Coletando dados financeiros

Lucca Simeoni Pavan *João Carlos de Carvalho*

10 de novembro de 2016

Sumário


```
knitr::opts_chunk$set(echo = TRUE, cache = TRUE, warning = FALSE, message = FALSE,
    error = FALSE, tidy = TRUE, tidy.opts = list(width.cutoff = 70))
```

1 Coletando dados usando o pacote `GetHFData`

Os dados podem ser coletados usando o pacote `GetHFData` desenvolvido por Perlin (2016). Para maiores detalhes sobre o pacote veja também Perlin and Ramos (2016). Primeiramente baixaremos os *layouts* da base de dados usando o comando `gthf_download_file`.

```
library(GetHFData)
layout_negocios <- "ftp://ftp.bmf.com.br/MarketData/NEG_LAYOUT_portuguese.txt"
ghfd_download_file(layout_negocios, out.file = "layout_negocios")
```

```
##  Attempt 1 - File exists, skipping dl
```

```
layout_oferta_compra <- "ftp://ftp.bmf.com.br/MarketData/OFER_CPA_LAYOUT_portuguese.txt"
ghfd_download_file(layout_oferta_compra, out.file = "layout_oferta_compra")
```

```
##  Attempt 1 - File exists, skipping dl
```

```
layout_oferta_venda <- "ftp://ftp.bmf.com.br/MarketData/OFER_VDA_LAYOUT_portuguese.txt"
ghfd_download_file(layout_oferta_venda, out.file = "layout_oferta_venda")
```

```
##  Attempt 1 - File exists, skipping dl
```

`Attempt 1` e `TRUE` significam que o download na primeira tentativa foi realizado com sucesso. A mensagem `File exists, skipping dl` aparece quando o comando for acionado pela segunda vez e portanto o documento já foi baixado. Os arquivos de *layout* podem ser abertos pelo bloco de notas.

O comando `ghfd_get_ftp_contents` acessa o ftp da Bovespa e retorna um vetor com todos os arquivos relacionadosm à negócios (todos os outros são ignorados).

```
library("GetHFData")
contents_equity <- ghfd_get_ftp_contents(type.market = "equity")
```

```
##
## Reading ftp contents for equity (attempt = 1|10)
```

```
contents_options <- ghfd_get_ftp_contents(type.market = "options")
```

```
##
## Reading ftp contents for options (attempt = 1|10)
```

```
contents_bmf <- ghfd_get_ftp_contents(type.market = "BMF")
```

```
##
## Reading ftp contents for BMF (attempt = 1|10)
```

Usando os comandos `head` e `tail` podemos ver os 6 primeiros e 6 últimos elementos dos arquivos baixados anteriormente.

```
head(contents_equity)
```

```
##              files      dates
## 1 NEG_20141103.zip 2014-11-03
## 2 NEG_20141104.zip 2014-11-04
## 3 NEG_20141105.zip 2014-11-05
## 4 NEG_20141106.zip 2014-11-06
## 5 NEG_20141107.zip 2014-11-07
## 6 NEG_20141110.zip 2014-11-10
##                                                          link
## 1 ftp://ftp.bmf.com.br/marketdata/Bovespa-Vista/NEG_20141103.zip
## 2 ftp://ftp.bmf.com.br/marketdata/Bovespa-Vista/NEG_20141104.zip
## 3 ftp://ftp.bmf.com.br/marketdata/Bovespa-Vista/NEG_20141105.zip
## 4 ftp://ftp.bmf.com.br/marketdata/Bovespa-Vista/NEG_20141106.zip
## 5 ftp://ftp.bmf.com.br/marketdata/Bovespa-Vista/NEG_20141107.zip
## 6 ftp://ftp.bmf.com.br/marketdata/Bovespa-Vista/NEG_20141110.zip
```

```
tail(contents_equity)
```

```
##                files      dates
## 462 NEG_20160823.zip 2016-08-23
## 463 NEG_20160824.zip 2016-08-24
## 464 NEG_20160825.zip 2016-08-25
## 465 NEG_20160826.zip 2016-08-26
## 466 NEG_20160829.zip 2016-08-29
```

```
## 467 NEG_20160830.zip 2016-08-30
##                                                                 link
## 462 ftp://ftp.bmf.com.br/marketdata/Bovespa-Vista/NEG_20160823.zip
## 463 ftp://ftp.bmf.com.br/marketdata/Bovespa-Vista/NEG_20160824.zip
## 464 ftp://ftp.bmf.com.br/marketdata/Bovespa-Vista/NEG_20160825.zip
## 465 ftp://ftp.bmf.com.br/marketdata/Bovespa-Vista/NEG_20160826.zip
## 466 ftp://ftp.bmf.com.br/marketdata/Bovespa-Vista/NEG_20160829.zip
## 467 ftp://ftp.bmf.com.br/marketdata/Bovespa-Vista/NEG_20160830.zip
```

O primeiro dia disponível para o mercado de ações (*equity*) é 2014-11-03 e o último é 2016-08-30. Os arquivos `.zip` armazenam dados das transações diárias e obviamente somente de segunda à sexta-feira.

