Séries Temporais - Trabalho Final

Mateus Lee Yu (RA: 236235)

Introdução

O banco Itaú Unibanco é um dos maiores bancos da América Latina, essa instituição financeira desempenha um papel significativo na economia brasileira. As ações da Itaú são amplamente negociadas na bolsa de valores por investidores individuais e institucionais.

A volatilidade de ações é uma métrica do risco associado a um ativo financeiro, ela é utilizada por investidores para a tomada de decisões de compra ou venda com o objetivo de maximizar o lucro e minimizar o prejuízo. A volatilidade depende de diversos fatores, como eventos econômicas e políticas dos países, e até de mudanças internas da empresa.

Nesse trabalho vamos modelar a volatilidade do banco Itaú Unibanco Holding S.A. (ITUB4.SA) por um modelo ARMA-GARCH, utilizado dados disponibilizados pelo *Yahoo! Finance* a partir do dia 01 de Janeiro de 2000 até 16 de Junho de 2023. Logo após a modelagem, colocaremos o modelo em produção com o auxílio do *GitHub Actions*. E ao final iremos realizar uma breve discussão dos resultados obtidos.

Análise dos Dados

Para a modelagem dos dados precisamos carregar dois pacotes disponíveis no R, o `yfR` que auxilia na coleta de dados, e `rugarch` para a modelagem.

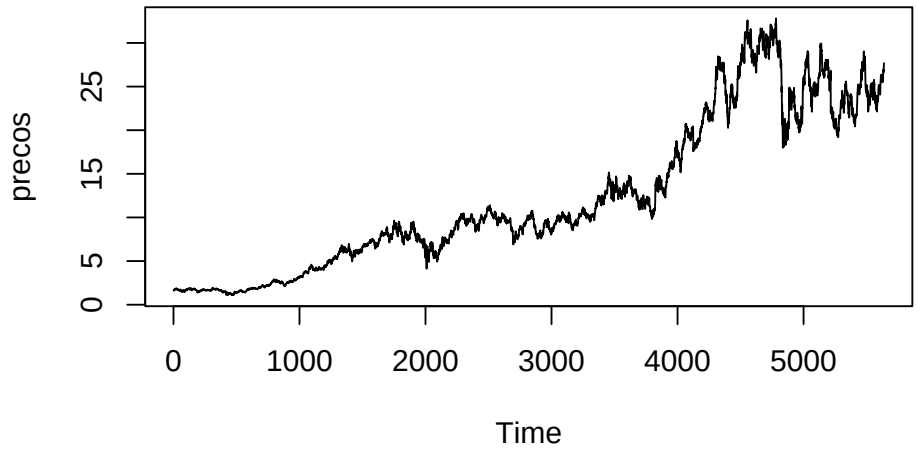
```
library(yfR)
library(rugarch)
```

Utilizando a função `yf_get` do pacote `yfR`, vamos armazenar todos os dados necessários para a nossa modelagem, além disso vamos guardar os preços das ações e os retornos em vetores separados.

```
dados <- yf_get(tickers = "ITUB4.SA",
                first_date = "2000-01-01",
                last_date = "2023-06-16")
precos <- as.ts(dados$price_adjusted, Daily, as.Date(dados$ref_date))
retornos <- as.ts(dados$ret_adjusted_prices)[-1]
```

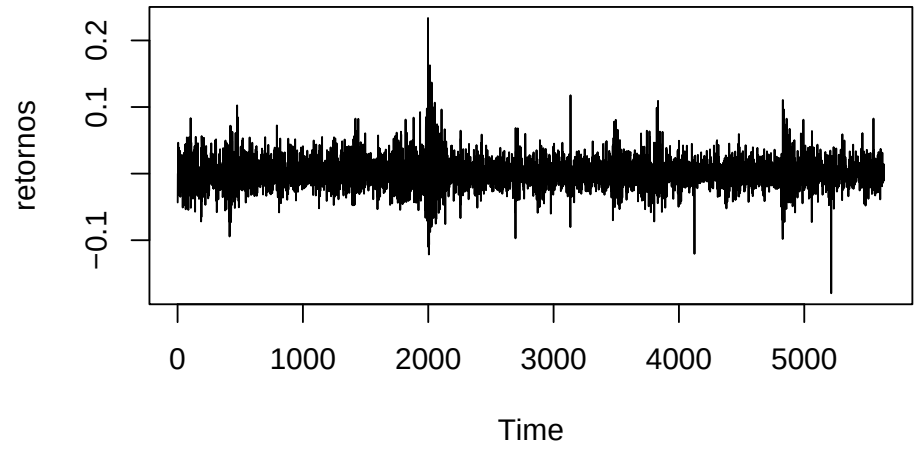
Primeiramente, vamos analisar os gráficos dos preços e dos retornos das ações.

```
ts.plot(precos)
```



