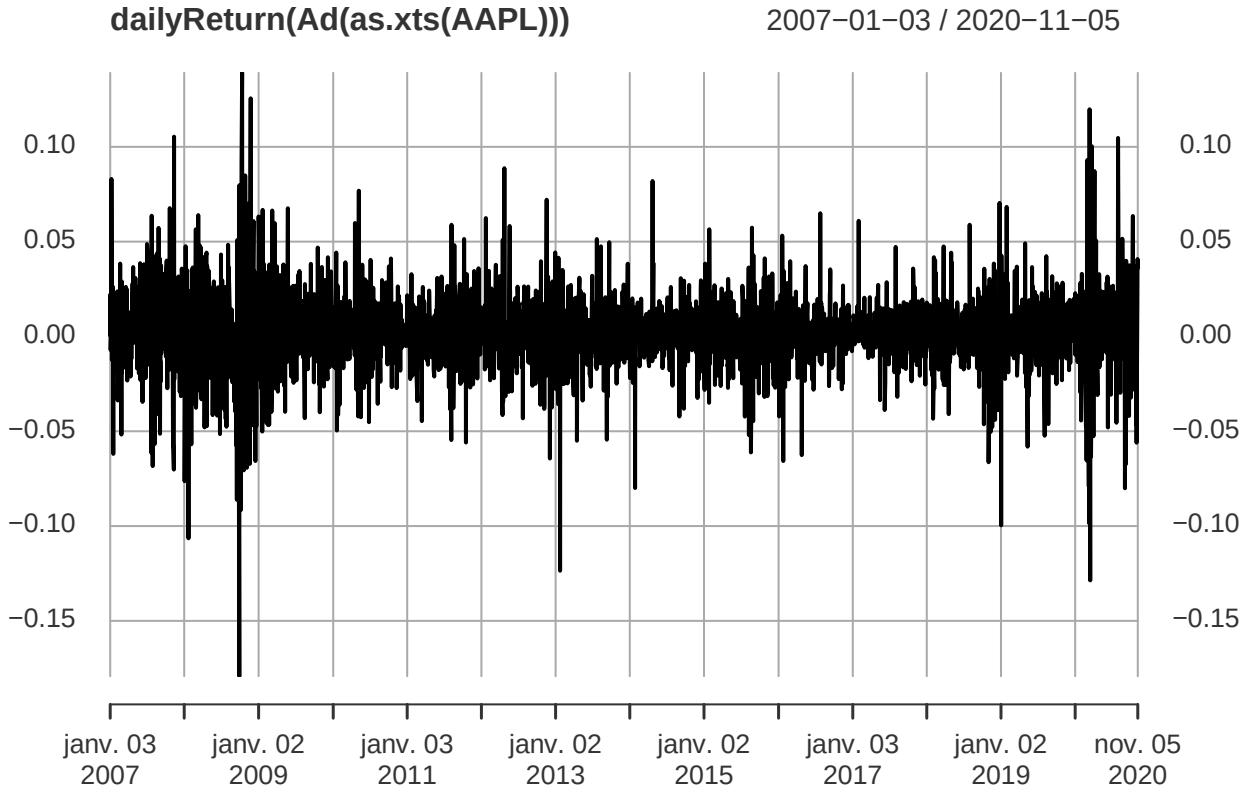
```
##            daily.returns
## 2007-01-03   0.000000000
## 2007-01-04   0.022195886
## 2007-01-05  -0.007121218
## 2007-01-08   0.004938238
## 2007-01-09   0.083070455
## 2007-01-10   0.047855250
```

```
AAPL2 <- getSymbols("AAPL", auto.assign = F)
head(dailyReturn(Ad(AAPL2)))
```

```
##            daily.returns
## 2007-01-03   0.000000000
## 2007-01-04   0.022195886
## 2007-01-05  -0.007121218
## 2007-01-08   0.004938238
## 2007-01-09   0.083070455
## 2007-01-10   0.047855250
```

Q: Faîtes un plot des rendements

```
plot(dailyReturn(Ad(as.xts(AAPL))) , type ="l")
```

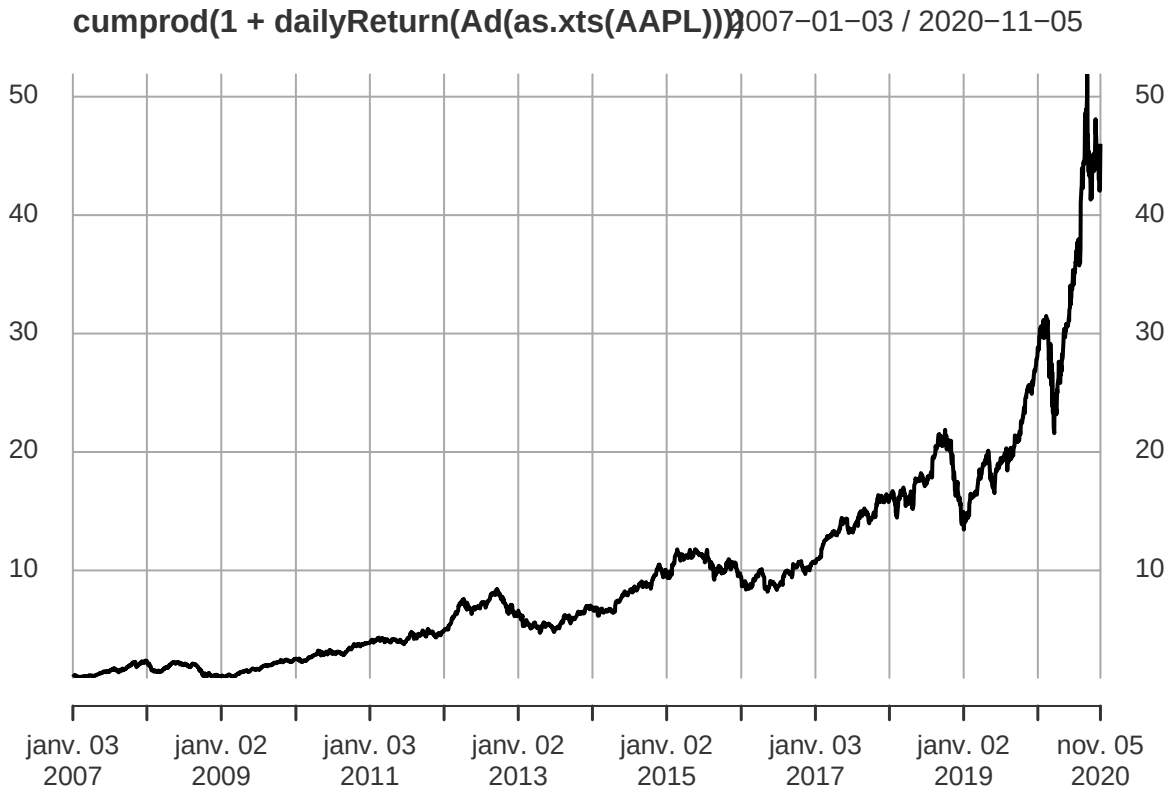


Q: Éxecutez et Commentez:

```
head(cumprod(1+dailyReturn(Ad(as.xts(AAPL)))))
```

```
##            daily.returns
## 2007-01-03      1.000000
## 2007-01-04      1.022196
## 2007-01-05      1.014917
## 2007-01-08      1.019929
## 2007-01-09      1.104654
## 2007-01-10      1.157518
```

```
plot(cumprod(1+dailyReturn(Ad(as.xts(AAPL)))) , type ="l")
```



Q: Faire une boucle, pour récupérer les log rendements journaliers des 10èrs jours:

LOG(J+1/J)

```
for(i in 1:10) {
  print(log(Ad(AAPL)[i+1]/Ad(AAPL)[i]))
  }
```

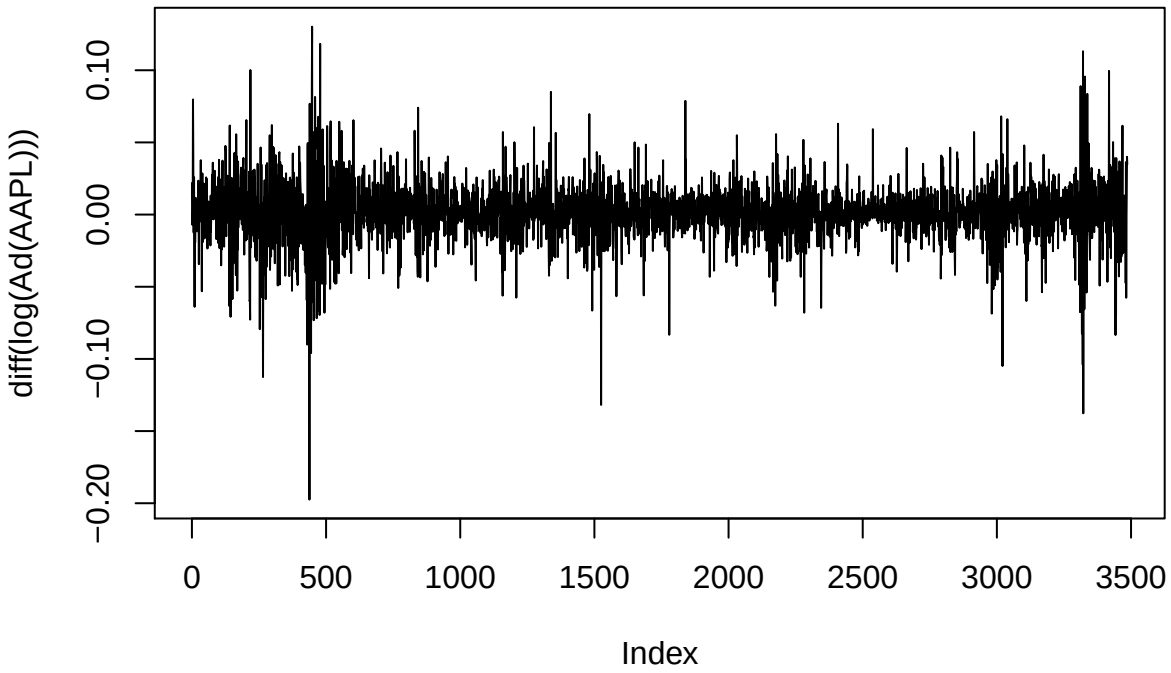
```
## [1] 0.02195314
## [1] -0.007146695
## [1] 0.004926085
## [1] 0.07980002
## [1] 0.04674546
## [1] -0.01244845
## [1] -0.01239372
## [1] 0.02587249
## [1] -0.0223911
## [1] -0.06392779
```

Q: Trouver une alternative (indice: *diff*).
Représentez le graphiquement

```
#diff(log(Ad(AAPL))) #solution
head(diff(log(Ad(AAPL)))) #pour afficher le début
```

```
## [1]  0.021953143 -0.007146695  0.004926085  0.079800021  0.046745457
## [6] -0.012448450
```

```
plot(diff(log(Ad(AAPL))),type="l")
```



Q: Autre alternative, découvrons la fonction *apply*. Au lieu de faire une bouclé, on peut "vectoriser" le travail ainsi:

```
#Exemple, on applique sur les colonnes "close" et "adjusted"
#la fonction logarithme sur les lignes ( margin=2 pour appliuer la fonction sur les lignes)
AAPL <- getSymbols("AAPL", auto.assign = F)
colnames(AAPL) <- c("Open", "High", "Low", "Close", "Volume", "Adjusted")

head(apply(AAPL[,c("Close","Adjusted")] , MARGIN = 2 , log))
```

```
##               Close  Adjusted
## 2007-01-03 1.096228 0.9502070
## 2007-01-04 1.118182 0.9721602
## 2007-01-05 1.111035 0.9650135
## 2007-01-08 1.115961 0.9699395
## 2007-01-09 1.195761 1.0497396
## 2007-01-10 1.242507 1.0964850
```

```
#ON applique par dessus en couche, la fonction diff:
log_ad_cl <- apply(AAPL[,c("Close","Adjusted")] , MARGIN = 2 , log)
head(apply(log_ad_cl ,2, diff))
```

```
##                   Close     Adjusted
## 2007-01-04  0.021953106  0.021953143
## 2007-01-05 -0.007146747 -0.007146695
## 2007-01-08  0.004926118  0.004926085
## 2007-01-09  0.079799851  0.079800021
## 2007-01-10  0.046745773  0.046745457
## 2007-01-11 -0.012448251 -0.012448450
```

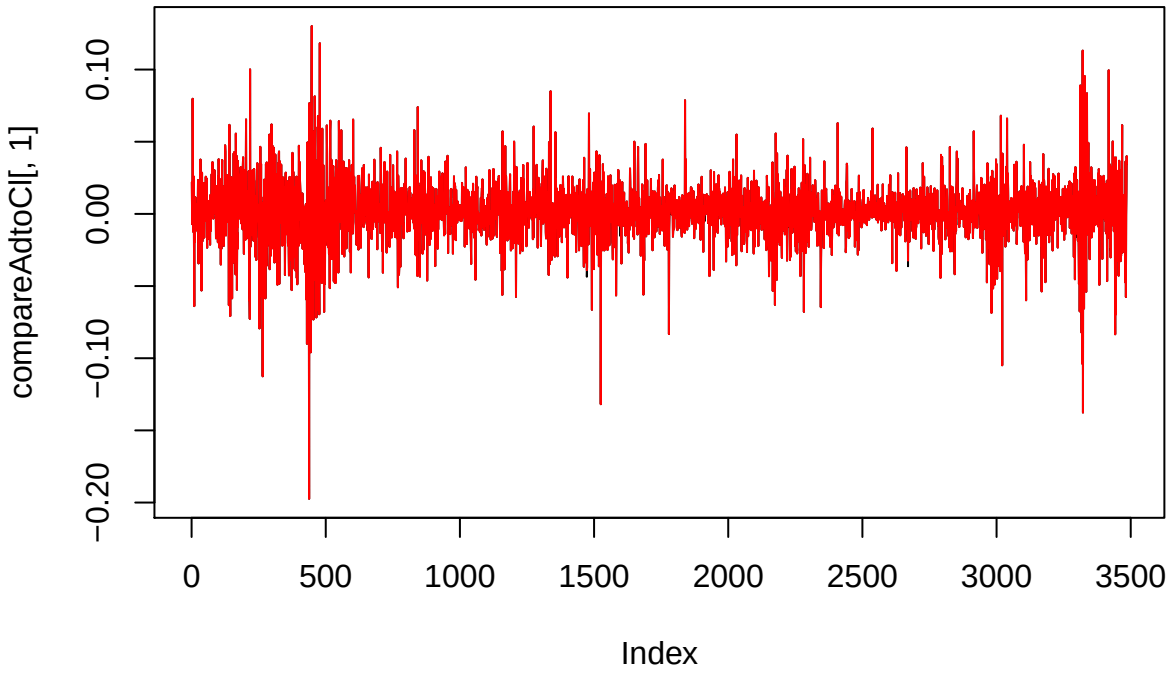
```
#ou bien
head(apply(apply(AAPL[,c("Close","Adjusted")] , MARGIN = 2 , log) ,2, diff))
```

```
##                   Close     Adjusted
## 2007-01-04  0.021953106  0.021953143
```

```
## 2007-01-05 -0.007146747 -0.007146695
## 2007-01-08  0.004926118  0.004926085
## 2007-01-09  0.079799851  0.079800021
## 2007-01-10  0.046745773  0.046745457
## 2007-01-11 -0.012448251 -0.012448450
```

```
#Stockons le dans une variable pour pouvoir l'utiliser
compareAdtoCl <-apply(apply(AAPL[,c("Close","Adjusted")] , MARGIN = 2 , log),2, diff)

#Commentez
plot(compareAdtoCl[,1] , type = "l")
lines(compareAdtoCl[,2] , type = "l" , col = "red")
```