Para sabermos os *tickers* (nomes dos ativos transacionados, ex. para o mercado de ações `PETR4`, é um *ticker* para ações da PETROBRAS) podemos usar o comando `ghfd_get_available_tickers_from_file` que obtém os *tickers* disponíveis de um arquivo baixado do ftp da Bovespa ou podemos usar o comando `ghfd_get_available_tickers_from_ftp` que obtém os *tickers* disponíveis em um mercado e uma data específicos. Os dois comandos apresentam como resultado um vetor numérico com os tickers e outro com o número de transações de cada *ticker*.

```
tickers_equity <- ghfd_get_available_tickers_from_ftp(my.date = "2015-11-03",
    type.market = "equity", max.dl.tries = 10)
```

```
##
## Reading ftp contents for equity (attempt = 1|10) Attempt 1 - File exists, skipping dl
```

```
head(tickers_equity)
```

```
##   tickers n.trades                     f.name
## 1   PETR4    52231 ftp files/NEG_20151103.zip
## 2   ITUB4    50437 ftp files/NEG_20151103.zip
## 3   BVMF3    47214 ftp files/NEG_20151103.zip
## 4   VALE5    41959 ftp files/NEG_20151103.zip
## 5   BBDC4    39403 ftp files/NEG_20151103.zip
## 6   ITSA4    37993 ftp files/NEG_20151103.zip
```

Existem 419 *tickers* para o mercado de ações na data especificada.

Para baixar os dados de transações de alta frequência e agregá-los para análise usamos o comando `ghfd_get_HF_data`. Para exemplo usarei os três *tickers* mais comercializados no mercado de ações em 03/11/2015, coletados no período de 30/06/2016 a 30/08/2016.

```
dados_top3 <- ghfd_get_HF_data(c("PETR4", "ITUB4", "BVMF3"), type.market = "equity",
    first.date = as.Date("2016-06-30"), last.date = as.Date("2016-08-30"),
    first.time = "9:00:00", last.time = "18:00:00", type.output = "agg",
    agg.diff = "1 hour", dl.dir = "ftp files", max.dl.tries = 10, clean.files = FALSE)
```

```
load("dados_top3.Rda")
head(dados_top3, n = 3)
```

```
##   InstrumentSymbol SessionDate       TradeDateTime n.trades last.price
## 1            BVMF3  2016-06-30 2016-06-30 10:00:00     2992      17.63
## 2            BVMF3  2016-06-30 2016-06-30 11:00:00     3642      17.67
```

```
## 3            BVMF3  2016-06-30 2016-06-30 12:00:00     2289      17.72
##   weighted.price  period.ret period.ret.volat sum.qtd  sum.vol n.buys
## 1       17.53706 0.021436848     0.0003225179 1523500 26716617   1238
## 2       17.62966 0.001700680     0.0003044433 1200900 21171287   1395
## 3       17.68812 0.002829655     0.0003512668 1156900 20463311   1079
##   n.sells Tradetime
## 1    1754  10:00:00
## 2    2247  11:00:00
## 3    1210  12:00:00
```

```
tail(dados_top3, n = 3)
```

```
##      InstrumentSymbol SessionDate       TradeDateTime n.trades last.price
## 1054            PETR4  2016-08-30 2016-08-30 15:00:00     4943      13.02
## 1055            PETR4  2016-08-30 2016-08-30 16:00:00     5006      13.06
## 1056            PETR4  2016-08-30 2016-08-30 17:00:00      489      13.15
##      weighted.price   period.ret period.ret.volat sum.qtd   sum.vol n.buys
## 1054       13.02425 -0.003062787     0.0003166287 4252300  55382934   1635
## 1055       13.02341  0.003072197     0.0003043510 5535600  72092146   2506
## 1056       13.09081  0.004583652     0.0003054307 9056300 118554268    184
##      n.sells Tradetime
## 1054    3308  15:00:00
## 1055    2500  16:00:00
## 1056     305  17:00:00
```