Podemos notar nitidamente que o valor do preço possui tendência estocástica, no início os preços são menores do que 5 reais, porém com o passar do tempo ela foi aumentando até se estabilizar próximo de 10 reais, mas perto do final do gráfico os preços aumentam de forma drástica com bastante flutuações.

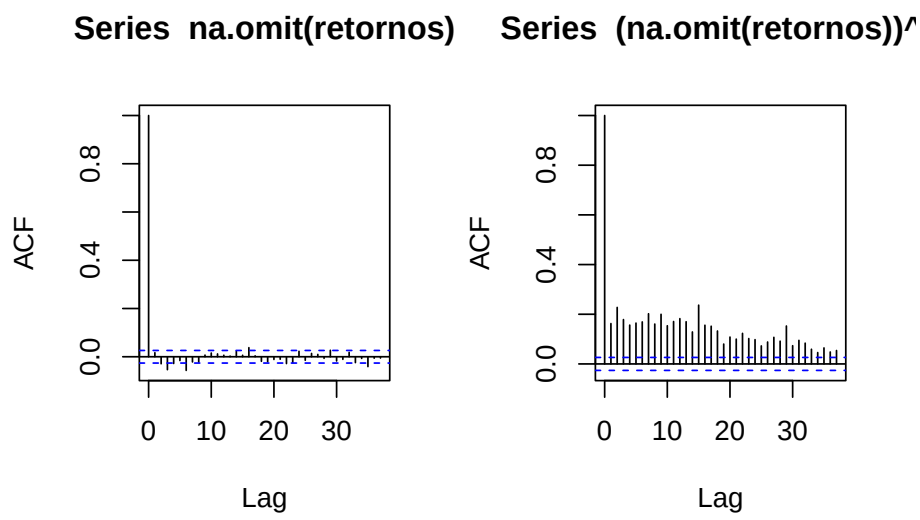
```
ts.plot(retornos)
```



Já no gráfico dos retornos conseguimos observar a presença de flutuações na volatilidade, ou seja, podemos tentar modelar os nossos dados assumindo que a variância é condicional nas observações passadas.

Como estamos trabalhando com modelos GARCH, precisamos verificar os gráficos de autocorrelação dos retornos e dos retornos ao quadrado.

```
op <- par(mfrow = c(1, 2))
acf(na.omit(retornos))
acf((na.omit(retornos))^2)
```




```
par(op)
```

Perceba que os retornos são não correlacionados, pois as maioria das autocorrelações estão dentro do intervalo de confiança e decaem para zero como uma função trigonométrica. Já o gráfico dos retornos ao quadrado, podemos notar claramente uma correlação entre os dados por conta da quantidade de autocorrelações fora da banda de confiança.

Para obter indicações formais de que a série de retornos apresenta heterocedasticidade condicional, vamos realizar o teste de Ljung-Box.

```
Box.test(retornos^2, type = "Ljung-Box", lag = 10)
```

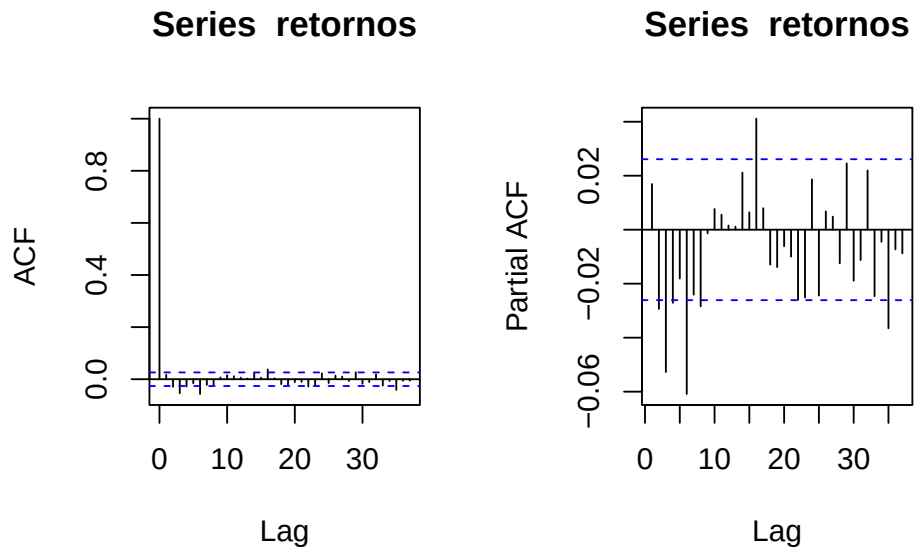
```
    Box-Ljung test

data:  retornos^2
X-squared = 1808, df = 10, p-value < 2.2e-16
```

