Questão 1

Shai Vaz, Heron Goulart, Alexandre Almeida, João Pedro Pedrosa, Roberto Orenstein

2023-10-05

Questão 1: S&P 500

Importar Base

```
# importar índice S&P500
ativo <- quantmod::getSymbols(
  "^GSPC",
  src = "yahoo",
  auto.assign = FALSE,
  from = '2015-09-01',
  to = '2023-09-01',
  periodicity = "daily")
```

Série em nível

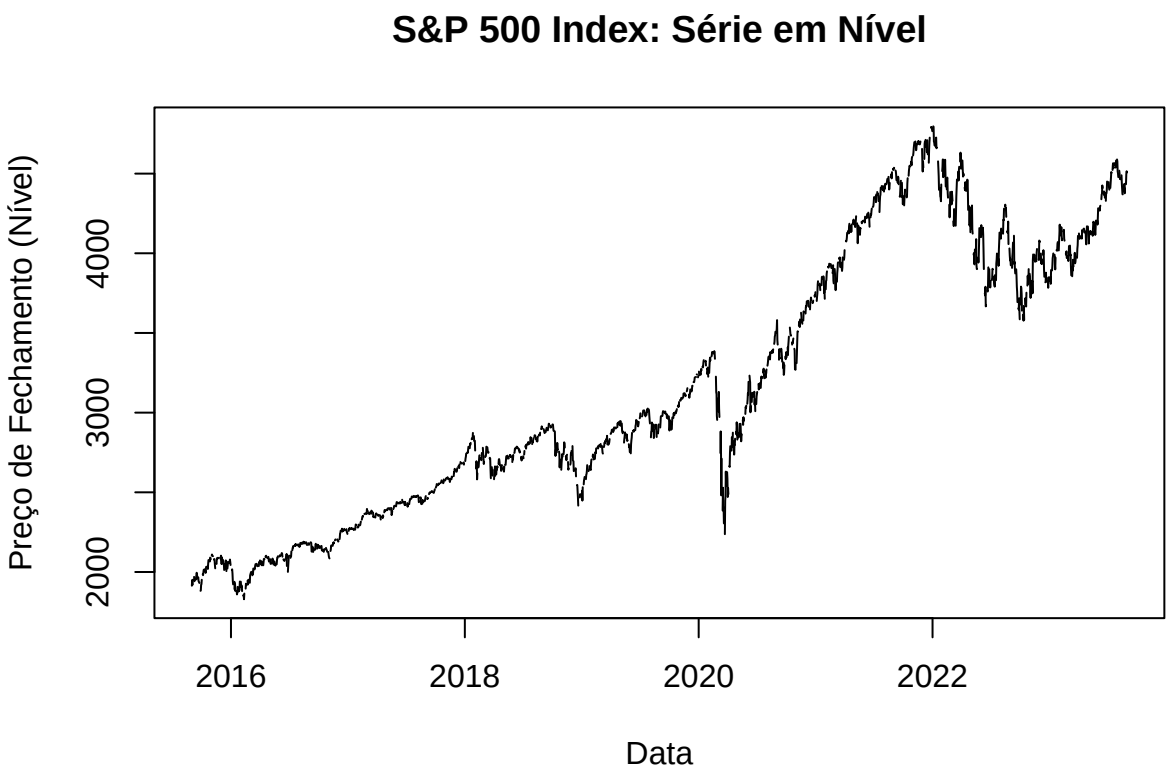
Selecionando os preços de fechamento, em nível, do índice.

```
close <- ativo[,"GSPC.Close"]
```

Os dados estão em `xts`, mas preferiremos por simplicidade trabalhar com o formato `ts`.

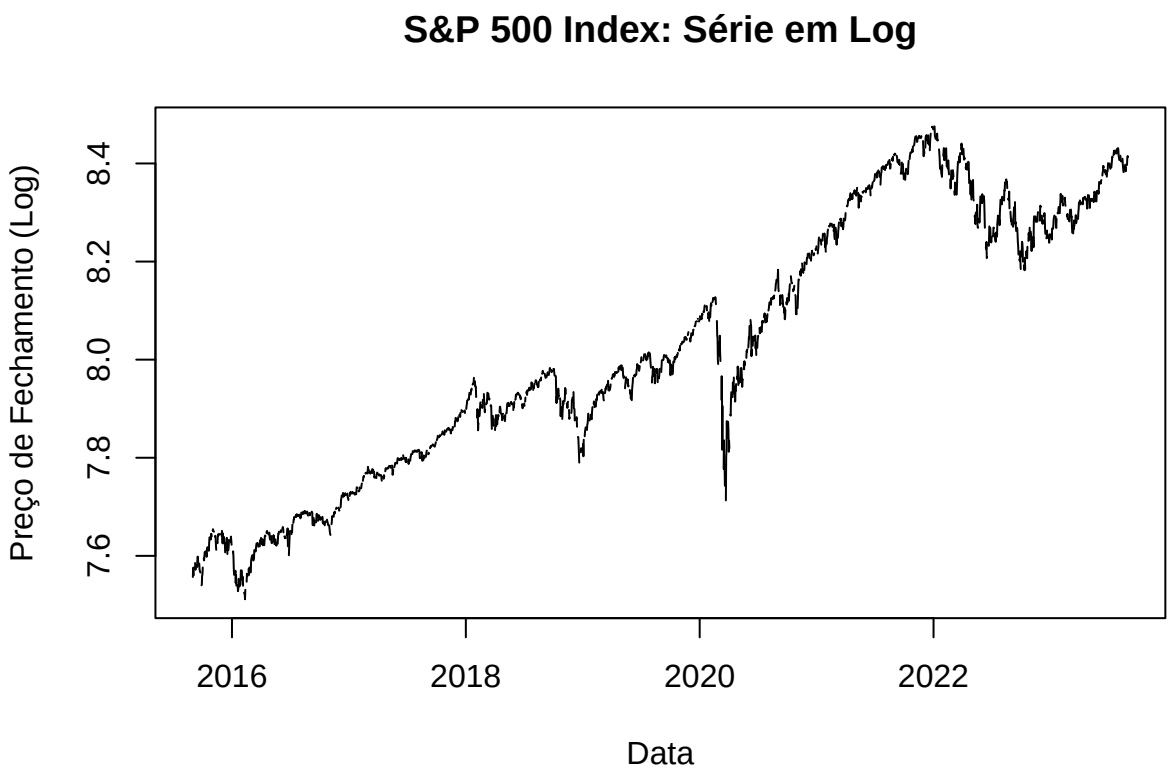
```
# ts (nível)
ts_close <- ts_ts(close)
```

```
plot(ts_close,
     xlab = "Data",
     ylab = "Preço de Fechamento (Nível)",
     main = "S&P 500 Index: Série em Nível")
```