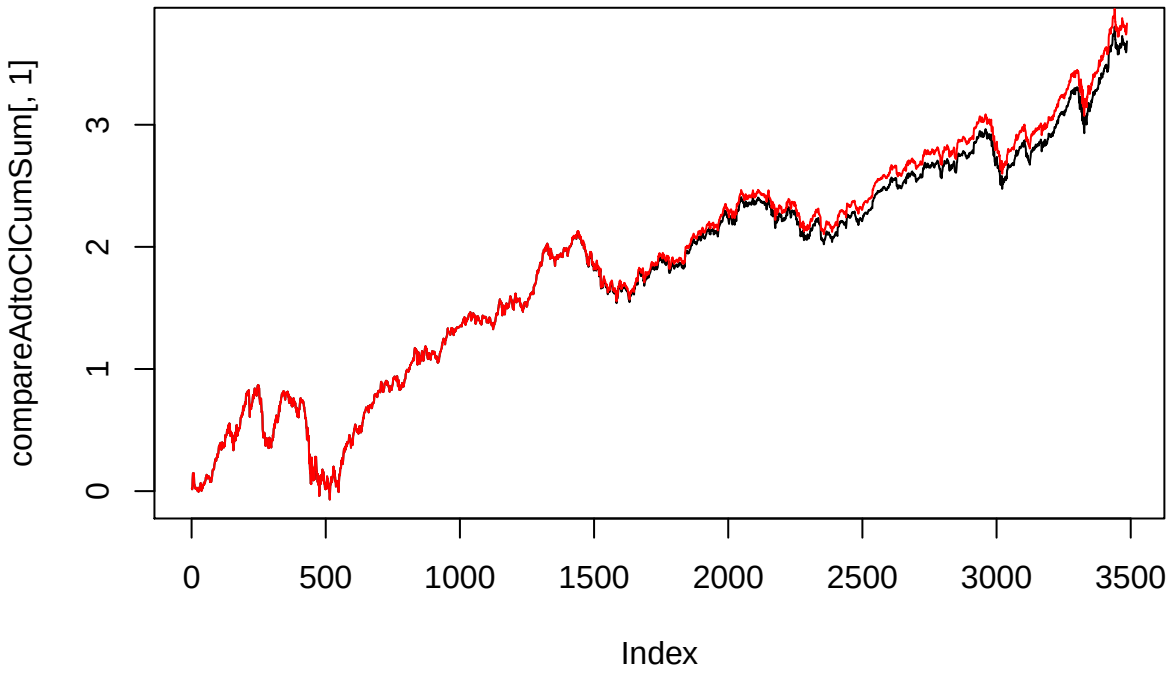


On va plutôt regarder les sommes cumulées:

```
compareAdtoClCumSum <- data.frame(apply(compareAdtoCl,2,cumsum))
head(compareAdtoClCumSum)
```

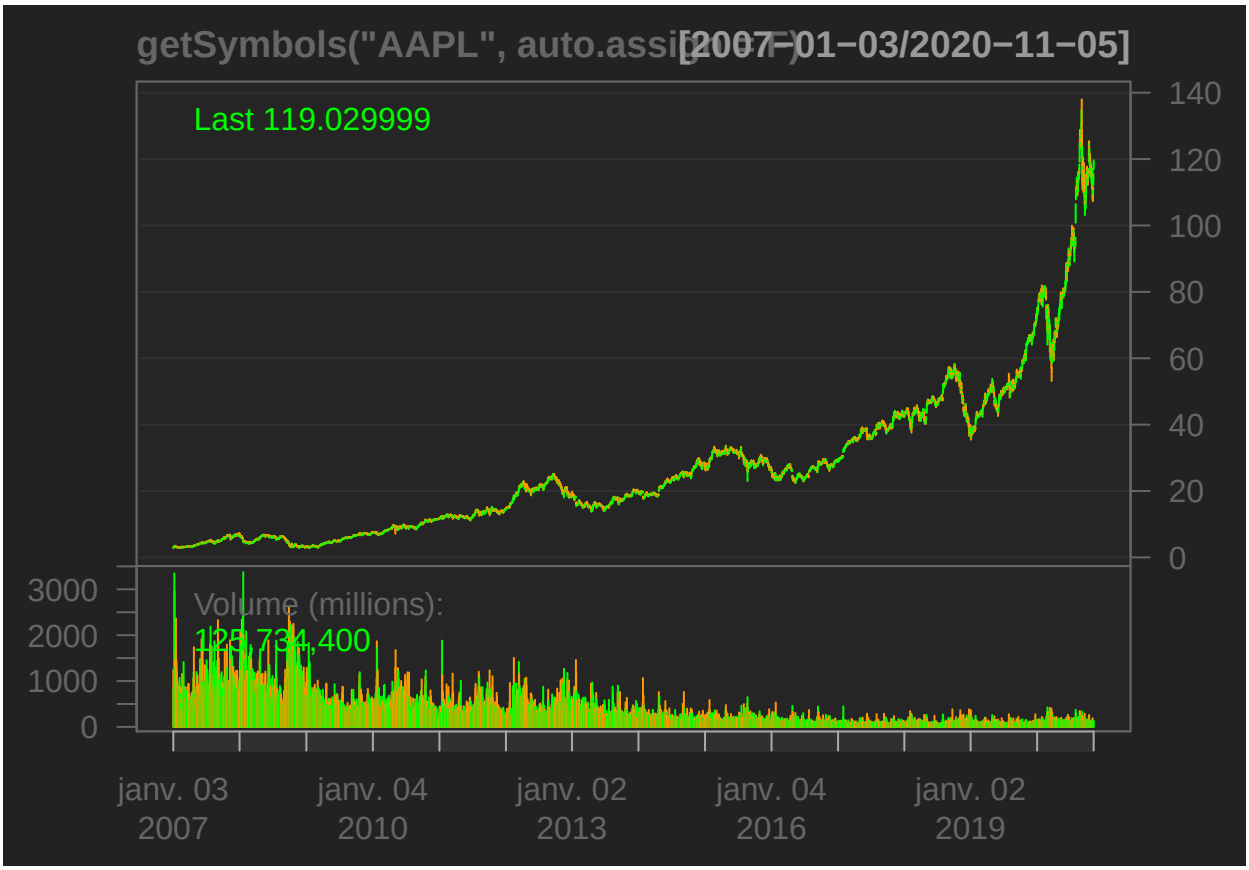
```
##                 Close   Adjusted
## 2007-01-04 0.02195311 0.02195314
## 2007-01-05 0.01480636 0.01480645
## 2007-01-08 0.01973248 0.01973253
## 2007-01-09 0.09953233 0.09953255
## 2007-01-10 0.14627810 0.14627801
## 2007-01-11 0.13382985 0.13382956
```

```
plot(compareAdtoClCumSum[,1] , type = 'l') #fermeture
lines(compareAdtoClCumSum[,2] , type = "l" , col = "red") #ajusté
```



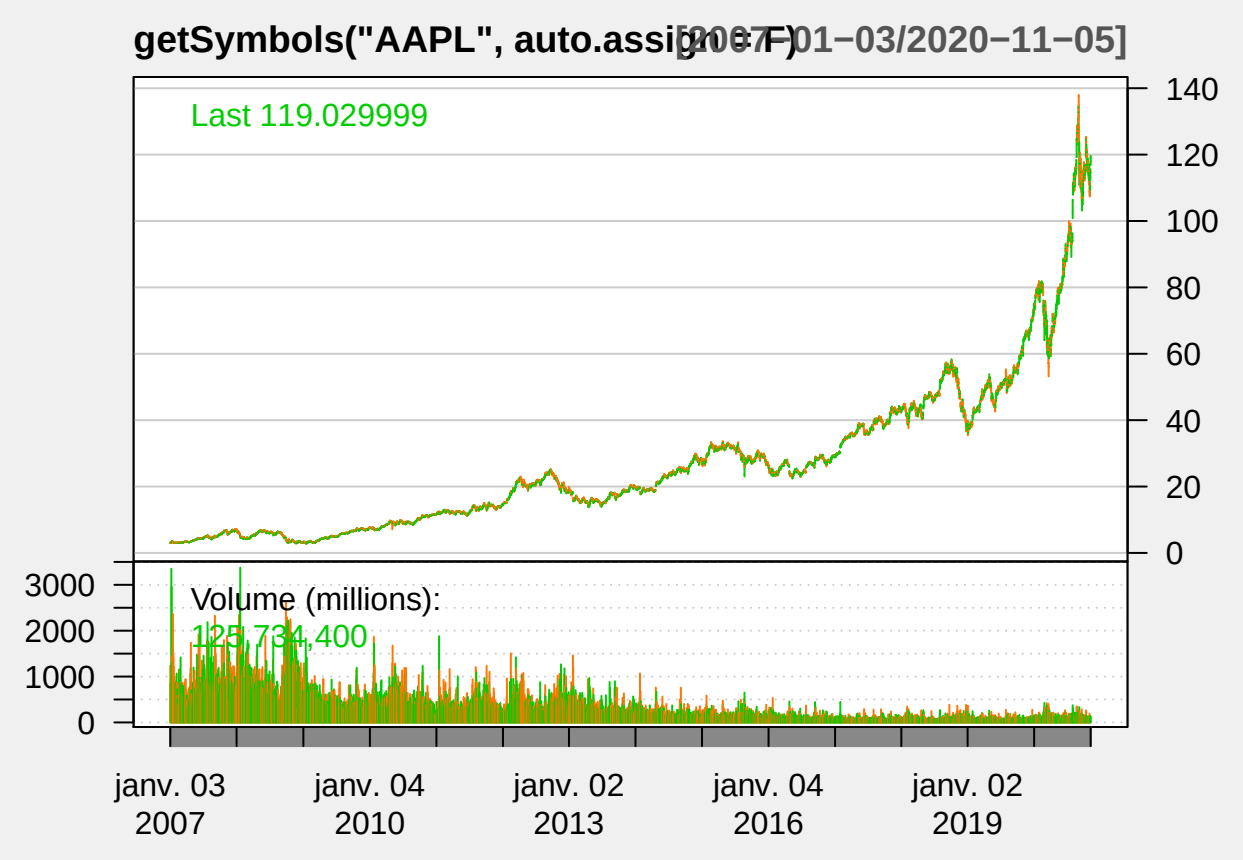
Q: Tester la fonction *barChart* suivante:

```
barChart(getSymbols("AAPL", auto.assign = F) ,theme=chartTheme('black'))
```