Obtemos um p-valor muito pequeno, logo rejeitamos a hipótese nula de que os retornos ao quadrado possuem autocorrelações nulas a favor da hipótese alternativa de que a série apresenta autocorrelação.

Modelagem dos Dados

Identificado a necessidade de modelar a variância condicional, iremos indentificar os parâmetros do componente ARMA da série, para isso precisamos do gráfico de autocorrelações e de autocorrelações parciais.

```
op <- par(mfrow = c(1, 2))
acf(retornos)
pacf(retornos)
```




```
par(op)
```

Novamente o gráfico do autocorrelações do retorno decai como uma função trigonométrica, indicando que o componente MA é igual a zero, porém ao olharmos para as autocorrelações parciais notamos que há 2 faixas que se destacam e outras 3 faixas foras, porém pertos da banda de confiança, então a parte AR do modelo pode ser um valor de 1 até 5.

Já a parte GARCH do modelo, foi escolhida os parâmetros mais simples possível, ou seja, utilizaremos o sGARCH(1, 1). Portanto, todos os modelos a serem testados são:

- ARMA(1, 0) - GARCH(1, 1);
- ARMA(2, 0) - GARCH(1, 1);
- ARMA(3, 0) - GARCH(1, 1);
- ARMA(4, 0) - GARCH(1, 1);
- ARMA(5, 0) - GARCH(1, 1).

Além disso, através do conhecimento adquirido na sala de aula, é mais vantajoso modelar a volatilidade com a distribuição t no nosso contexto.

Utilizando o pacote `rugarch`, especificamos os parâmetros do nosso modelo com a função `ugarchspec` e ajustamos o nosso modelo com `ugarchfit`.

```
spec_1 <- ugarchspec(mean.model = list(armaOrder = c(1, 0),
                                       include.mean = FALSE),
                     variance.model = list(model = "sGARCH",
                                           garchOrder = c(1, 1)),
                     distribution = "std")
fit_1 <- ugarchfit(spec_1, retornos, solver = "hybrid")

spec_2 <- ugarchspec(mean.model = list(armaOrder = c(2, 0),
                                       include.mean = FALSE),
```

```
                    variance.model = list(model = "sGARCH",
                                          garchOrder = c(1, 1)),
                    distribution = "std")
fit_2 <- ugarchfit(spec_2, retornos, solver = "hybrid")

spec_3 <- ugarchspec(mean.model = list(armaOrder = c(3, 0),
                                       include.mean = FALSE),
                    variance.model = list(model = "sGARCH",
                                          garchOrder = c(1, 1)),
                    distribution = "std")
fit_3 <- ugarchfit(spec_3, retornos, solver = "hybrid")

spec_4 <- ugarchspec(mean.model = list(armaOrder = c(4, 0),
                                       include.mean = FALSE),
                    variance.model = list(model = "sGARCH",
                                          garchOrder = c(1, 1)),
                    distribution = "std")
fit_4 <- ugarchfit(spec_4, retornos, solver = "hybrid")

spec_5 <- ugarchspec(mean.model = list(armaOrder = c(5, 0),
                                       include.mean = FALSE),
                    variance.model = list(model = "sGARCH",
                                          garchOrder = c(1, 1)),
                    distribution = "std")
fit_5 <- ugarchfit(spec_5, retornos, solver = "hybrid")
```