Série em Log

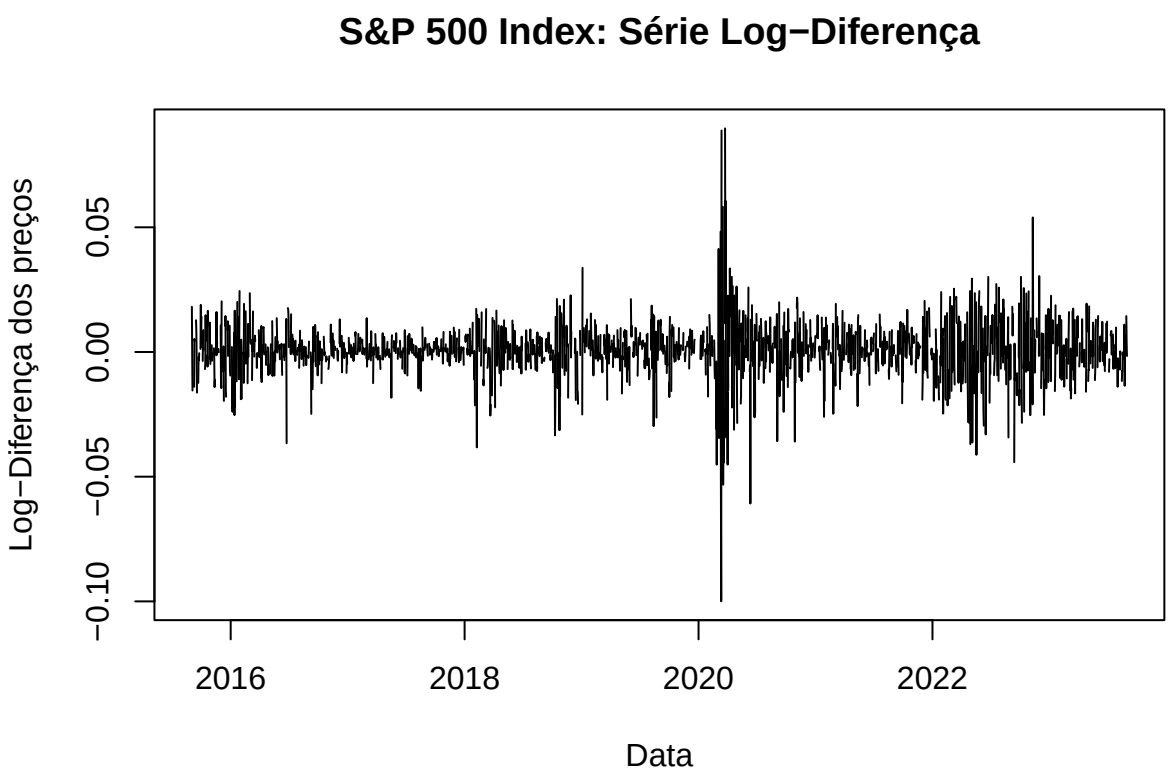
```
log_close <- log(close)
ts_log_close <- log(ts_close)
plot(ts_log_close,
     xlab = "Data",
     ylab = "Preço de Fechamento (Log)",
     main = "S&P 500 Index: Série em Log")
```



Série em Log-Dif

Tiraremos a primeira diferença da série em log.

```
dif_log_close <- diff(log_close)[-1,]
ts_dif_log_close <- diff(ts_log_close)
plot(ts_dif_log_close,
     xlab = "Data",
     ylab = "Log-Diferença dos preços",
     main = "S&P 500 Index: Série Log-Diferença"
     )
```



Teste de Phillips-Perron (PPT)

Para a série Log-Nível

```
tidy(ppt_log_close) %>%
  kable(
    col.names = c(
      "Statistic: Dickey-Fuller Z (alpha)",
      "P Value",
      "Parameter: Truncation lag",
      "Method",
      "Alternative Hypothesis"
    ),
    caption = "Teste Philips Perron: Série Log Nível"
  )
```

Table 1: Teste Philips Perron: Série Log Nível

Statistic: Dickey-Fuller Z (alpha)	P Value	Parameter: Truncation lag	Method	Alternative Hypothesis
-18.5791	0.0930023	8	Phillips-Perron Unit Root Test	stationary

Nesse caso, não rejeitamos a hipótese nula ao nível de significância 0.05. Há indícios portanto que a série não é estacionária e tem raíz unitária.

Para a série Log-Dif

```
ppt_dif_log_close <- pp.test(dif_log_close)
```

```
tidy(ppt_dif_log_close) %>%
  kable(
    col.names = c(
      "Statistic: Dickey-Fuller Z (alpha)",
      "P Value",
      "Parameter: Truncation lag",
      "Method",
      "Alternative Hypothesis"
    ),
    caption = "Teste Philips Perron"
  )
```

Table 2: Teste Philips Perron

Statistic: Dickey-Fuller Z (alpha)	P Value	Parameter: Truncation lag	Method	Alternative Hypothesis
-2404.682	0.01	8	Phillips-Perron Unit Root Test	stationary

Nesse caso, podemos rejeitar a hipótese nula e aceitar a hipótese alternativa ao nível de significância 0.05. Portanto, a série é estacionária.

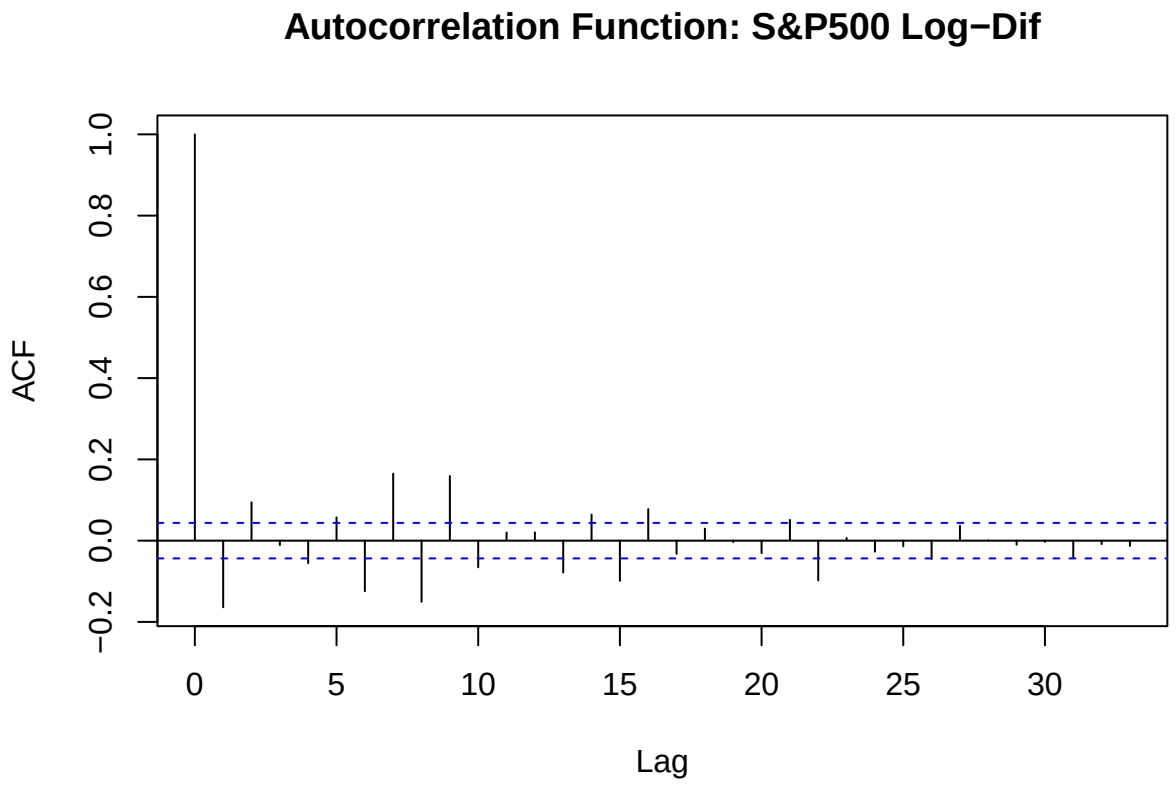
Modelando a média com um ARIMA(p,d,q)

Autocorrelation Functions

Primeiramente, vamos avaliar as funções de autocorrelação e autocorrelação parcial da nossa série escolhida, a série log-dif.