Por fim o comando `ghfd_read_file` baixa os dados na sua forma bruta, ou seja apenas lê o arquivo `.zip` baixado do ftp da Bovespa. Nesta opção fica disponível o código da corretora que efetuou a transação.

```
library("GetHFData")
path <- path.expand("~/artigo_macroeconometria_lucca_joao/ftp files/NEG_20160830.zip")
dados_bruto <- ghfd_read_file(out.file = path, my.assets = NULL, first.time = "10:00:00",
    last.time = "17:00:00", type.output = "raw")
```

```
##  - Imported  713224 lines, 475 unique tickers
##    -> Processing file - Found 713224 lines, 475 unique tickers
```

```
head(dados_bruto)
```

```
## # A tibble: 6 x 10
##   SessionDate InstrumentSymbol TradePrice TradedQuantity    Tradetime
##         <date>            <chr>      <dbl>          <dbl>        <chr>
## 1  2016-08-30           AALC34      32.81            800 16:10:39.669
## 2  2016-08-30           AAPL34      34.50           3600 16:05:22.618
## 3  2016-08-30           AAPL34      34.15           8700 16:10:39.669
## 4  2016-08-30           ABCB10      14.21            500 10:00:57.694
## 5  2016-08-30           ABCB10      14.00           1000 15:01:20.909
## 6  2016-08-30           ABCB10      14.00            400 15:15:49.496
## # ... with 5 more variables: CrossTradeIndicator <int>, BuyMember <dbl>,
## #   SellMember <dbl>, TradeDateTime <time>, TradeSign <dbl>
```

```
tail(dados_bruto)
```

```
## # A tibble: 6 x 10
##   SessionDate InstrumentSymbol TradePrice TradedQuantity    Tradetime
##         <date>            <chr>      <dbl>          <dbl>        <chr>
## 1  2016-08-30           XTED11      22.56             30 16:42:14.335
## 2  2016-08-30           XTED11      22.52             85 16:42:14.335
## 3  2016-08-30           XTED11      22.57            500 16:42:14.335
## 4  2016-08-30           XTED11      22.52              3 16:42:14.335
## 5  2016-08-30           XTED11      22.55              6 16:42:14.335
## 6  2016-08-30           XTED11      22.52            172 16:44:59.661
## # ... with 5 more variables: CrossTradeIndicator <int>, BuyMember <dbl>,
## #   SellMember <dbl>, TradeDateTime <time>, TradeSign <dbl>
```

```
head(dados_bruto[, 5:8])
```

```
## # A tibble: 6 x 4
##      Tradetime CrossTradeIndicator BuyMember SellMember
##          <chr>               <int>     <dbl>      <dbl>
## 1 16:10:39.669                   0        40         40
## 2 16:05:22.618                   1       238        238
## 3 16:10:39.669                   0        40         40
## 4 10:00:57.694                   0        58        174
## 5 15:01:20.909                   0       735        114
## 6 15:15:49.496                   0        15        114
```

```
tail(dados_bruto[, 9:10])
```

```
## # A tibble: 6 x 2
##         TradeDateTime TradeSign
##                 <time>     <dbl>
## 1 2016-08-30 16:42:14        -1
## 2 2016-08-30 16:42:14        -1
## 3 2016-08-30 16:42:14        -1
## 4 2016-08-30 16:42:14        -1
## 5 2016-08-30 16:42:14        -1
## 6 2016-08-30 16:44:59        -1
```

2 Coletando dados usando `quantmod`

Dados do mercado financeiro podem ser baixados por um outro pacote chamado **quantmod**. Este pacote baixa os dados de fontes como o Yahoo Finance, Goole Finace e diversas outras fontes (Tsay 2012). A forma que este pacote trabalha á diferente do pacote `GetHFData`. Com o **quantmod** não é necessário designar objetos, pois este pacote trabalha com objetos ocultos. Segue uma demonstração:

```
library(quantmod)
getSymbols(c("PETR4", "ITUB4", "BVMF3"), src = "google", env = globalenv())
```

```
## [1] "PETR4" "ITUB4" "BVMF3"
```

```
head(PETR4)
```

```
##            PETR4.Open PETR4.High PETR4.Low PETR4.Close PETR4.Volume
## 2007-01-02      25.00      25.22     24.88       25.22     10221200
## 2007-01-03      25.08      25.20     24.00       24.35     19822400
## 2007-01-04      24.25      24.38     23.70       23.82     20910800
## 2007-01-05      23.60      24.00     22.55       23.10     24798200
## 2007-01-08      23.25      23.57     22.90       23.30     19406000
## 2007-01-09      22.98      23.20     22.30       22.76     25847800
```

```
tail(BVMF3)
```

```
##            BVMF3.Open BVMF3.High BVMF3.Low BVMF3.Close BVMF3.Volume
## <NA>            18.30      19.01     18.26       18.80     11857600
## 2016-11-01      18.80      18.96     18.01       18.25     16018300
## 2016-11-03      18.20      18.33     17.70       17.72      9751600
## 2016-11-04      17.69      18.15     17.52       17.59      8132300
## 2016-11-07      17.94      18.60     17.85       18.56      9484100
## 2016-11-08      18.31      18.69        NA       18.40      6010800
```

```
dim(ITUB4)
```

```
## [1] 2437    5
```

Podemos perceber que os dados fornecidos pelo Google Finance são diários e iniciam em 2007, porém alguns intervalos de datas não estão disponíveis para dados com fonte no Google Finance (provavelmente por conflito de feriados) e os dados para ativos financeiros brasileiros até a data de escrita deste documento só foram encontrados no Google Finance. Realizei o comando `tail` para os demais ativos e estes também não apresentaram as datas mais recentes. Esta base de dados também não fornece o preço ajustado (para ativos financeiros brasileiros) e os dados brutos com discriminação por corretora como na seção anterior. Além disso ao utilizar o pacote `bizdays` que estabelece os dias úteis para o mercado financeiro brasileiro com base no caendário da AMBIMA ainda existe algum conflito entre os dias úteis apresentados pelo Google Finance e pelo `bizdays`. Para mais detalhes sobre o pacote `quantmod` veja Ryan (2016).