Para selecionar o modelo mais adequado será utilizada a validação cruzada do tipo *rolling window*, e o critério de decisão será o erro quadrático médio, a implementação da função está disponível abaixo.

```
validacao_cruzada <- function(fit, n_teste) {
  serie <- fit@model[["modeldata"]][["data"]]
  n_observacoes_totais <- length(serie)
  n_observacoes_teste <- n_teste
  n_observacoes_treinamento <- n_observacoes_totais - n_observacoes_teste
  erro_quadratico_medio <- 0
  for (i in 1:n_observacoes_teste) {
    janela <- serie[i:(i + n_observacoes_treinamento - 1)]
    erro <- ugarchforecast(fit, n.ahead = 1)@forecast[["seriesFor"]][1] -
            serie[i + n_observacoes_treinamento]
    erro_quadratico_medio <- erro_quadratico_medio + erro^2
  }
```

```
  erro_quadratico_medio <- erro_quadratico_medio/n_observacoes_teste
  return(erro_quadratico_medio)
}
```

Aplicando a função `validacao_cruzada` com 30 dias de teste para cada um dos modelos, obtemos os seguinte valores.

```
validacao_cruzada(fit_1, 30)
```

```
[1] 0.0001761546
```

```
validacao_cruzada(fit_2, 30)
```

```
[1] 0.000180056
```

```
validacao_cruzada(fit_3, 30)
```

```
[1] 0.0001839475
```

```
validacao_cruzada(fit_4, 30)
```

```
[1] 0.0001815668
```

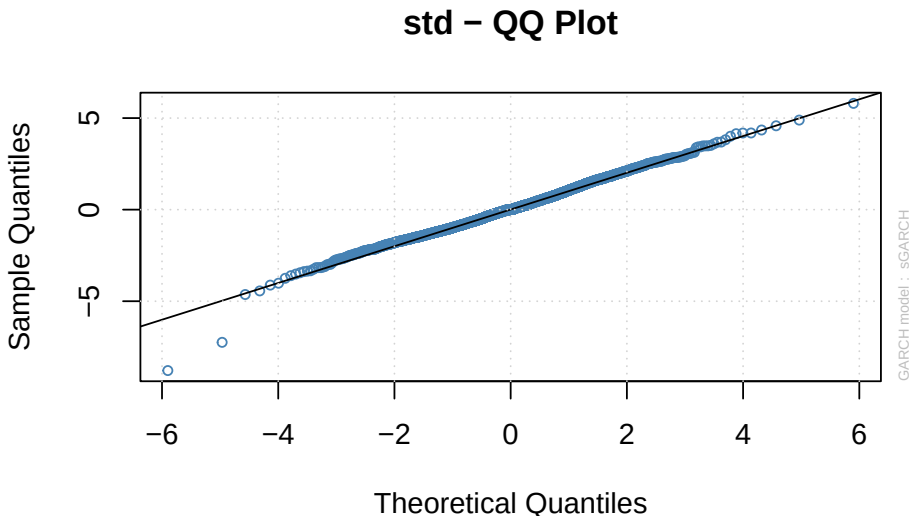
```
validacao_cruzada(fit_5, 30)
```

```
[1] 0.0001833483
```

O modelo que possui o menor erro quadrático médio é o ARMA(1, 0) - GARCH(1, 1), logo utilizaremos ele para as previsões no nosso modelo em produção.

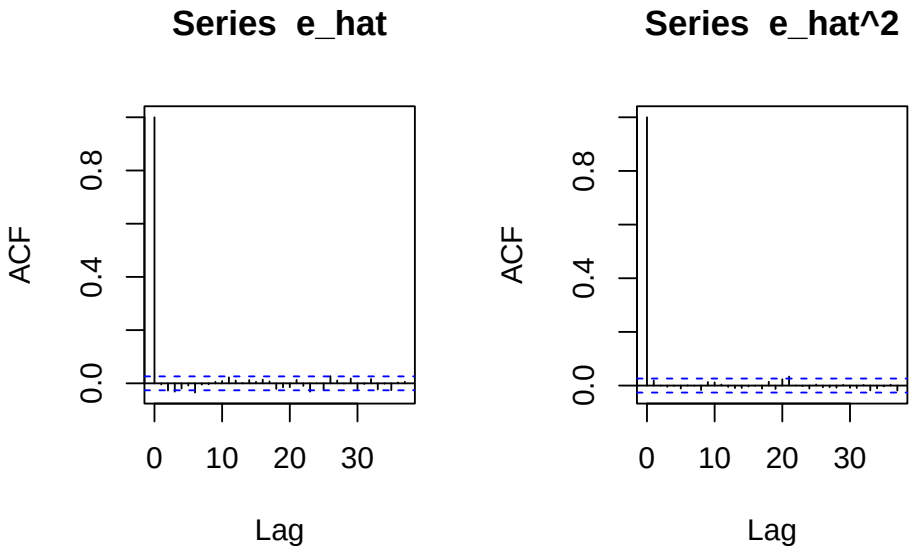
Mas antes, precisamos realizar o diagnóstico desse modelo. Vamos analisar o gráfico QQ Plot dos resíduos, autocorrelações dos resíduos e autocorrelações dos resíduos ao quadrado.

```
plot(fit_1, which = 9)
```