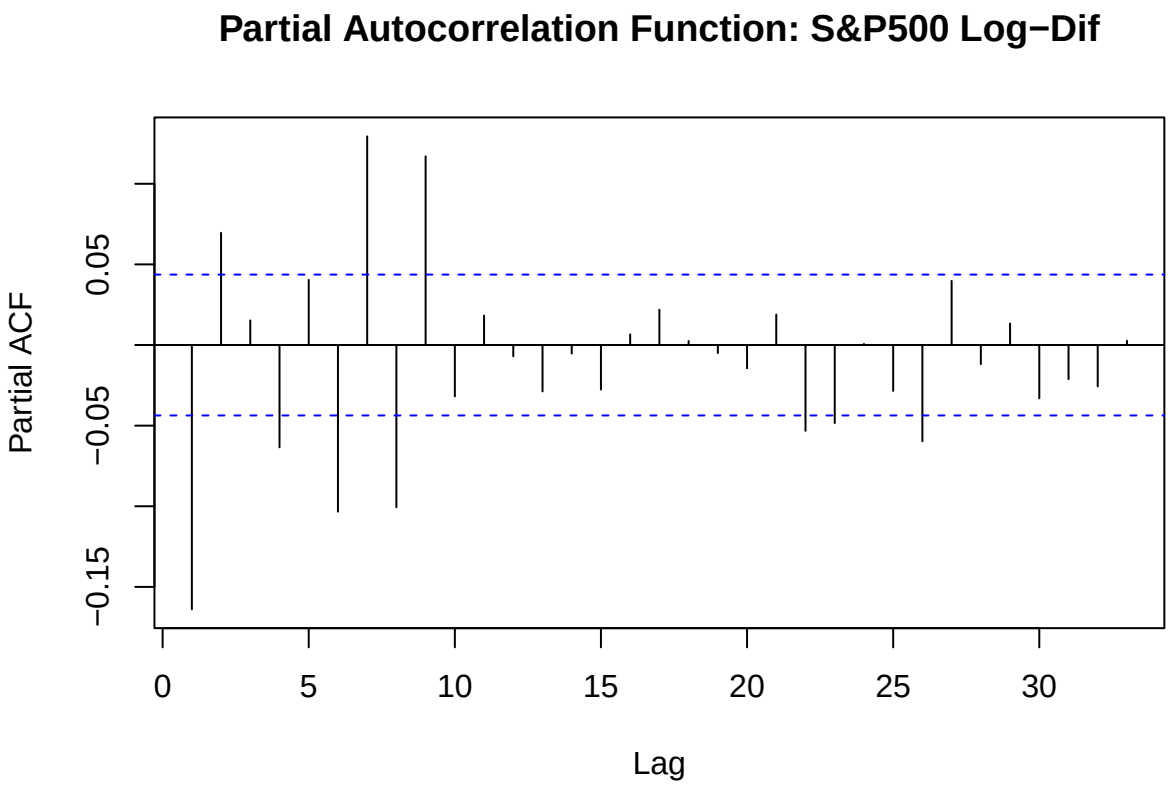
```
s1 <- ts(dif_log_close)
```

```
acf(s1,
    main = "Autocorrelation Function: S&P500 Log-Dif"
    )
```



Temos uma série com ACF que aparece truncada na segunda defasagem, indicando um componente MA(2).

```
pacf(
  s1,
  main = "Partial Autocorrelation Function: S&P500 Log-Dif"
  )
```



Temos uma série com PACF truncada na segunda defasagem, indicando um componente AR(2).

Críterios de Informação

Podemos verificar a decisão através dos critérios de informação. Utilizaremos a função `auto.arima`, que, de forma automática, calcula todos os modelos dentro de limites dados, calcula seus critérios de informação e escolhe o modelo que minimiza. Aqui, escolhemos os limites de 5 lags para cada parte do modelo (AR e MA). Primeiro, minimizaremos o BIC.

```
model_auto_bic <- auto.arima(
  max.p = 5,
  max.q = 5,
  s1,
  seasonal = FALSE,
  stepwise = FALSE,
  approximation = FALSE,
  ic = "bic"
  )
```

```
tidy(model_auto_bic) %>%
  kable(
    col.names = c(
      "Term",
      "Estimate",
      "Standard Error"
```

```
    ),
    caption = "BIC Minimizer Model"
  )
```

Table 3: BIC Minimizer Model

Term	Estimate	Standard Error
ar1	-1.7401133	0.0257552
ar2	-0.8770843	0.0258495
ma1	1.6310849	0.0354612
ma2	0.7419635	0.0358254

Agora, pelo critério do AIC.

```
model_auto_aic <- auto.arima(
  max.p = 5,
  max.q = 5,
  s1,
  seasonal = FALSE,
  stepwise = FALSE,
  approximation = FALSE,
  ic = "aic"
  )
```

```
tidy(model_auto_aic) %>%
  kable(
    col.names = c(
      "Term",
      "Estimate",
      "Standard Error"
    ),
    caption = "AIC Minimizer Model"
  )
```

Table 4: AIC Minimizer Model

Term	Estimate	Standard Error
ar1	-1.7390897	0.0259024
ar2	-0.8760746	0.0260392
ma1	1.6294213	0.0356304
ma2	0.7402674	0.0360618
intercept	0.0004274	0.0002373

Concluímos que, de fato, a escolha do modelo ARMA(2,2) para a média da série da Log-diferença é a ideal nesse caso, sustentado tanto pela análise das ACF e PACF quanto pelos critérios de informação.

ARMA(2,2)

Implementando o modelo escolhido ARMA(2,2).

```r
m202 <- arima(s1, order = c(2, 0, 2))
```

```r
stargazer(
  m202,
  header = FALSE,
  float = FALSE,
  dep.var.labels = "$\\Delta \\log (P)$"
)
```

	Dependent variable:
	$\Delta \log(P)$
ar1	-1.739^{***}
	(0.026)
ar2	-0.876^{***}
	(0.026)
ma1	1.629^{***}
	(0.036)
ma2	0.740^{***}
	(0.036)
intercept	0.0004^{*}
	(0.0002)
Observations	2,013
Log Likelihood	6,154.515
σ^2	0.0001
Akaike Inf. Crit.	$-12{,}297.030$
Note:	*p<0.1; **p<0.05; ***p<0.01