3 Organizando a base de dados

A partir de agora urilizarei os dados obtidos usando o pacote `GetHFData` devido ao problema com o pacote `quantmod` relatado anteriormente. a base de dados `dados_top3` contém as informações sobre os trê ativos `PETR4, ITUB4, BVMF3` no mesmo banco de dados. Portanto temos que separar este banco de dados em três outros arquivos cada um com informações a respeito de apenas um tipo de ação.

Para isso podemos utilizar a função `filter()` do pacote `dplyr`. O banco de dados `dados_top3` possui dimensão (1056, 13).

```
library(dplyr)
PETR4_data <- filter(dados_top3, InstrumentSymbol == "PETR4")
BVMF3_data <- filter(dados_top3, InstrumentSymbol == "BVMF3")
ITUB4_data <- filter(dados_top3, InstrumentSymbol == "ITUB4")
```

Então ele será dividido em três bancos de dados de mesma dimensão (352, 13).

3.1 Plotando os retornos dos ativos

```
library(ggplot2)
library(gridExtra)
plot_PETR4 <- ggplot(PETR4_data, aes(TradeDateTime, period.ret)) +
    geom_line() + scale_x_datetime(date_labels = "%Y-%m-%d %H:%M:%S") +
    xlab("") + ylab("PETR4") + theme(axis.text.x = element_text(size = 7,
    vjust = 0.7, hjust = 0.9))
plot_BVMF3 <- ggplot(BVMF3_data, aes(TradeDateTime, period.ret)) +
    geom_line() + scale_x_datetime(date_labels = "%Y-%m-%d %H:%M:%S") +
    xlab("") + ylab("BVMF3") + theme(axis.text.x = element_text(size = 7,
    vjust = 0.7, hjust = 0.9))
plot_ITUB4 <- ggplot(ITUB4_data, aes(TradeDateTime, period.ret)) +
    geom_line() + scale_x_datetime(date_labels = "%Y-%m-%d %H:%M:%S") +
    xlab("") + ylab("ITUB4") + theme(axis.text.x = element_text(size = 7,
    vjust = 0.7, hjust = 0.9))
grid.arrange(plot_PETR4, plot_BVMF3, plot_ITUB4, name = "Retornos das ações",
    nrow = 3)
```

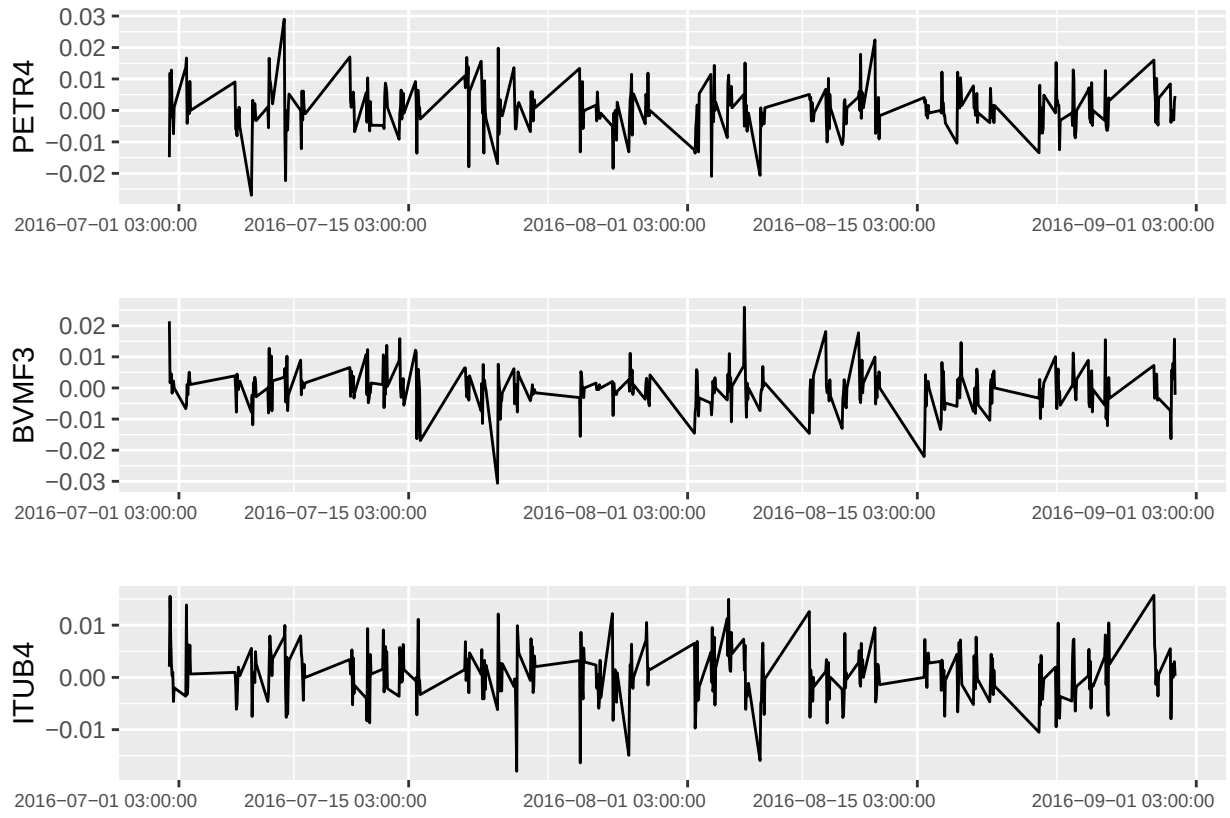


Figura 1: Retornos doa ativos

Visualente podemos perceber que os retornos oscilam em torno de zero e que esporadicamente ocorrems valores extremos.