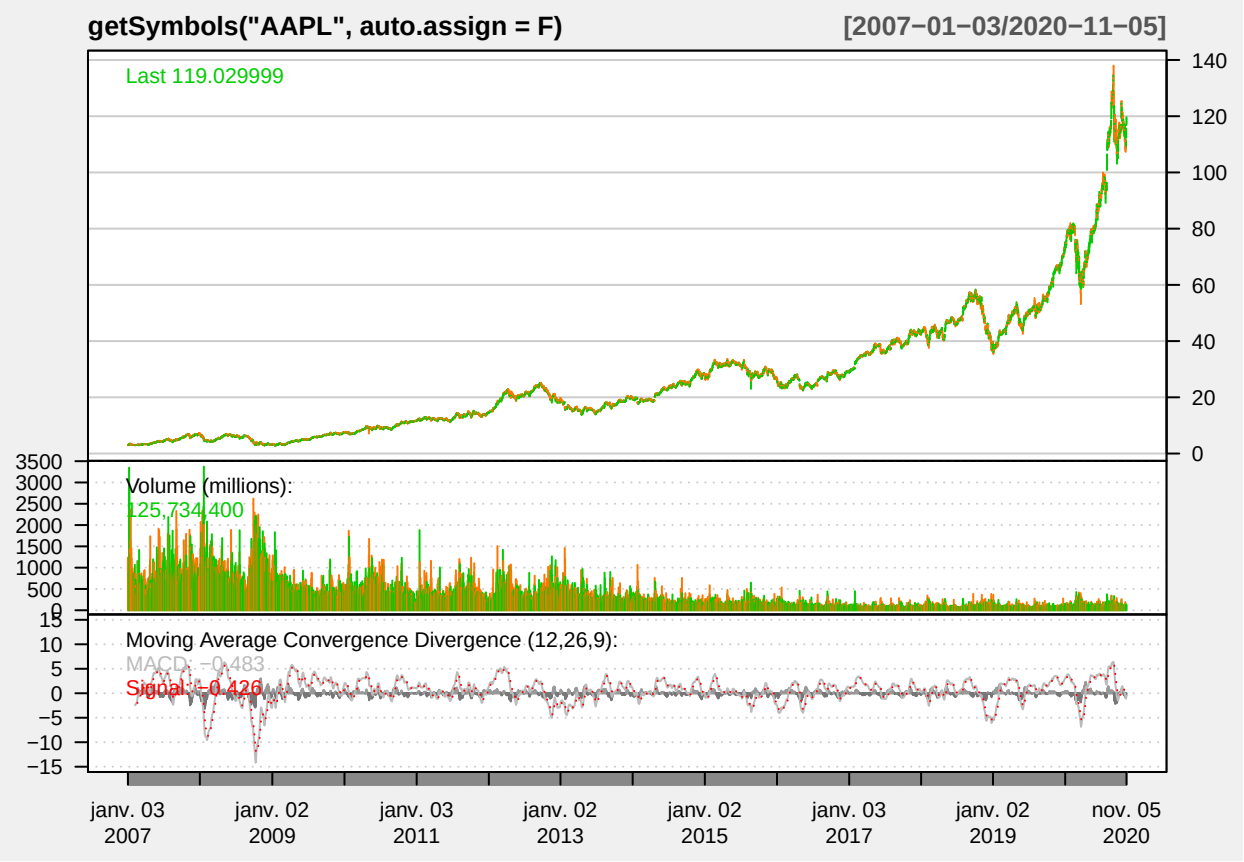


Q: Ajouter le MACD de la manière suivante:

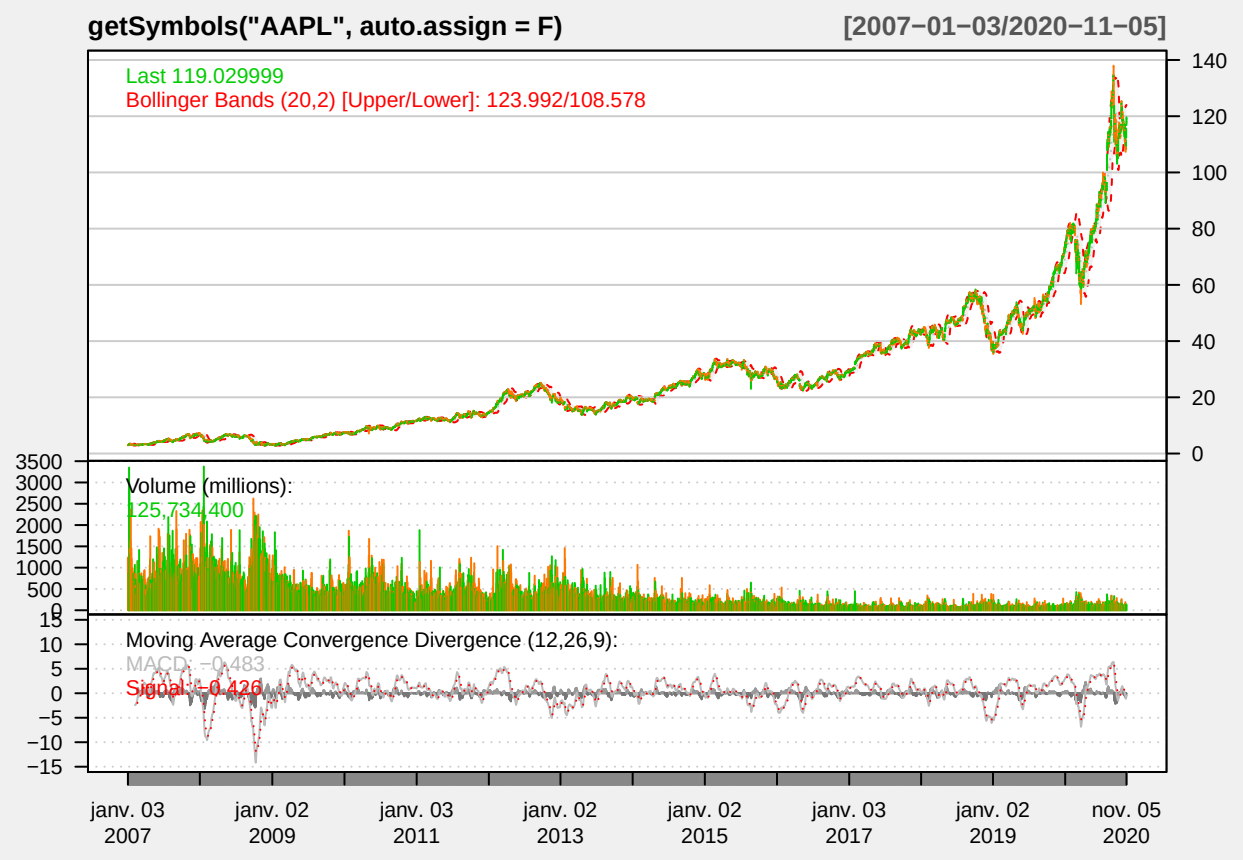
```
barChart(getSymbols("AAPL", auto.assign = F) ,theme=chartTheme('white'))
```