Análise de Resíduos

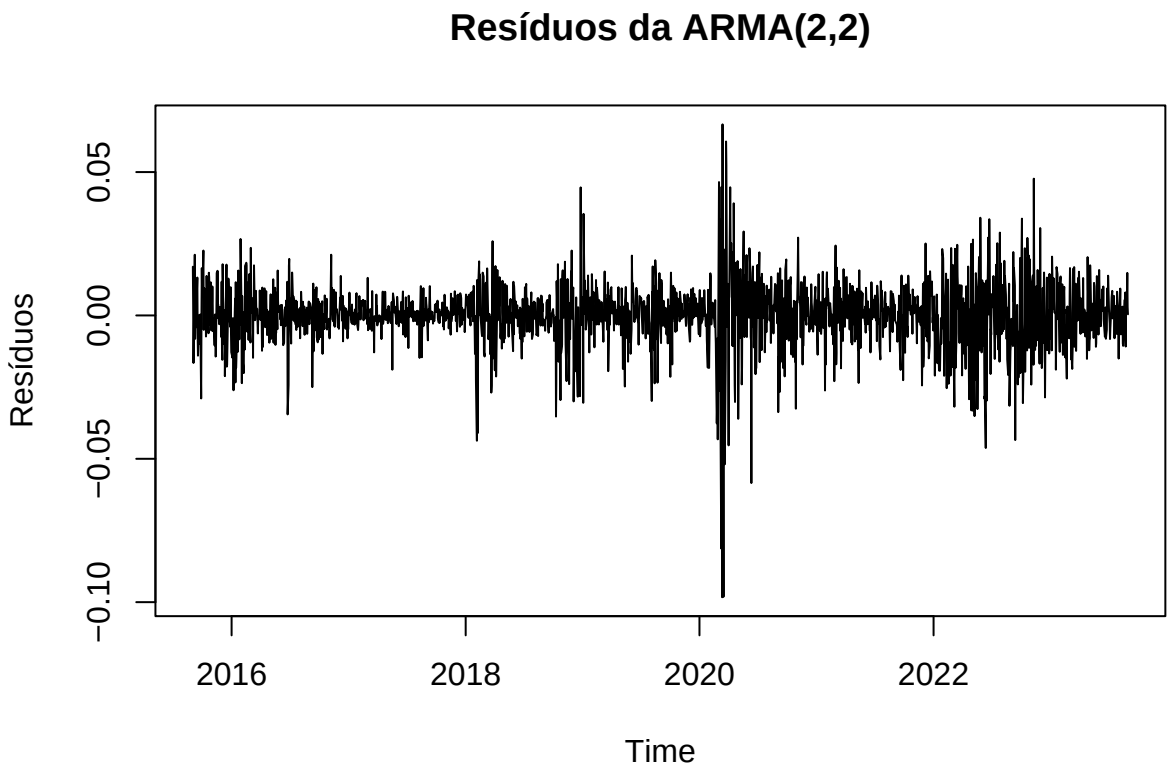
Série dos resíduos

Primeiro, coletamos a série dos resíduos.

```r
res_m202 <- residuals(m202)
```

Visualizando.

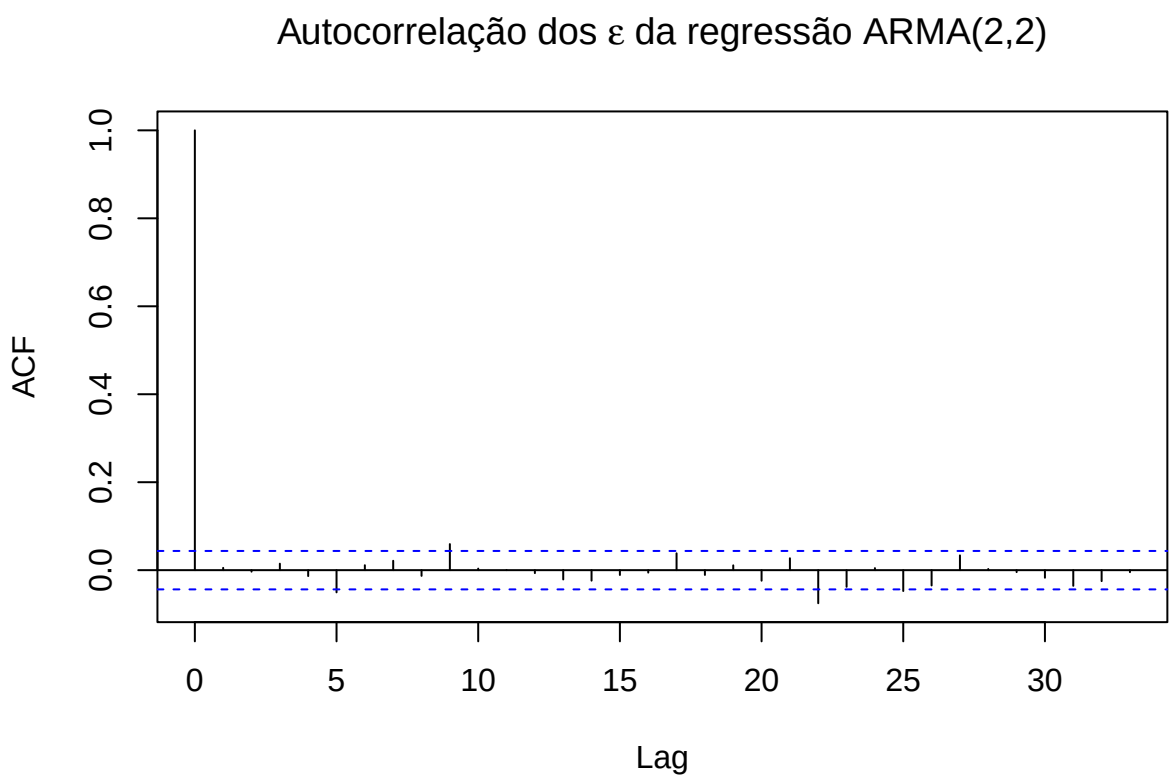
```r
datas <- index(dif_log_close)
plot(res_m202,
     xlab = "Time",
     ylab = "Resíduos",
     x = datas,
     type = "l",
     main = "Resíduos da ARMA(2,2)")
```



Autocorrelação dos resíduos

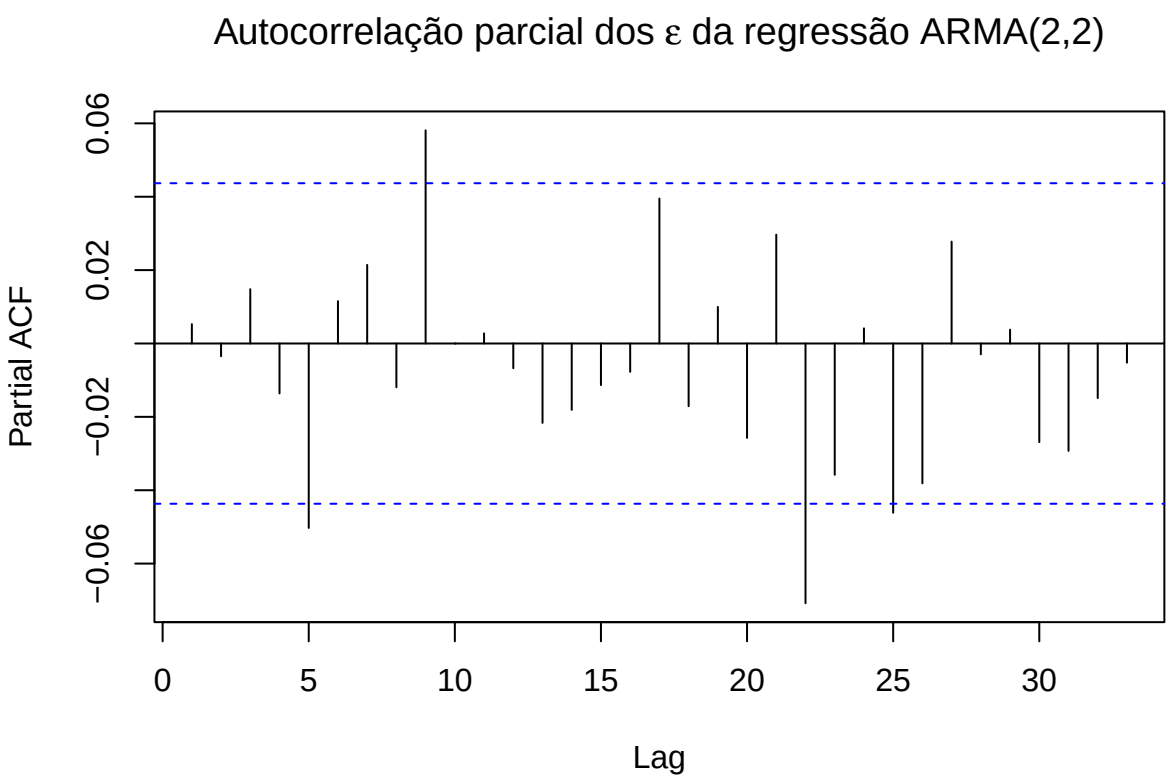
Quanto à ACF:

```
acf(res_m202,
    main = TeX(r"(Autocorrelação dos $\epsilon$ da regressão ARMA(2,2))"))
```