4 Propriedades da distribuição dos retornos

Para conhecermos as características da distribuição dos dados usaremos o pacote `fBasics`. Inicialmente são apresentadas algumas estatísticas básicas.

```
library(fBasics)
basicStats(PETR4_data$period.ret)
```

```
##             X..PETR4_data.period.ret
## nobs                      352.000000
## NAs                         0.000000
## Minimum                    -0.026943
## Maximum                     0.028986
## 1. Quartile                -0.003815
## 3. Quartile                 0.004444
## Mean                        0.000435
## Median                      0.000000
## Sum                         0.153196
## SE Mean                     0.000388
## LCL Mean                   -0.000329
## UCL Mean                    0.001199
## Variance                    0.000053
## Stdev                       0.007287
## Skewness                    0.017503
## Kurtosis                    1.384462
```

```
basicStats(BVMF3_data$period.ret)
```

```
##             X..BVMF3_data.period.ret
## nobs                      352.000000
## NAs                         0.000000
## Minimum                    -0.030617
## Maximum                     0.025946
## 1. Quartile                -0.003256
## 3. Quartile                 0.003008
## Mean                       -0.000101
## Median                      0.000000
## Sum                        -0.035675
## SE Mean                     0.000337
## LCL Mean                   -0.000764
## UCL Mean                    0.000561
## Variance                    0.000040
## Stdev                       0.006322
## Skewness                   -0.058064
## Kurtosis                    2.784860
```

```
basicStats(ITUB4_data$period.ret)
```

```
##             X..ITUB4_data.period.ret
## nobs                      352.000000
## NAs                         0.000000
## Minimum                    -0.017962
```

```
## Maximum                    0.015730
## 1. Quartile               -0.002247
## 3. Quartile                0.003588
## Mean                       0.000605
## Median                     0.000324
## Sum                        0.212899
## SE Mean                    0.000269
## LCL Mean                   0.000076
## UCL Mean                   0.001133
## Variance                   0.000025
## Stdev                      0.005040
## Skewness                  -0.064426
## Kurtosis                   0.967430
```

Ao realizar as estatísticas básicas, percebi um problema. O cálculo do retorno feito pelo pacote `GetHFData` é o cálculo de retorno simples, o que torna a soma dos retornos inadequada. Para que a soma dos retornos seja correta, o cálculo do retorno deveria ser em logarítmo, o que possibilita a soma dos retornos. Portanto, para trabalhamors com os retornos dos ativos, devemos calcular o logarítmo do retorno. Conforme Tsay (2012) será utilizado o retorno ponderado pois leva em conta o fracionamento das ações, se ocorridos no período estudado (ainda não foi feito, os testes posteriores foram feitos com o retorno apresentado pelo pacote).

4.1 Teste de normalidade

Para testar a normalidade usarei o teste proporto por Jarque and Bera (1987).

```
normalTest(PETR4_data$period.ret, method = "jb")
```

```
##
## Title:
##  Jarque - Bera Normalality Test
##
## Test Results:
##   STATISTIC:
##     X-squared: 29.1554
##   P VALUE:
##     Asymptotic p Value: 4.666e-07
##
## Description:
##  Mon Nov 07 17:39:55 2016 by user: Lucca
```

```
normalTest(BVMF3_data$period.ret, method = "jb")
```

```
##
## Title:
##  Jarque - Bera Normalality Test
##
## Test Results:
##   STATISTIC:
##     X-squared: 116.6585
##   P VALUE:
##     Asymptotic p Value: < 2.2e-16
```

```
##
## Description:
##  Mon Nov 07 17:39:56 2016 by user: Lucca
```

```
normalTest(ITUB4_data$period.ret, method = "jb")
```

```
##
## Title:
##  Jarque - Bera Normalality Test
##
## Test Results:
##   STATISTIC:
##     X-squared: 14.6224
##   P VALUE:
##     Asymptotic p Value: 0.000668
##
## Description:
##  Mon Nov 07 17:39:56 2016 by user: Lucca
```

Para os três ativos, o teste de normalidade é rejeitado, pois os valores-p são menores que o nível de significância, inclusive de 1%. Portanto, não podemos afiramr que a série possui distribuição normal. Porém, o que realmente interessa é se o termo de erro da sequência é um ruído branco. Para descobrirmos isso, é preciso identificar o tipo e a ordem da série e depois realizar testes sobre o resíduo da equação de diferenças estocástica estimada.