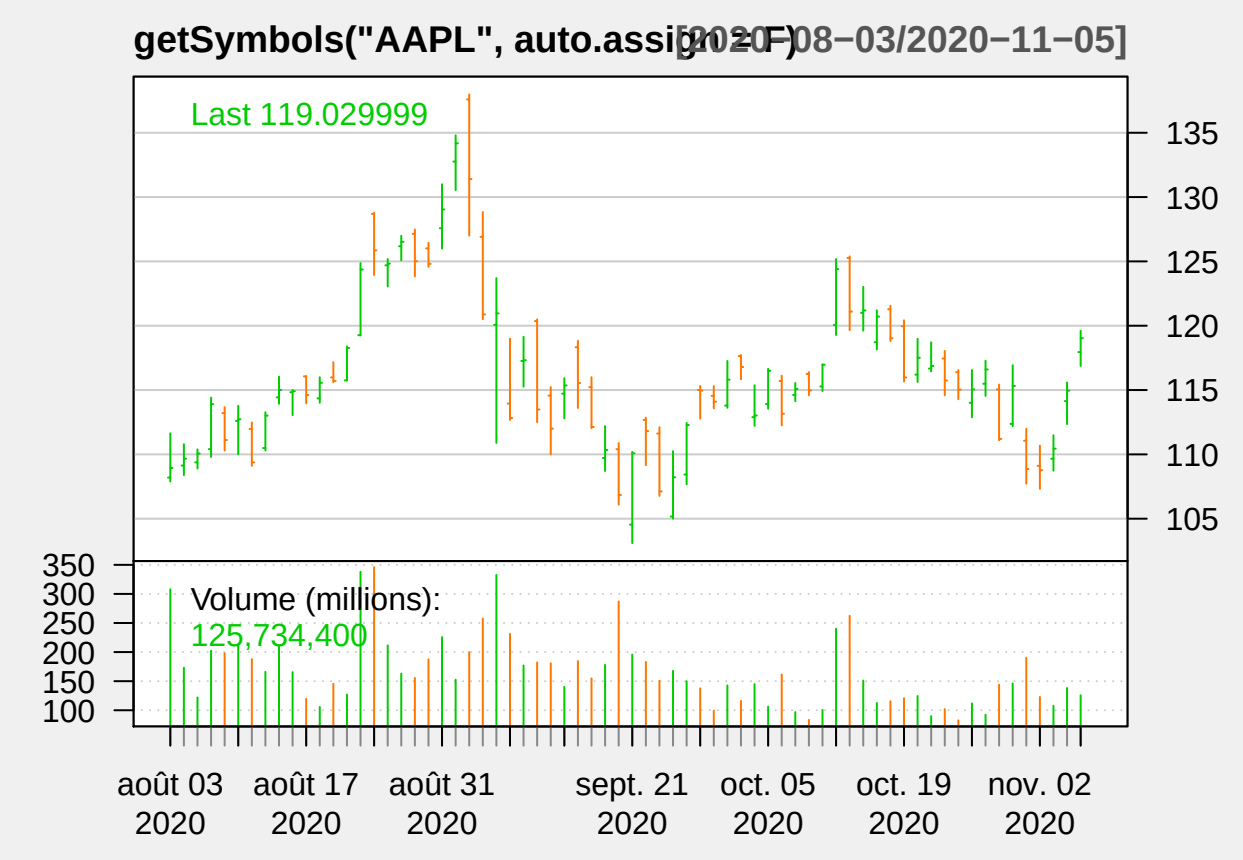

```
addMACD()
```


getSymbols("AAPL", auto.assign = F)
[2007−01−03/2020−11−05]
Last 119.029999
140
120
100
80
60
40
20
0
3500
3000
2500
2000
1500
1000
500
0
Volume (millions):
125,734,400
15
10
5
0
−5
−10
−15
Moving Average Convergence Divergence (12,26,9):
MACD: −0.483
Signal: −0.426
janv. 03
2007
janv. 02
2009
janv. 03
2011
janv. 02
2013
janv. 02
2015
janv. 03
2017
janv. 02
2019
nov. 05
2020

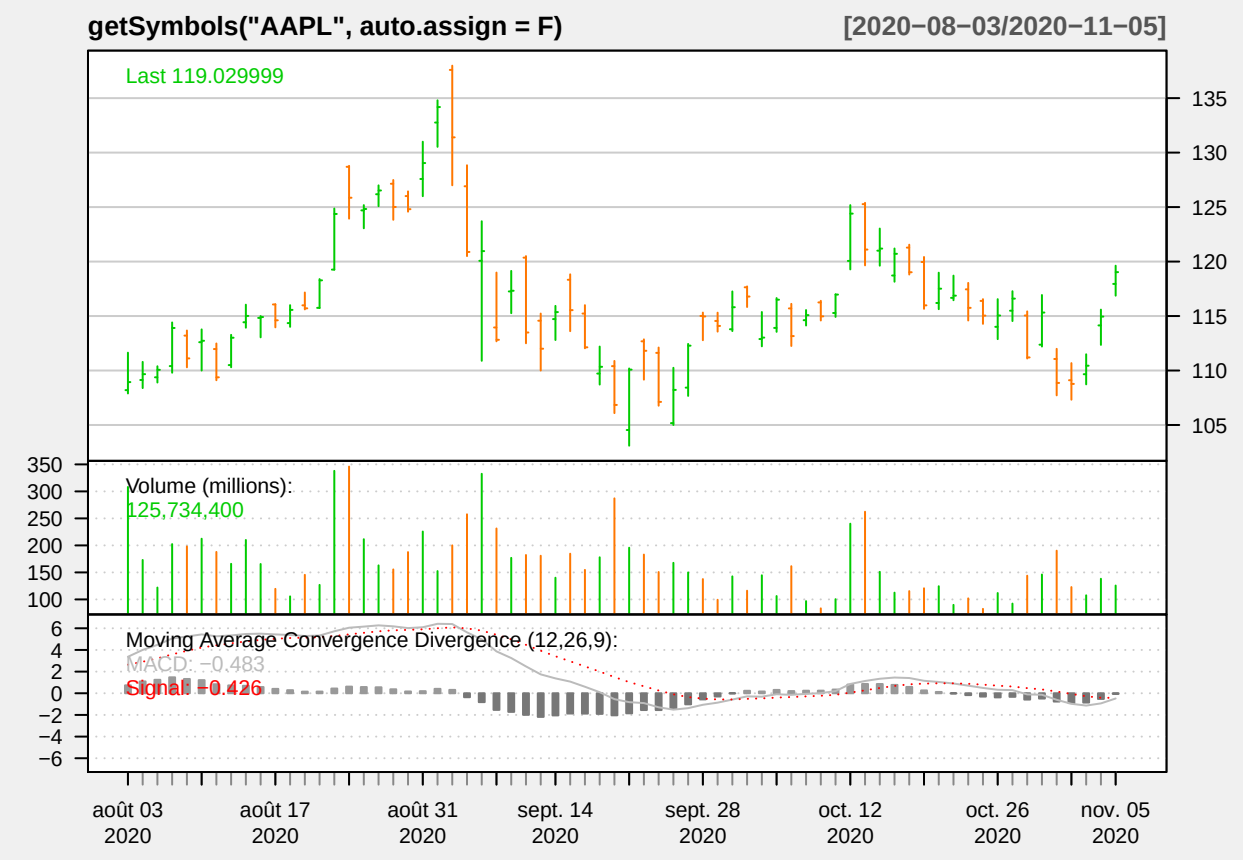
```
addBBands()
```



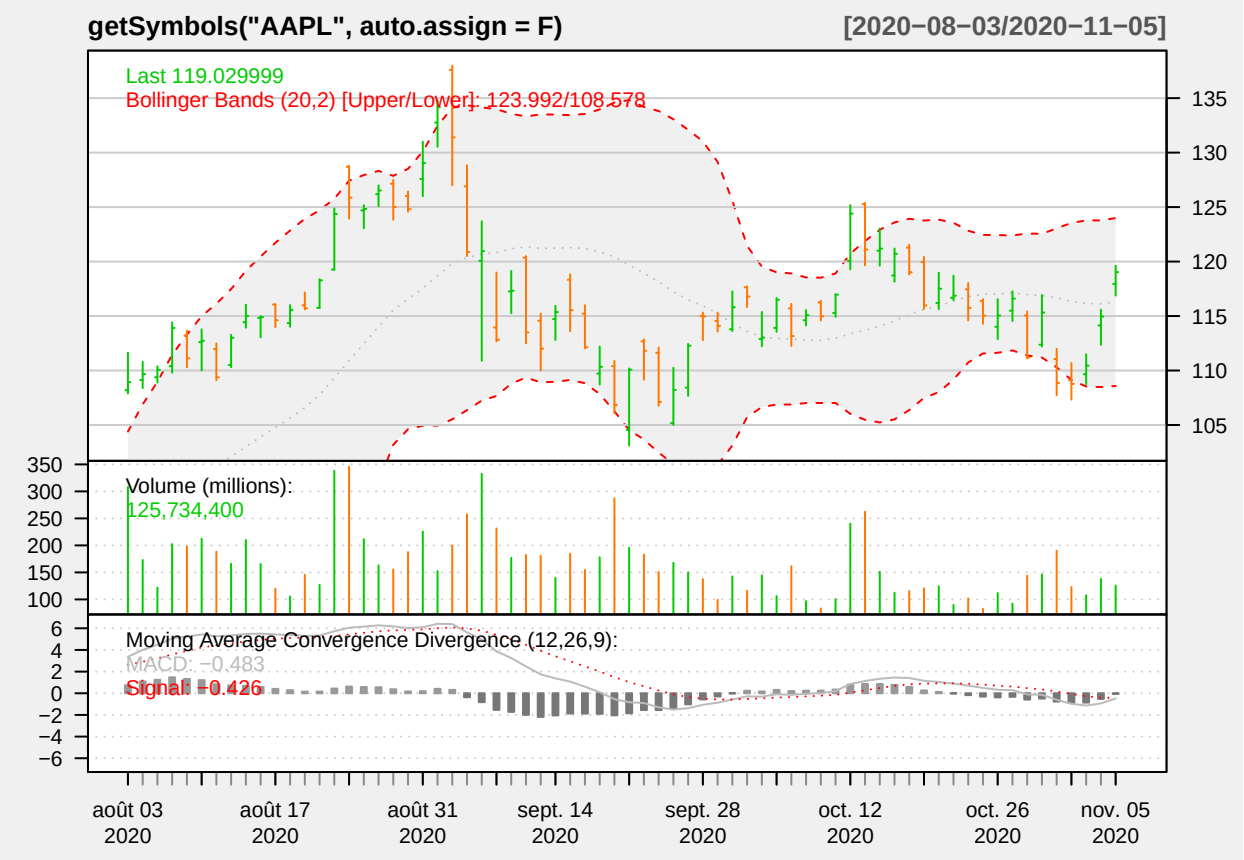

```
#Sur les quatre derniers mois
barChart(getSymbols("AAPL", auto.assign = F ) ,subset='last 4 months',theme=chartTheme('white'))
```