Quanto à PACF:

```
pacf(res_m202,
     main = TeX(r"(Autocorrelação parcial dos $\epsilon$ da regressão ARMA(2,2))"))
```

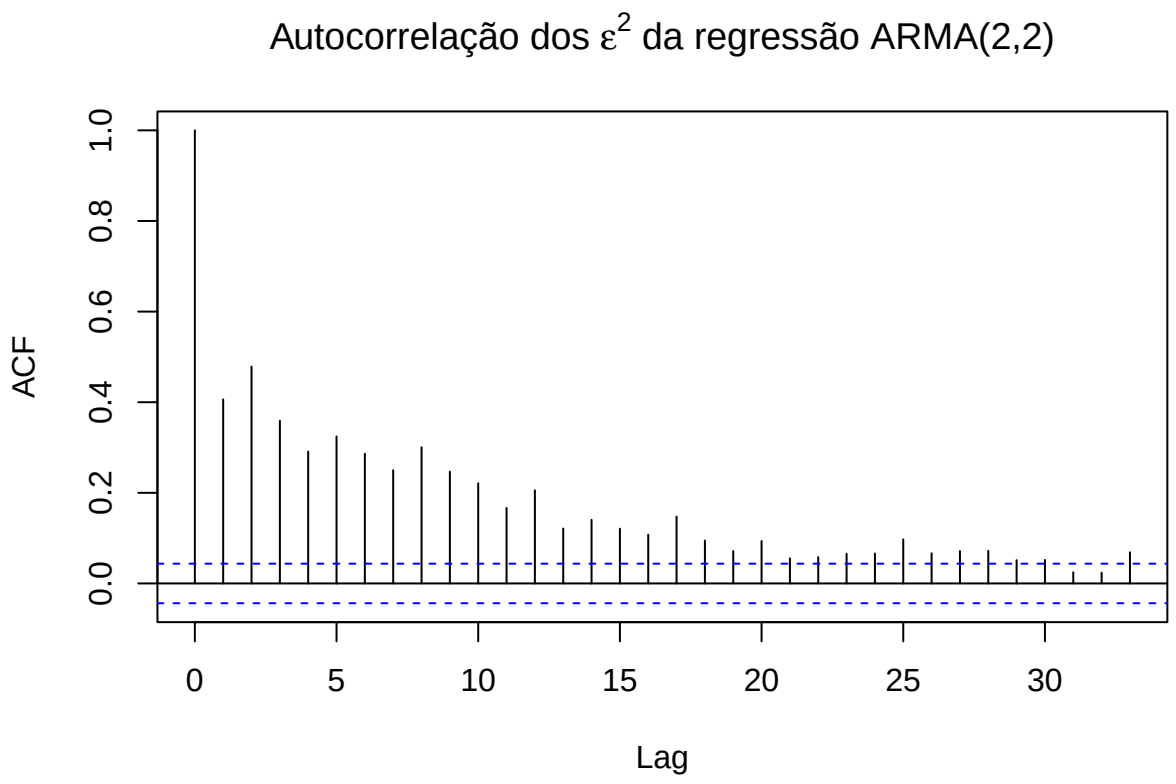


Podemos ver que, como esperado, os resíduos não exibem padrão claro de autocorrelação serial. Mas será que isso vale também para os quadrados dos resíduos, que estimam sua variância?

Autocorrelação dos quadrados dos resíduos

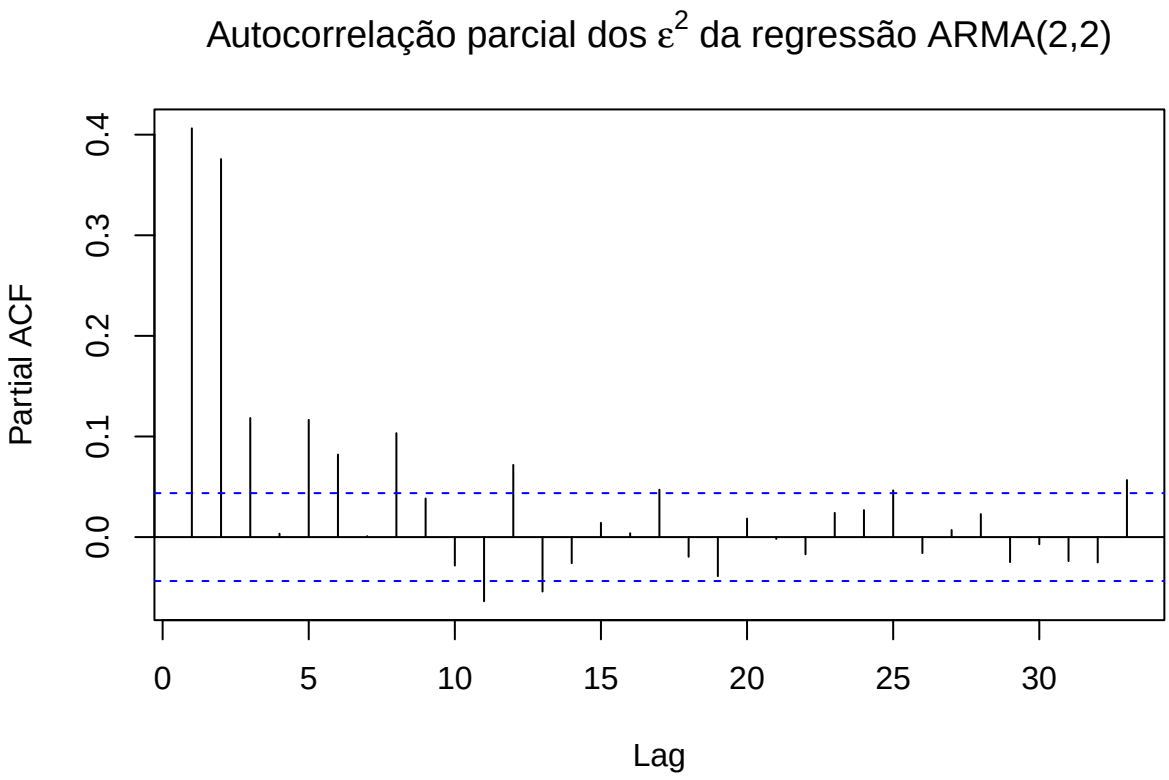
Quanto à ACF:

```
acf(res_m202^2,
    main = TeX(r"(Autocorrelação dos $\epsilon^2$ da regressão ARMA(2,2))")
  )
```



E quanto à PACF:

```
pacf(res_m202^2,
     main = TeX(r"(Autocorrelação parcial dos $\epsilon^2$ da regressão ARMA(2,2))"))
```



Vemos que quando analisamos os resíduos ao quadrado, encontramos a presença de autocorrelação serial, indicando que a série não é homoscedástica. Vamos realizar um teste mais robusto de autocorrelação na série.

Teste de Ljung-Box

Para a série de resíduos:

```
box_teste <- Box.test(res_m202, lag = 250, type = "Ljung-Box")
tidy(box_teste) %>%
  kable(
    digits = 5,
    col.names = c(
      "Estatística de Teste",
      "P Value",
      "Lags",
      "Method"
    )
  )
```

Estatística de Teste	P Value	Lags	Method
216.9475	0.93558	250	Box-Ljung test

Assim, não podemos rejeitar a hipótese nula de não haver autocorrelação serial nos resíduos, como era esperado olhando para as funções.