5 Visualização de dados financeiros

Para sabermos como se comportam os dados de interesse podemos plotar o histograma dos dados e analizar o formato obtido.

```
par(mfrow = c(1, 3))
histo_PETR4 <- hist(PETR4_data$period.ret, main = "Retorno PETR4", nclass = 30)
histo_BVMF3 <- hist(BVMF3_data$period.ret, main = "Retorno BVMF3", nclass = 30)
histo_ITUB4 <- hist(ITUB4_data$period.ret, main = "Retorno ITUB4", nclass = 30)
```

Podemos ver que o formato dos histogramas se assemelham à uma distribuição normal, porém, como já apresentado pelo teste de normalidade, os dados não possuem distribuição normal. Outra forma de visualizar os dados é estimar sua densidade empírica e comparar com a densidade de uma distribuição normal. Isto é feito por meio de um método de suavização não paramétrico (Tsay 2012). Esta densidade empírica pode ser vista como uma versão refinada do histograma.

A densidade normal para cada ativo foi construída usando suas respectivas média e desvio padrão. As Figuras 2 e 3 fornecem uma referência visual sobre a hipótese de normalidade dos retornos. AS três densidades estimadas possuem um pico maior e caudas mais longas que a densidade normal. Conforme Tsay (2012) isto é normal para dados financeiros e, em geral, a diferença entre a linha pontilhada e a linha sólida não é normalmente distribuída. O que é consistente com o teste de normalidade realizado anteriormente.

```
par(mfrow = c(1, 3))
# para estimar as densidades dos retornos
dens_PETR4 <- density(PETR4_data$period.ret)
dens_BVMF3 <- density(BVMF3_data$period.ret)
dens_ITUB4 <- density(ITUB4_data$period.ret)
range(PETR4_data$period.ret)
```

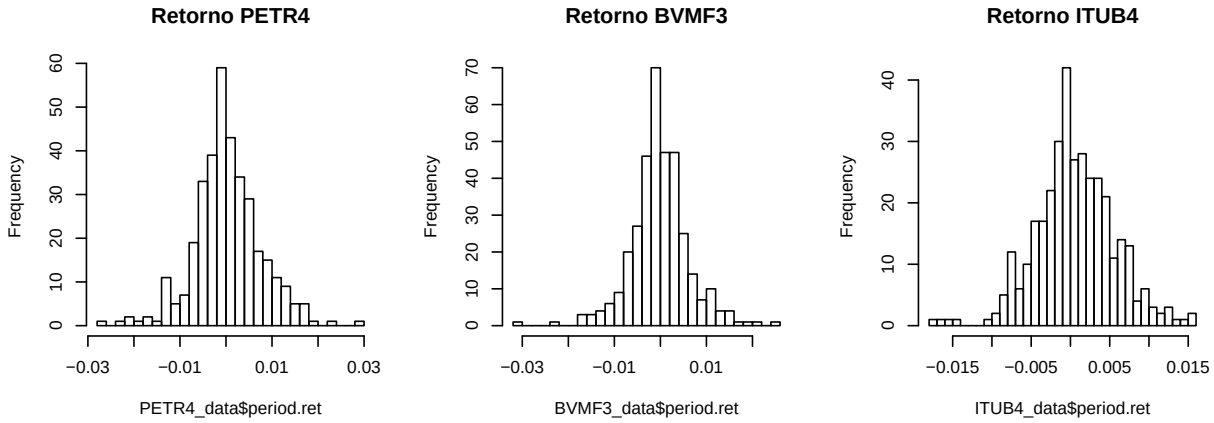


Figura 2: Histogramas dos ativos

```
## [1] -0.02694301  0.02898551
```

```
range(BVMF3_data$period.ret)
```

```
## [1] -0.03061728  0.02594595
```

```
range(ITUB4_data$period.ret)
```

```
## [1] -0.01796231  0.01573034
```

```
seq_x <- seq(-0.1, 0.1, 0.001)  # Cria uma sequência com incremento de 0.001
# Criando uma densidade normal
norm_PETR4 <- dnorm(seq_x, mean(PETR4_data$period.ret), stdev(PETR4_data$period.ret))
norm_BVMF3 <- dnorm(seq_x, mean(BVMF3_data$period.ret), stdev(BVMF3_data$period.ret))
norm_ITUB4 <- dnorm(seq_x, mean(ITUB4_data$period.ret), stdev(ITUB4_data$period.ret))
plot(dens_PETR4$x, dens_PETR4$y, xlab = "retorno PETR4", ylab = "densidade",
    type = "l")
lines(seq_x, norm_PETR4, lty = 2)
plot(dens_BVMF3$x, dens_BVMF3$y, xlab = "retorno BVMF3", ylab = "densidade",
    type = "l")
lines(seq_x, norm_BVMF3, lty = 2)
plot(dens_ITUB4$x, dens_ITUB4$y, xlab = "retorno ITUB4", ylab = "densidade",
    type = "l")
lines(seq_x, norm_ITUB4, lty = 2)
```

Outra forma interessante de visualizar os dados é o gráfico de barras, porém é necessário os preços de abertura e fechamento e o preço máximo e preço mínimo, o que não está disponível na amostra utilizada. Entretanto, podemos apresentar o grafico de média móvel dos preços. Par aisso énecessário baixar o script `ma.R` que realiza a suavização em http://faculty.chicagobooth.edu/ruey.tsay/teaching/introTS/ma.R.

```
source("ma.R")  # compila o script
preco_PETR4 <- as.numeric(PETR4_data$weighted.price)
preco_BVMF3 <- as.numeric(BVMF3_data$weighted.price)
```