getSymbols("AAPL", auto.assign=F) [2020-08-03/2020-11-05]
Last 119.029999
135
130
125
120
115
110
105
350
300
250
200
150
100
Volume (millions):
125,734,400
août 03
2020
août 17
2020
août 31
2020
sept. 21
2020
oct. 05
2020
oct. 19
2020
nov. 02
2020

```
addMACD()
```


getSymbols("AAPL", auto.assign = F)
[2020−08−03/2020−11−05]
Last 119.029999
135
130
125
120
115
110
105
350
300
250
200
150
100
Volume (millions):
125,734,400
6
4
2
0
−2
−4
−6
Moving Average Convergence Divergence (12,26,9):
MACD: −0.483
Signal: −0.426
août 03
2020
août 17
2020
août 31
2020
sept. 14
2020
sept. 28
2020
oct. 12
2020
oct. 26
2020
nov. 05
2020

```
addBBands()
```



Librarie: Performances Analytiques

```
#install.packages("PerformanceAnalytics")
library(PerformanceAnalytics)
```

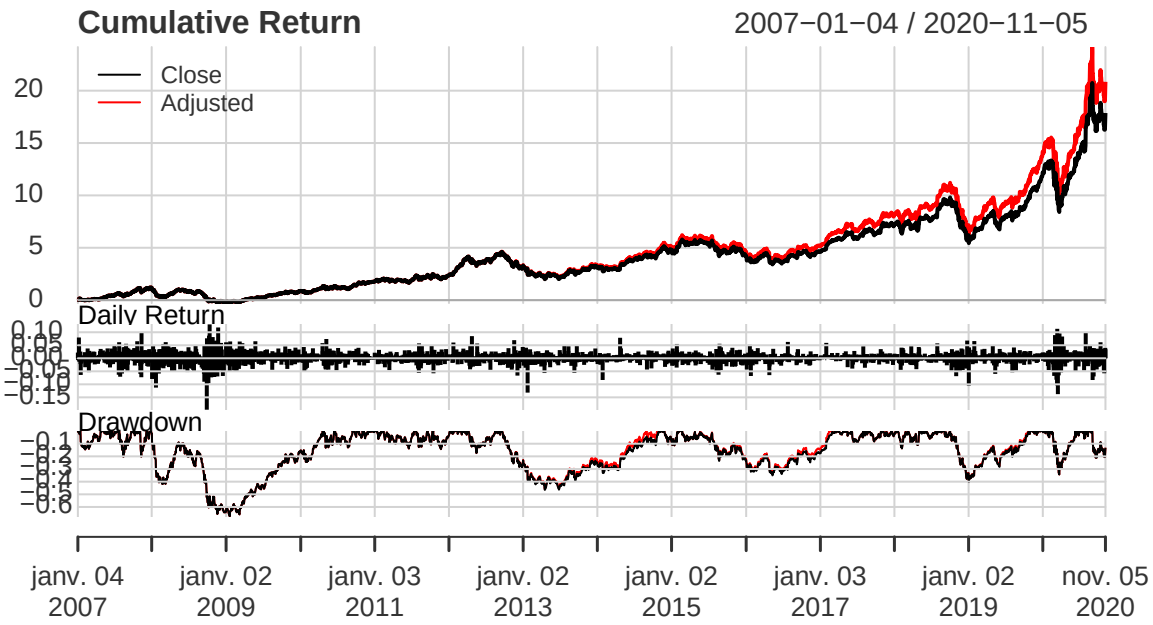
```
##
## Attaching package: 'PerformanceAnalytics'
```

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

Pour utiliser les fonctions de la librarie *PerformanceAnalytics* il faut avoir des objets type *xts*, on va donc convertir comme ci dessous. Commentez la figure:

```
data <- as.xts(compareAdtoCl)
charts.PerformanceSummary(data, main = "Comparaison")
```