Agora, para a série de seríduos ao quadrado:

```
res2_m202 <- res_m202^2
box_teste <- Box.test(res2_m202, lag = 250, type = "Ljung-Box")
tidy(box_teste) %>%
  kable(
    digits = 5,
    col.names = c(
      "Estatística de Teste",
      "P Value",
      "Lags",
      "Method"
    )
  )
```

Estatística de Teste	P Value	Lags	Method
2751.789	0	250	Box-Ljung test

Nesse caso, ao fazer o teste de Ljung-Box encontramos um p-valor estatísticamente igual a zero. Rejeitamos portanto a hipótese nula de ausência de autocorrelação serial no quadrado dos resíduos. Concluímos que embora não haja correlação entre os resíduos, há correlação serial entre seu quadrados resíduos. Logo, faz-se necessária uma modelagem para a variância.

GARCH

Modelo

Há suporte na literatura para o uso do modelo GARCH(1,1) para modelar a volatilidade de ativos financeiros, vide (Hansen and Lunde 2005). Utilizaremos então o modelo ARMA(2,2) encontrado no passo anterior para modelar a média, e um GARCH(1,1) para modelar a variância.

```
modelo_garch <- garchFit(
  formula = ~arma(2,2) + garch(1,1),
  data = s1,
  trace = FALSE)
```

Temos o seguinte modelo:

$$\Delta \log(P_t) = c + \phi_1 \Delta \log(P_{t-1}) + \phi_2 \Delta \log(P_{t-2}) + \psi_1 \epsilon_{t-1} + \psi_2 \epsilon_{t-2} + \epsilon_t$$

$$\epsilon_t | \mathcal{I}_t \sim \mathcal{N}(0, \sigma_t^2),$$

$$\sigma_t^2 = \omega + \alpha_1 \epsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2 + \eta_t$$

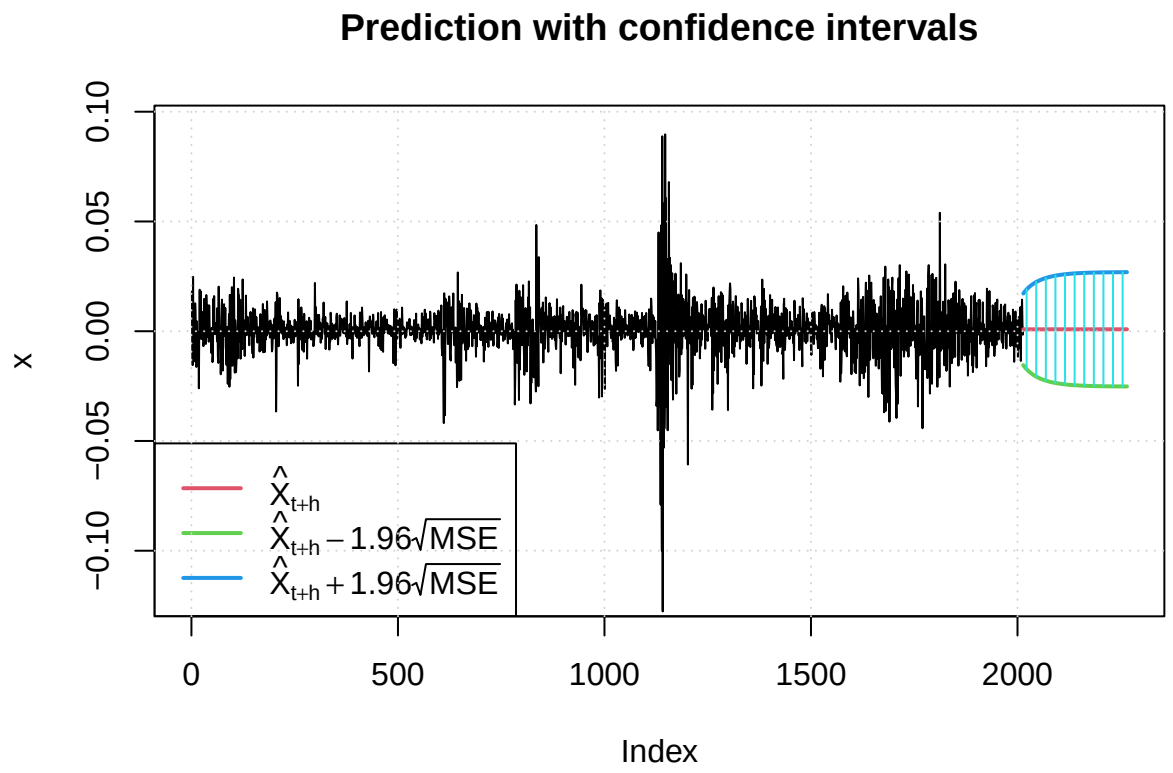
```r
stargazer(
  modelo_garch,
  header = FALSE,
  float = FALSE,
  dep.var.labels = "$\\Delta \\log (P)$",
  digits = 3,
  covariate.labels=c(
    "c",
    "$\\phi_1$",
    "$\\phi_2$",
    "$\\psi_1$",
    "$\\psi_2$",
    "$\\omega$",
    "$\\alpha_1$",
    "$\\beta_1$"
  ))
```

	Dependent variable:
	$\Delta \log(P)$
c	0.002^{***}
	(0.0005)
ϕ_1	-0.952^{***}
	(0.033)
ϕ_2	-0.903^{***}
	(0.034)
ψ_1	0.952^{***}
	(0.041)
ψ_2	0.902^{***}
	(0.035)
ω	0.00000^{***}
	(0.00000)
α_1	0.202^{***}
	(0.025)
β_1	0.777^{***}
	(0.023)
Observations	2,013
Log Likelihood	$-6{,}649.232$
Akaike Inf. Crit.	-6.598
Bayesian Inf. Crit.	-6.576
Note:	*p<0.1; **p<0.05; ***p<0.01

Previsão

Agora, podemos aplicar uma previsão utilizando nosso modelo levando em conta a variância condicional. Primeiro, temos uma previsão da série log-dif. Note que realizamos o forecasting com um ano-padrão de 252 dias úteis.

```
garch_predict = predict(
  modelo_garch,
  n.ahead = 252,
  nx = length(s1),
  plot = TRUE)
```



Em seguida, podemos exibir a previsão da volatilidade, representada pelo Desvio Padrão.