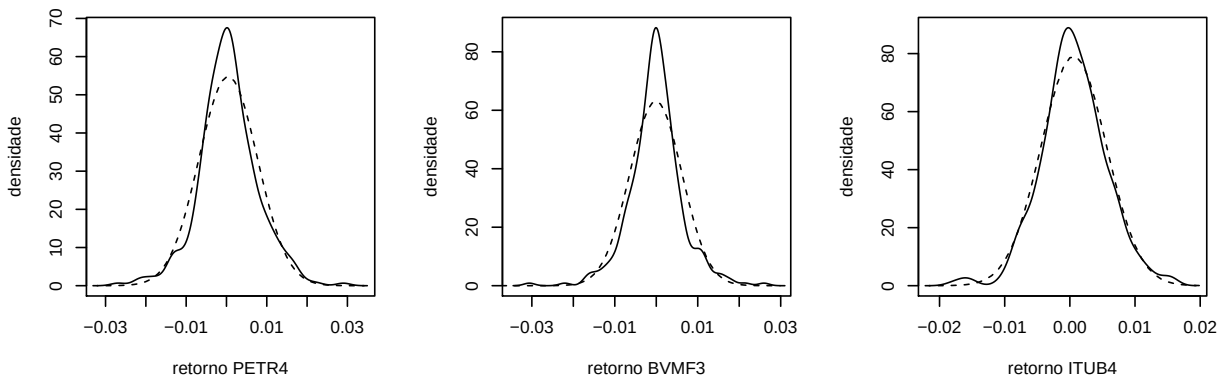
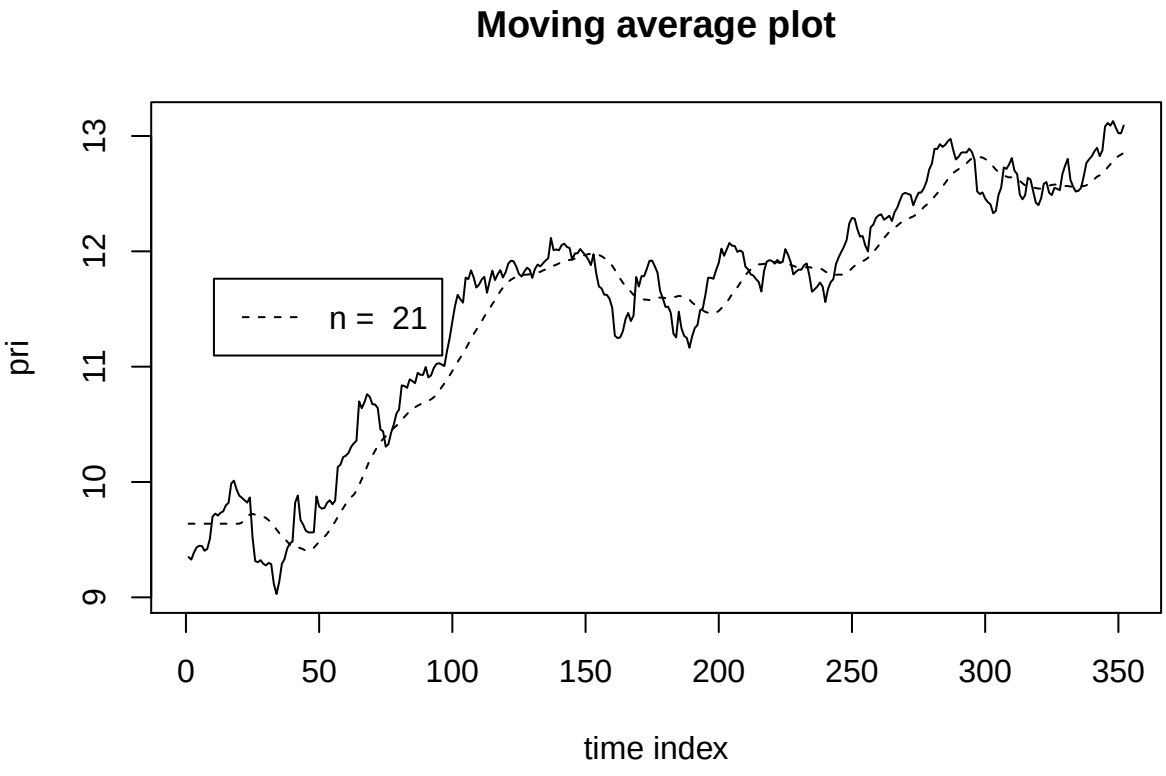