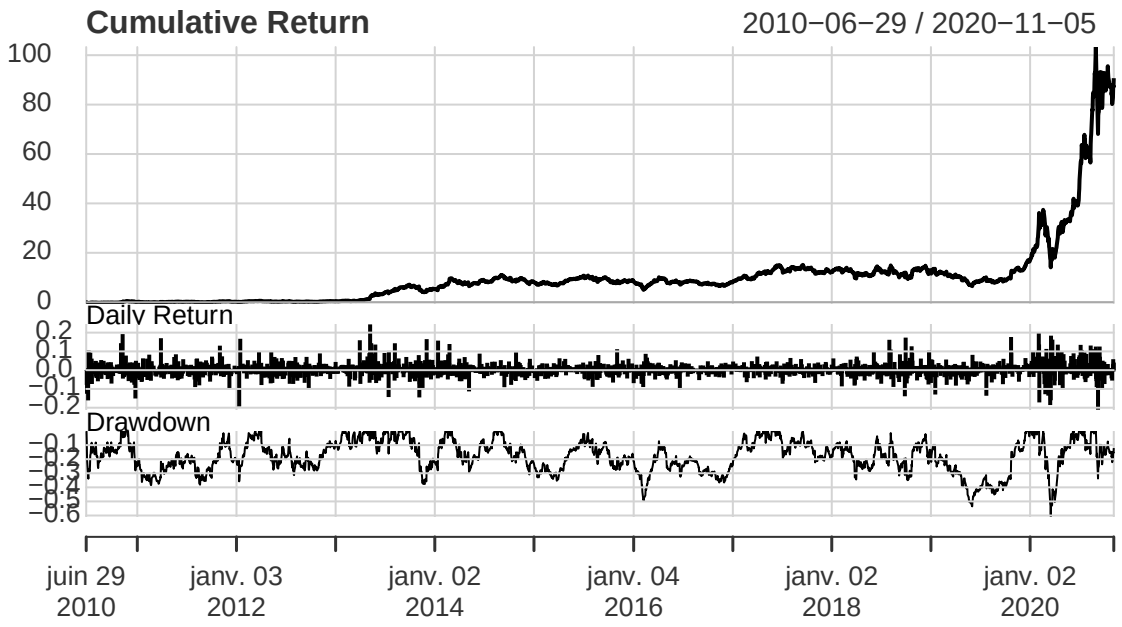
Comparaison



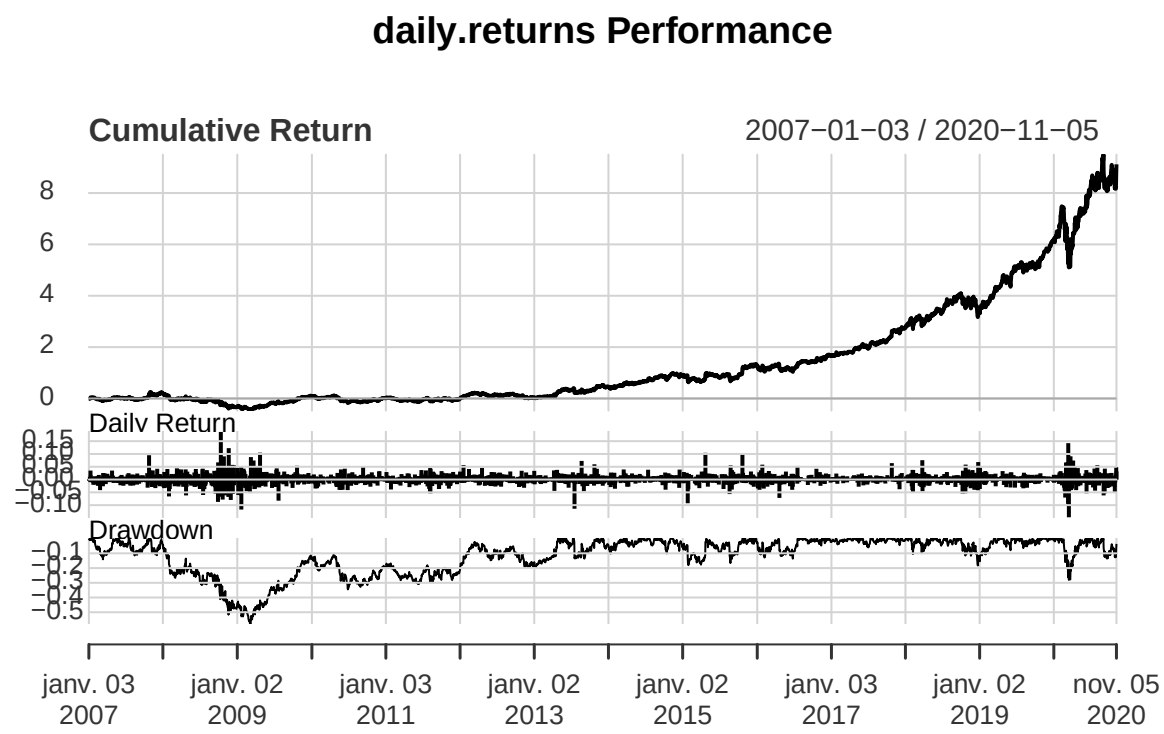
Comparons les données de 2 firmes

ON choisit de calculer le rendement journalier des prix ajustés avec la fonction *dailyReturn()* pour deux compagnies:
- Tesla : TSLA et stocker les infromations dans une variable appelée *TSLA* et appliquer *charts.PerformanceSummary*
- MIcrosoft: MFST et stocker les infromations dans une variable appelée *TSLA* et appliquer *charts.PerformanceSummary*

```
TSLA <- dailyReturn(Ad(getSymbols("TSLA" , auto.assign = F)))
charts.PerformanceSummary(TSLA)
```




```
MSFT <- dailyReturn(Ad(getSymbols("MSFT" , auto.assign = F)))
charts.PerformanceSummary(MSFT)
```



Q: Comparer les dates de début.

Par la suite on cherche à concaténer les deux tables, il faut donc renommer les colonnes pour éviter toute confusion.

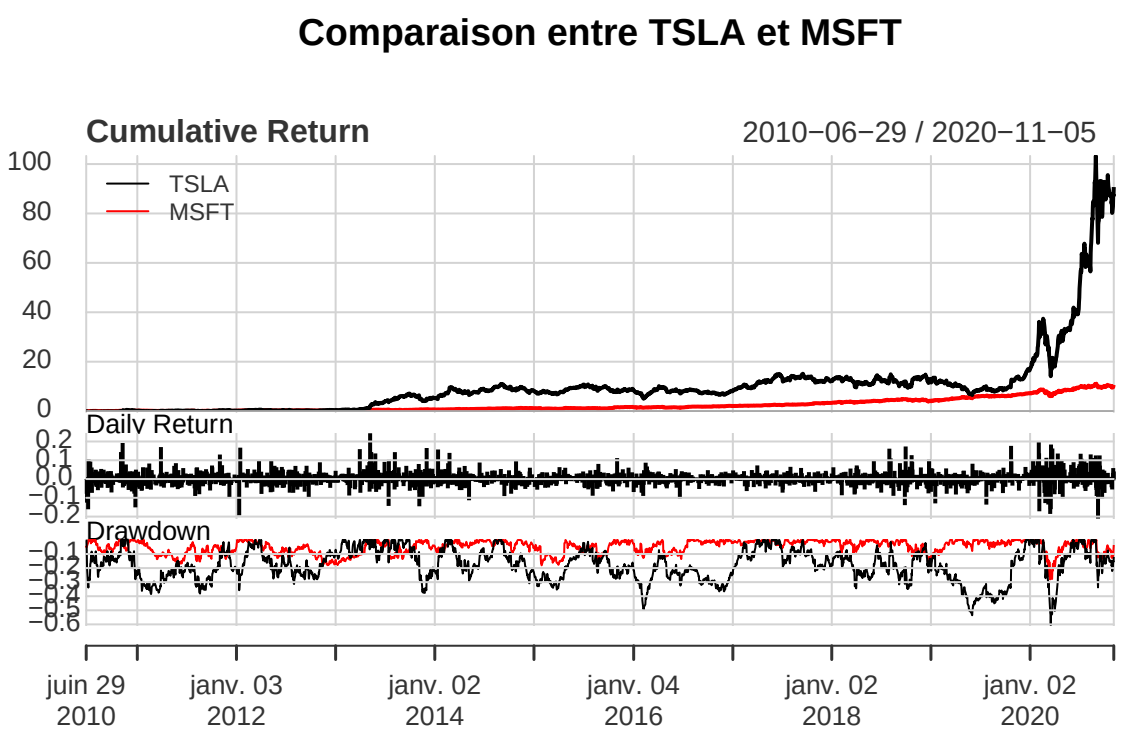
```
colnames(MSFT) <-"MSFT"
colnames(TSLA) <-"TSLA"
```

Q: Concatenez maintenant avec la fonction *merge()* , que remarquez vous ?
Retirez les lignes contenant des *NA*. (indice: soit à l'aide d'un argument de la fonction *merge* ou soIt avec la fonction *na.omit*). Stockez ce tableau concaténé dans une variable appelée *MSFTvsTSLA*:

```
#solution1
MSFTvsTSLA <- merge(TSLA,MSFT,all=F)
#solution2
MSFTvsTSLA <- merge(TSLA,MSFT)
MSFTvsTSLA <- na.omit(MSFTvsTSLA)
```

Q: Appliquer *charts.PerformanceSummary* sur ce nouveau tableau *MSFTvsTSLA*

```
charts.PerformanceSummary(MSFTvsTSLA , main = "Comparaison entre TSLA et MSFT")
```



Balance entre : return/risk

On va calculer le ratio de Sharp (mesure de la rentabilité) du portefeuille risqué. Rappel: > Ratio de Sharp = (Rendement - taux sans risque) / volatilité

-Si le ratio est négatif, on en conclut que le portefeuille sous performe un placement sans risque et donc il n'est pas logique d'investir dans un tel portefeuille.

-Si le ratio est compris entre 0 et 1, cela signifie que l'excédent de rendement par rapport au taux sans risque est plus faible que le risque pris.

-Si le ratio est supérieur à 1, alors le portefeuille surperforme un placement sans risque et donc il génère une plus forte rentabilité.

Plus le ratio est élevé et plus le portefeuille est performant.

Q: Tester et commentez, qui de Microsoft ou Tesla possède un plus grosr ratio

```r
table.AnnualizedReturns(MSFTvsTSLA , scale=252 , Rf = 0.005/252) #252 le nombre de jours ouvrés dans l'
```

Calculer le VaR pour les portfolios des stocks

ON va reprendre les données de Microsoft, mais cette fois ci entre le 1er janvier 2013 et le 1er janvier 2017. On peut préciser la source (yahoo) car il en existe d'autres (Google Finance, FRED, Oanda, local databases, CSVs).

Attention pour ladate, il faut respecter le format suivant: "yyyy-mm-dd".

```r
#On fixe les dates qui définissent la période
debut="2013-01-01"
fin="2017-01-01"

#on recupère les prix
MSFT <- getSymbols("MSFT", src = "yahoo", auto.assign = F, from = debut, to = fin)

#On recupère les rendements journaliers
MSFT.daily <- dailyReturn(MSFT)
```

On va calculer le Value-at-Risk(VaR) qui représente "la perte potentielle maximale d'un investisseur sur la valeur d'un actif ou d'un portefeuille d'actifs financiers compte tenu d'un horizon de détention et d'un intervalle de confiance."