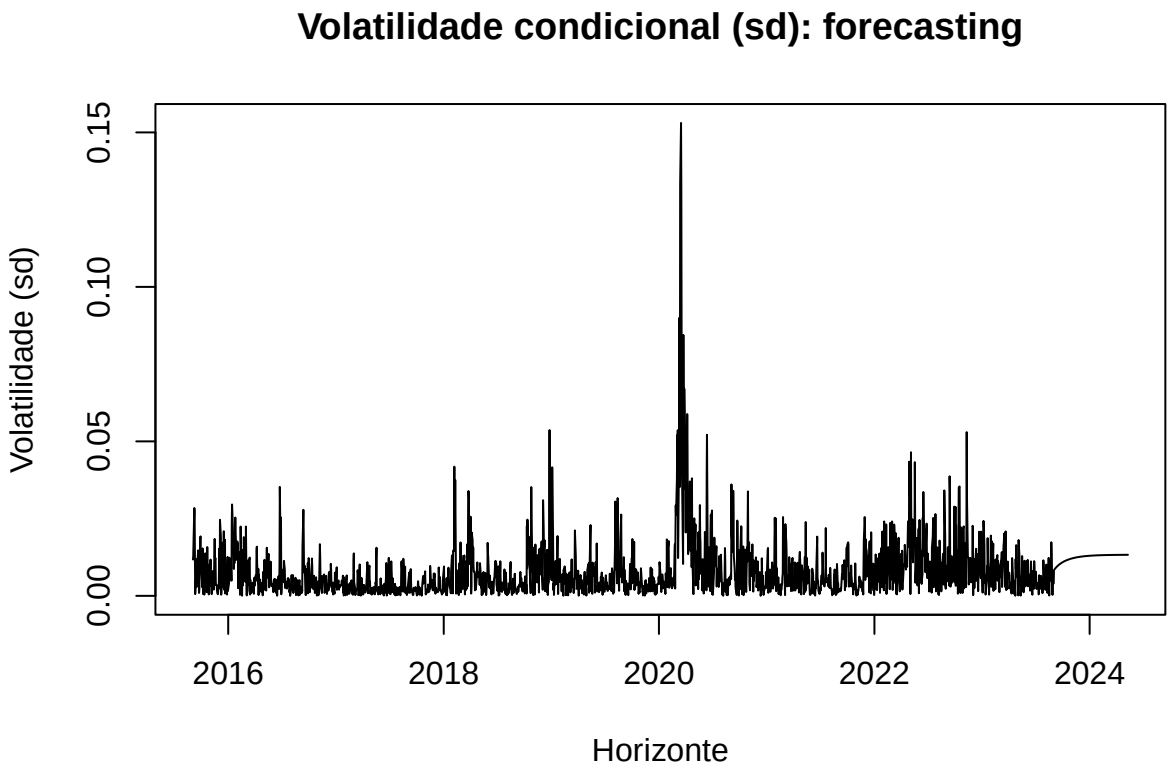
```
garch_predict_xts <- xts(
  garch_predict$standardDeviation,
  order.by = seq(as.Date("2023-09-02"), by = "days", length.out = 252))

var_s1 <- rollapply(dif_log_close, width = 2, FUN = sd, fill = NA)[-1]

var_predict <- rbind(var_s1, garch_predict_xts)
```

```
plot(
  ts(var_predict),
  type = "l",
```

```
  x = index(var_predict),
  xlab = "Horizonte",
  ylab = "Volatilidade (sd)",
  main = "Volatilidade condicional (sd): forecasting"
  )
```



References

Hansen, Peter R., and Asger Lunde. 2005. "A Forecast Comparison of Volatility Models: Does Anything Beat a GARCH(1,1)?" *Journal of Applied Econometrics* 20 (7): 873–89. https://doi.org/10.1002/jae.800.

Questão 2

Shai Vaz, Heron Goulart, Alexandre Almeida, João Pedro Pedrosa, Roberto Orenstein

2023-10-04

Questão 2: Modelo Fama-French

Modelo

Adaptando Cochrane (2005) e Fama and French (1993) :

1. Etapa painel

$$R^e_{i,t} = R_{i,t} - Rf_{i,t} = a_i + \widehat{\beta_{i,Mkt}}(R_{Mkt,t} - Rf) + \widehat{\beta_{i,SMB}}SMB + \widehat{\beta_{i,HML}}HML + \epsilon_{i,t}$$

2. Etapa Cross-Section

$$\mathbb{E}[R^e_{i,t}] = \alpha + \beta_{i,Mkt}\widehat{\lambda_{Mkt}} + \beta_{i,SMB}\widehat{\lambda_{SMB}} + \beta_{i,HML}\widehat{\lambda_{HML}} + \eta_i$$

Importação dos dados

Baixamos os portfólios.

```
port <- read_xlsx(
  "Avg_Val_Weight_Ret_25port1.xlsx",
  ) %>%
  mutate(
    month = as_date(month)
  )
```

Agora, baixamos os fatores.

```
fac <- read_xlsx(
  "F-F_Research_Data_Factors1.xlsx"
  ) %>%
  mutate(
    month = as_date(month)
  )
```

Unificamos os dois.

```r
FF <- full_join(
  fac, port,
  by = "month"
  )

remove(fac, port)
```

Em seguida, pivotamos as colunas de portfólios para long format. A coluna `i`, contém os portfólios, e a coluna `R` contém os seus retornos.

```r
FF_long <- FF %>%
  pivot_longer(
    !1:5,
    names_to = "i",
    values_to = "R"
  ) %>%
  mutate(
    Mkt = as.numeric(Mkt),
    SMB = as.numeric(SMB),
    HML = as.numeric(HML),
    RF = as.numeric(RF)
  )
```

Vamos calcular também o Excesso de Retorno dos portfólios. Notemos que o retorno de mercado (`Mkt`) já está representado também em excesso de retornos.

```r
FF_long <- FF_long %>%
  mutate(
    Re = R - RF
  )
```

Regressão em Painel

Agora, vamos regredir os retornos de cada um dos 25 portfólios nos 3 fatores de Fama-French. Há uma versão em que a variável dependente é o Retorno Bruto, e outra em que a variável dependente é o retorno bruto, e outra com o excesso de retorno.

```r
painel_R <- pvcm(
  R ~ Mkt + SMB + HML,
  data = FF_long,
  model = "within",
  index = c("i", "month")
)
```

```r
betas_R <- coef(painel_R) %>%
  rownames_to_column(var = "i") %>%
  rename(
    a = "(Intercept)",
    b_Mkt = Mkt,
    b_SMB = SMB,
    b_HML = HML
  )
```

```r
painel_Re <- pvcm(
  Re ~ Mkt + SMB + HML,
  data = FF_long,
  model = "within",
  index = c("i", "month")
)
```

```r
betas_Re <- coef(painel_Re) %>%
  rownames_to_column(var = "i") %>%
  rename(
    a = "(Intercept)",
    b_Mkt = Mkt,
    b_SMB = SMB,
    b_HML = HML
  )
```

Resultados do Painel

	Retornos Brutos				Excesso de retornos			
Portfólios	a	β_{Mkt}	β_{SMB}	β_{HML}	a	β_{Mkt}	β_{SMB}	β_{HML}
BIG HiBM	0.093	1.174	-0.173	1.013	-0.176	1.177	-0.171	1.011
BIG LoBM	0.367	1.027	-0.153	-0.266	0.098	1.030	-0.151	-0.269
ME1 BM2	-0.138	1.070	1.535	0.207	-0.406	1.073	1.538	0.205
ME1 BM3	0.126	1.073	1.244	0.544	-0.142	1.076	1.246	0.541
ME1 BM4	0.348	0.939	1.221	0.578	0.079	0.942	1.224	0.575
ME2 BM1	0.030	1.085	1.133	-0.215	-0.239	1.088	1.136	-0.218
ME2 BM2	0.288	1.018	0.991	0.124	0.019	1.021	0.994	0.121
ME2 BM3	0.292	0.987	0.823	0.346	0.024	0.990	0.825	0.343
ME2 BM4	0.301	0.964	0.811	0.569	0.033	0.967	0.814	0.566
ME2 BM5	0.315	1.066	0.916	0.881	0.047	1.069	0.919	0.878
ME3 BM1	0.143	1.127	0.807	-0.219	-0.125	1.130	0.810	-0.222
ME3 BM2	0.370	1.016	0.541	0.039	0.101	1.019	0.544	0.037
ME3 BM3	0.328	0.983	0.441	0.324	0.060	0.986	0.443	0.322
ME3 BM4	0.326	0.996	0.468	0.565	0.057	0.999	0.470	0.562
ME3 BM5	0.210	1.111	0.578	0.860	-0.059	1.114	0.581	0.858
ME4 BM1	0.330	1.074	0.331	-0.338	0.061	1.077	0.334	-0.341
ME4 BM2	0.280	1.026	0.231	0.108	0.011	1.029	0.233	0.105
ME4 BM3	0.282	1.004	0.204	0.344	0.013	1.007	0.207	0.342
ME4 BM4	0.287	1.036	0.204	0.567	0.018	1.039	0.207	0.564
ME4 BM5	0.115	1.186	0.315	0.947	-0.153	1.189	0.317	0.944
ME5 BM2	0.276	0.973	-0.194	0.024	0.007	0.976	-0.191	0.021
ME5 BM3	0.262	0.957	-0.237	0.329	-0.006	0.960	-0.235	0.326
ME5 BM4	0.031	1.029	-0.189	0.664	-0.238	1.032	-0.186	0.661
SMALL HiBM	0.399	0.975	1.306	0.900	0.130	0.978	1.309	0.898
SMALL LoBM	-0.426	1.275	1.464	0.360	-0.695	1.278	1.466	0.358

Regressão em Cross-Section

Novamente, teremos uma versão em que os retornos dos portfólios estão brutos, e outra versão em que estão líquidos da taxa livre de risco, portanto em excesso de retorno.