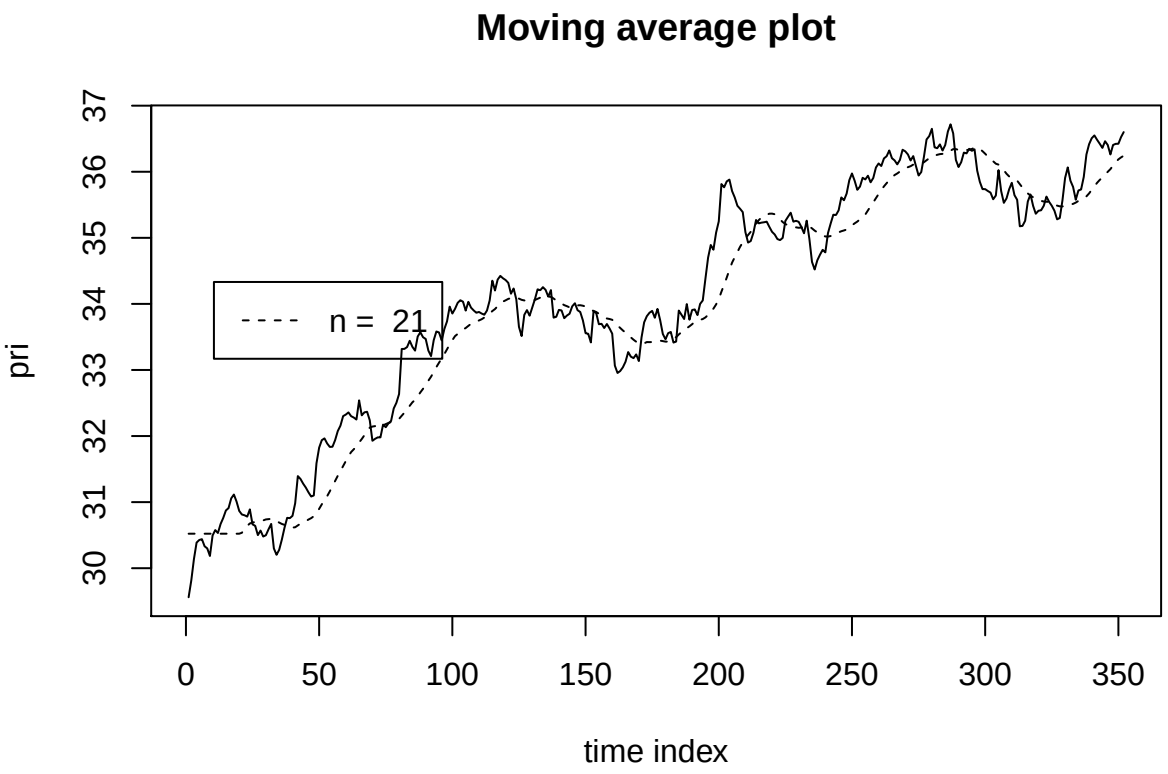
Figura 3: Densidade empírica dos ativos (linha sólida) e densidade normal (linha pontilhada)

```
preco_ITUB4 <- as.numeric(ITUB4_data$weighted.price)
par(mfrow = c(1, 3))
ma_PETR4 <- ma(preco_PETR4, 21)
```




```
ma_BVMF3 <- ma(preco_BVMF3, 21)
```


```
ma_ITUB4 <- ma(preco_ITUB4, 21)
```



FOram usadas 21 defasagens. O número de defasagens foi escolhido arbitrariamente e remete à uma média de três dias anteriores, já que a frequência de dados é por hora e o pregão fica aberto entre 10:00 hrs e 17:55 hrs.

Para encontrar a matriz de correlação do retorno dos ativos primeiramente contruiremos uma matriz com os três retornos.

```
matriz_retornos <- data.frame(PETR4_data$period.ret, BVMF3_data$period.ret,
    ITUB4_data$period.ret)
cor(matriz_retornos)
```

```
##                       PETR4_data.period.ret BVMF3_data.period.ret
## PETR4_data.period.ret             1.0000000             0.3191739
## BVMF3_data.period.ret             0.3191739             1.0000000
## ITUB4_data.period.ret             0.4650334             0.3071115
##                       ITUB4_data.period.ret
## PETR4_data.period.ret             0.4650334
## BVMF3_data.period.ret             0.3071115
## ITUB4_data.period.ret             1.0000000
```

Referências

Jarque, Carlos M., and Anil K. Bera. 1987. "A Test for Normality of Observations and Regression Residuals." *International Statistical Review / Revue Internationale de Statistique* 55 (2): 163–72. doi:10.2307/1403192.

Perlin, Marcelo. 2016. *GetHFData: Download and Aggregate High Frequency Trading Data from Bovespa*. https://CRAN.R-project.org/package=GetHFData.

Perlin, Marcelo, and Henrique Ramos. 2016. "GetHFData: A R Package for Downloading and Aggregating High Frequency Trading Data from Bovespa." SSRN Scholarly Paper ID 2824058. Rochester, NY: Social Science Research Network. https://papers.ssrn.com/abstract=2824058.

Ryan, Jeffrey A. 2016. *Quantmod: Quantitative Financial Modelling Framework*. https://CRAN.R-project.org/package=quantmod.

Tsay, Ruey S. 2012. *An Introduction to Analysis of Financial Data with R*. 1 edition. Hoboken, N.J: Wiley.