Podemos notar que praticamente todos os resíduos estão em cima da linha, com a exceção de dois pontos presentes no canto inferior esquerdo.

```
e_hat = fit_1@fit$residuals/fit_1@fit$sigma
op = par(mfrow = c(1,2))
acf(e_hat)
acf(e_hat^2)
```




```
par(op)
```

Nos gráficos acima podemos observar que as autocorrelação dos resíduos e dos resíduos ao quadrados estão todas dentro do intervalo de confiança.

Portanto, podemos concluir que o modelo ARMA(1, 0) - GARCH(1, 1) é adequado.

Colocando o Modelo em Produção

A plataforma escolhida para colocar o modelo em produção foi o GitHub, com o auxílio da ferramenta Actions foi possível a execução diária do script abaixo.

```
library(scales)
library(yfR)
library(rugarch)

data_hoje <- Sys.Date()

dados <- yf_get(tickers = "ITUB4.SA",
                first_date = "2000-01-01",
                last_date = data_hoje)

retornos <- as.ts(dados$ret_adjusted_prices)[-1]

spec_1 <- ugarchspec(mean.model = list(armaOrder = c(1, 0),
                                       include.mean = FALSE),
                     variance.model = list(model = "sGARCH",
                                           garchOrder = c(1, 1)),
                     distribution = "std")
fit_1 <- ugarchfit(spec_1, retornos, solver = "hybrid")

retorno_amanha <- ugarchforecast(fit_1, n.ahead = 1)@forecast[["seriesFor"]][1]

retorno_hoje <- tail(yf_get(tickers = "ITUB4.SA",
                            first_date = data_hoje-7,
                            last_date = data_hoje)$ret_adjusted_prices,
                     n = 1)

linha <- data.frame(data_hoje, retorno_hoje, retorno_amanha)

write.table(linha,
            file = "retornos.csv",
            append = TRUE,
            sep = ",",
            row.names = FALSE,
            col.names = FALSE)
```

E o arquivo .yml com as instruções do *workflow* utilizado está logo abaixo.

```
name: automatizacao
```

```
on:
  schedule:
    - cron: "0 15 * * *"

jobs:
  scrape:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: r-lib/actions/setup-r@v2

      - name: instala pacotes
        run: |
          sudo apt install libcurl4-openssl-dev
          R -e 'install.packages("scales")'
          R -e 'install.packages("yfR")'
          R -e 'install.packages("rugarch")'

      - name: executa script
        run: Rscript main.R

      - name: commit
        run: |
          git config --local user.name github-actions
          git config --local user.email "actions@github.com"
          git add --all
          git commit -am "retornos.csv atualizado"
          git push
        env:
          REPO_KEY: ${{secrets.GITHUB_TOKEN}}
          username: github-actions
```

Conclusão

O valor dos retornos previstos estão na tabela abaixo.

data_hoje	retorno_hoje	retorno_amanha
2023-06-21	0.0136126137094483	0.0002120896239206
2023-06-22	0.00973576788654973	0.000151654580909666
2023-06-23	-0.0096418966190801	-0.000148934686993502
2023-06-24	-0.00312935159498051	-4.84573672603491e-05
2023-06-25	-0.00312935159498051	-4.84396150845952e-05

data_hoje	retorno_hoje	retorno_amanha
2023-06-26	-0.00312935159498051	-4.84474101619013e-05
2023-06-27	-0.00139521853578295	-2.1610363540446e-05

Podemos notar que o modelo não conseguiu prever corretamente os valores com uma margem pequena, porém foi possível identificar se a volatilidade sobe ou desce no dia seguinte.

Para uma análise futura poderíamos tentar outras combinações do modelo ARMA-GARCH, além disso seria interessante utilizar variantes do modelo GARCH, como iGARCH ou tGARCH.