Q: Inspecter la fonction VaR avec un *help* et calculer le VAR avec un niveau de confiance de 0.95 et avec la méthode *historical*.
Faire la même chose avec un niveau de confiance de 0.99.
Quelles sont les autres méthodes pour calculer le VAR ?

```r
VaR(MSFT.daily , p = 0.95 , method = "historical")
```

```
##     daily.returns
## VaR   -0.01959641
```

```r
VaR(MSFT.daily , p = 0.99 , method = "historical")
```

```
##     daily.returns
## VaR   -0.03821757
```

Q: Compararison du VAR avec les 3 méthodes différentes sur les 3 portefeuilles
Etapes:
1. Récupérer les prix ajustés pour: MSFT, AAPL et TSLA.
2. Fusionner les prix en un seul tableau appelé *all.adprices*(n'oubliez pas de retirez les valeurs manquantes NA)
3. Cette fois ci, au lieu d'utiliser le *DailyReturn* pour le rendement journalier, on va utiliser la fonction *ROC(all.adprices , type ="discrete")* pour calculer le changement sur une série de période. Appelez l'objet *all.returns*. et renommez les colonnes avec le noms des 3 entreprises.
4. Calculez sur *all.returns* le VAR avec un intervalle de confiance de 0.95 et le *portfolio_method = "component"* et la *method = "historical")*
5. Commentez le résultat; on s'apperçoit, d'après les contributions, que TSLA est plus à risque que AAPL et MSFT.
5bonus. Explorer les fonctions ETL, ES.
6. Comparez les 3 méthodes pour calculer le VAR (avec p=0.95 et portfolio_method = "single") : Hist, Gaus, Mod. (Conseil, stockez les séparemment dans les variables: *Var.Hist*, *VAR.Gaus* et *VAR.mod* puis créer un *data.frame* nommé *All.VAR* pour les rassembler)
7. A la fin vous devez obtenir le tableau suivant:

—	MSFT	AAPL	TSLA
Hist	-0.02290069	-0.02596607	-0.04905951
Gaus	-0.02547391	-0.02817714	-0.05629856
Mod	-0.02309386	-0.02749275	-0.05226586

8. Utilisez la fonction *abs* sur le tableau pour avoir des valeur positives.

```
getSymbols(c('MSFT','AAPL','TSLA'))
```

```
## [1] "MSFT" "AAPL" "TSLA"
```

```
all.adprices <- na.omit(merge(Ad(MSFT),Ad(AAPL),Ad(TSLA)))
#autre solution
all.adprices <- merge(Ad(MSFT),Ad(AAPL),Ad(TSLA) , all = F)

all.returns <- ROC(all.adprices)[-1]
colnames(all.returns) <- c('MSFT','AAPL','TSLA')
VaR(all.returns, p = 0.95, method = "historical" , portfolio_method = "component")
```

```
## no weights passed in, assuming equal weighted portfolio
```

```
## $hVaR
##  hVaR 95%
## 0.0283259
##
## $contribution
##         MSFT         AAPL         TSLA
## -0.002271672 -0.003303641 -0.008587639
##
## $pct_contrib_hVaR
##      MSFT      AAPL      TSLA
## 0.1603953 0.2332594 0.6063453
```

```
# first do normal VaR calc
VAR.Hist <- VaR(all.returns, p=.95, method="historical", portfolio_method = "single")
# now use Gaussian
VAR.Gaus <-VaR(all.returns, p=.95, method="gaussian" , portfolio_method = "single")
# now use modified Cornish Fisher calc to take non-normal distribution into account
VAR.Mod <- VaR(all.returns, p=.95, method="modified", portfolio_method ="single")

ALL.VAR <- data.frame(rbind(VAR.Hist,VAR.Gaus,VAR.Mod))
rownames(ALL.VAR) <- c("Hist","Gaus","Mod")

ALL.VAR <- abs(ALL.VAR )
```

Pour Marius, qui a essayé de faire le merge sur les daily return, voici la correction ^^ :

```
DR_comp <-na.omit(merge(MSFTvsTSLA, dailyReturn(AAPL)))
head(DR_comp)
```

```
##                    TSLA         MSFT daily.returns
## 2010-06-29  0.000000000 -0.041135373  -0.045210555
## 2010-06-30 -0.002511511 -0.012870025  -0.018113049
## 2010-07-01 -0.078472514  0.006519132  -0.012125727
## 2010-07-02 -0.125683060  0.004749473  -0.006197794
## 2010-07-06 -0.160937500  0.023635548   0.006843752
## 2010-07-07 -0.019242706  0.020151183   0.040381358
```

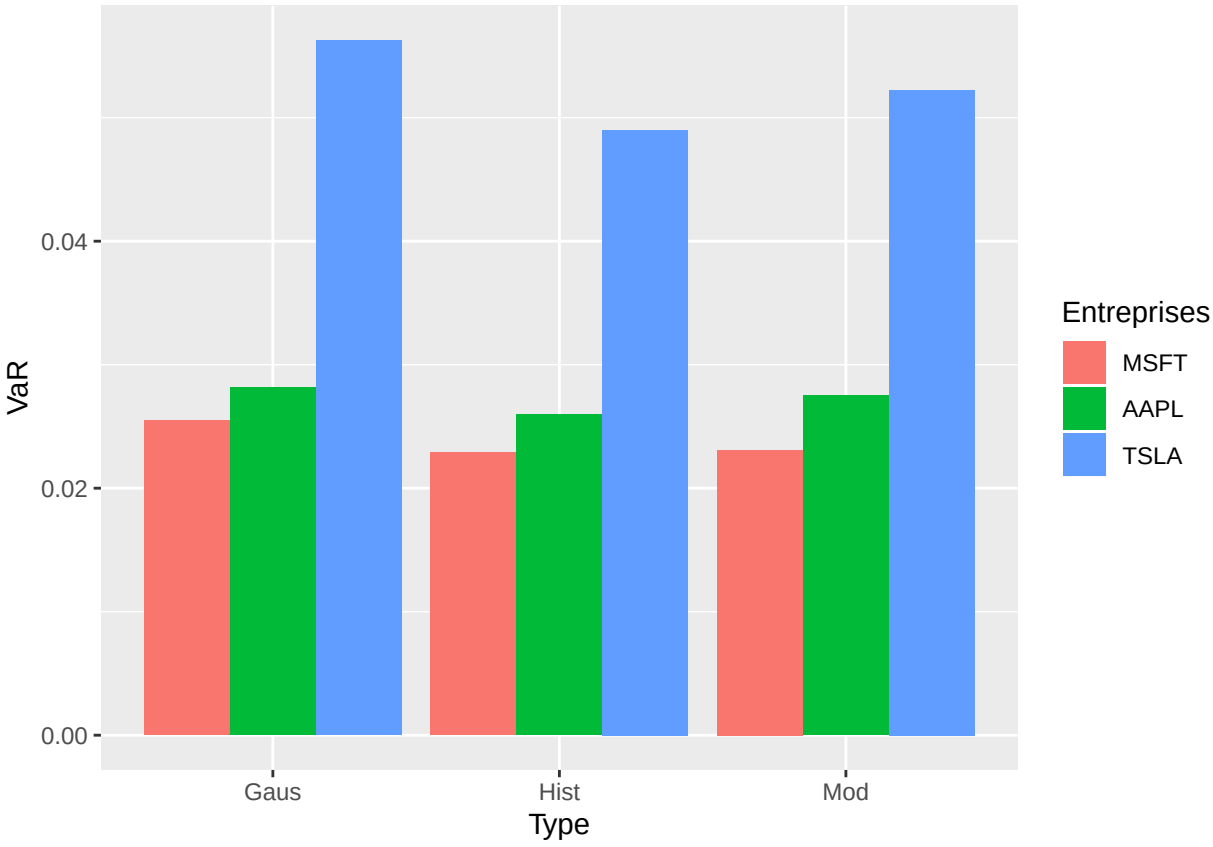
Q: On va représenter ces informations graphique à l'aide de la librarire *ggplot*.
Avant cela, exécutez:

```
ALL.VAR$Type <- c("Hist","Gaus","Mod")
library("reshape2")
library("ggplot2")

#si elles ne sont pas installées, faire:
#install.packages("reshape2")
#install.packages("ggplot2")
plotVar <- melt(ALL.VAR, variable.name = "Entreprises", value.name = "VaR")
```

```
## Using Type as id variables
```

```
ggplot(plotVar , aes(x=Type,y=VaR , fill = Entreprises)) +
  geom_bar(stat = "identity" , position = "dodge" )
```



Découvrir la librairie highcharter pour une visalisation interactive

```
install.packages("highcharter")
```

```
library("highcharter")
```

```
lot1 <- highchart(type = "stock") %>%
  hc_title(text = "Stocks Evolution of ") %>%
  hc_add_series(Ad(AAPL), name="Apple") %>%
    hc_add_series(Ad(MSFT), name="Microsoft")
```