Inicialmente, preparamos o dataframe de Retornos Esperados (ER) dos portfólios. A seguir, juntaremos esse dado com os betas estimados anteriormente.

```
FF_expected <- FF_long %>%
  select(
    i,R,Re
  ) %>%
  group_by(i) %>%
```

```
  summarise(
    ER = mean(R),
    ERe = mean(Re)
  )
```

Agora, juntamos com os betas.

```
FF_ER <- FF_expected %>%
  full_join(
    betas_R,
    by = "i"
  )
```

```
FF_ERe <- FF_expected %>%
  full_join(
    betas_Re,
    by = "i"
  )
```

A seguir, rodamos regressões em Cross-Section dos retornos esperados nos betas, tanto com retornos brutos quanto com excessos de retorno.

```
cross_ER <- lm(
  ER ~ b_Mkt + b_SMB + b_HML,
  data = FF_ER
  )
```

```
cross_ERe <- lm(
  ERe ~ b_Mkt + b_SMB + b_HML,
  data = FF_ERe
  )
```

Conclusões

Percebe-se que os prêmios de riscos λ não variam se utilizamos qualquer um dos modelos, alterando apenas o intercepto. O que faz sentido. A diferença na constante é precisamente a esperança do retorno do ativo livre de risco $\mathbb{E}[R_f]$.

A seguir, temos os resultados da regressão em Cross-Section, com as duas variáveis dependentes diferentes ($\mathbb{E}[R_i]$ e $\mathbb{E}[R_i^e]$) utilizadas.

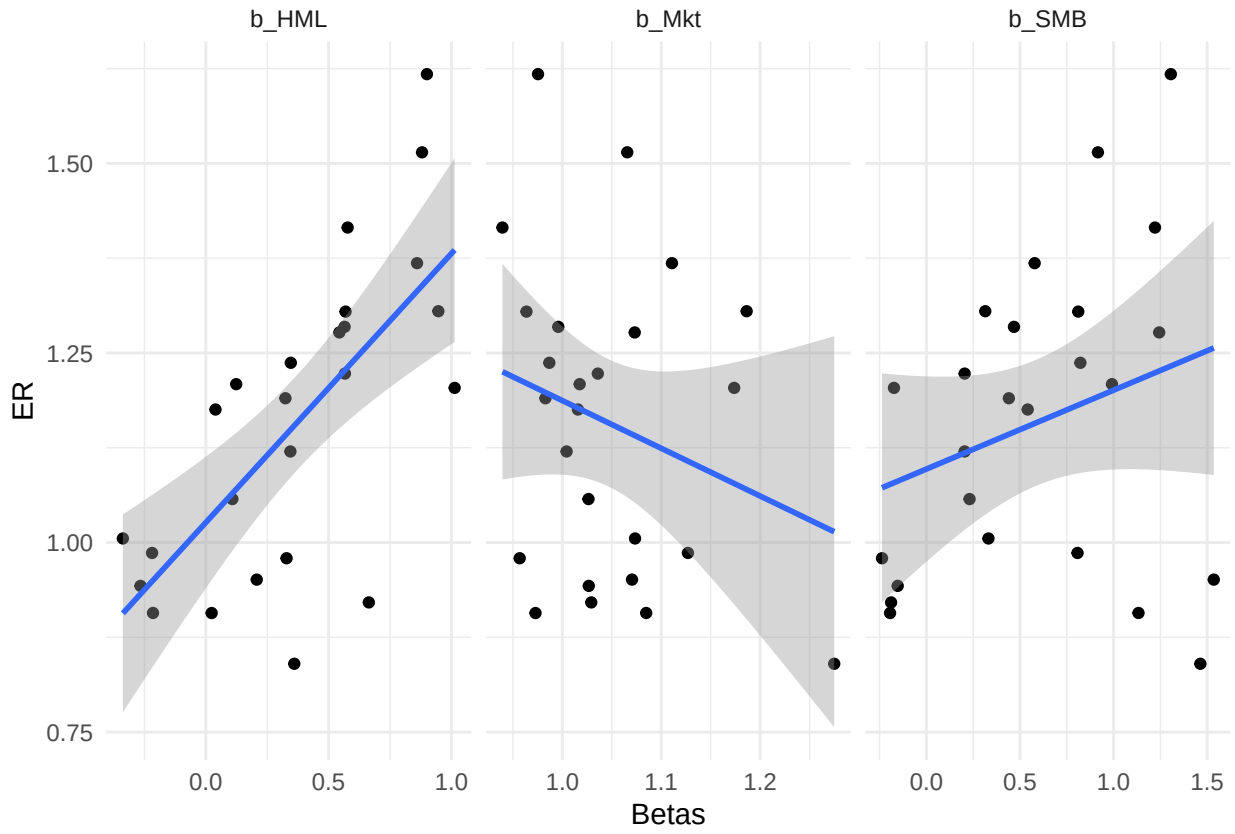
	Dependent variable:	
	Expected Return	Expected Excess Return
	(1)	(2)
λ_{Mkt}	-0.984^{***}	-0.984^{***}
	(0.323)	(0.323)
λ_{SMB}	0.117^{**}	0.117^{**}
	(0.046)	(0.046)
λ_{HML}	0.371^{***}	0.371^{***}
	(0.064)	(0.064)
Constant	1.982^{***}	1.719^{***}
	(0.334)	(0.335)
Observations	25	25
R^2	0.679	0.679
Adjusted R^2	0.633	0.633
Residual Std. Error (df = 21)	0.124	0.124
F Statistic (df = 3; 21)	14.783^{***}	14.783^{***}
Note:		*p<0.1; **p<0.05; ***p<0.01

Gráfico

Podemos visualizar as inclinações λ no gráfico a seguir.

```
FF_ER %>%
  pivot_longer(
    cols = c("b_Mkt","b_SMB","b_HML"),
    names_to = "names",
    values_to = "Betas"
  ) %>%
  ggplot() +
  aes(x=Betas, y=ER) +
  geom_point() +
  geom_smooth(method = "lm") +
  facet_wrap(
    "names",
    scales = "free_x"
  ) +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



References

Cochrane, John H. 2005. *Asset Pricing*. Princeton, NJ: Princeton University Press.

Fama, Eugene F., and Kenneth R. French. 1993. "Common Risk Factors in the Returns on Stocks and Bonds." *Journal of Financial Economics* 33 (1): 3–56. https://doi.org/10.1016/0304-405x(93)90023-5.

Questão 3

Shai Vaz, Alexandre Almeida, Heron Goulart, João Pedro Pedrosa, Roberto Orenstein

2023-10-04

Questão 3

Dados

```
ipca <- rbcb::get_series(code = 433, start_date = "2001-07-01", end_date = "2023-07-01")

ipca <- ipca %>%
  set_names(c("Date","MoM")) %>%
  mutate(
    MoM = ifelse(Date == "2001-07-01", 0, MoM),
    Index = 100 * cumprod(MoM / 100 + 1),
    IndexLog = log(Index),
    IndexLogDiff = c(NA, diff(IndexLog))
  )
```

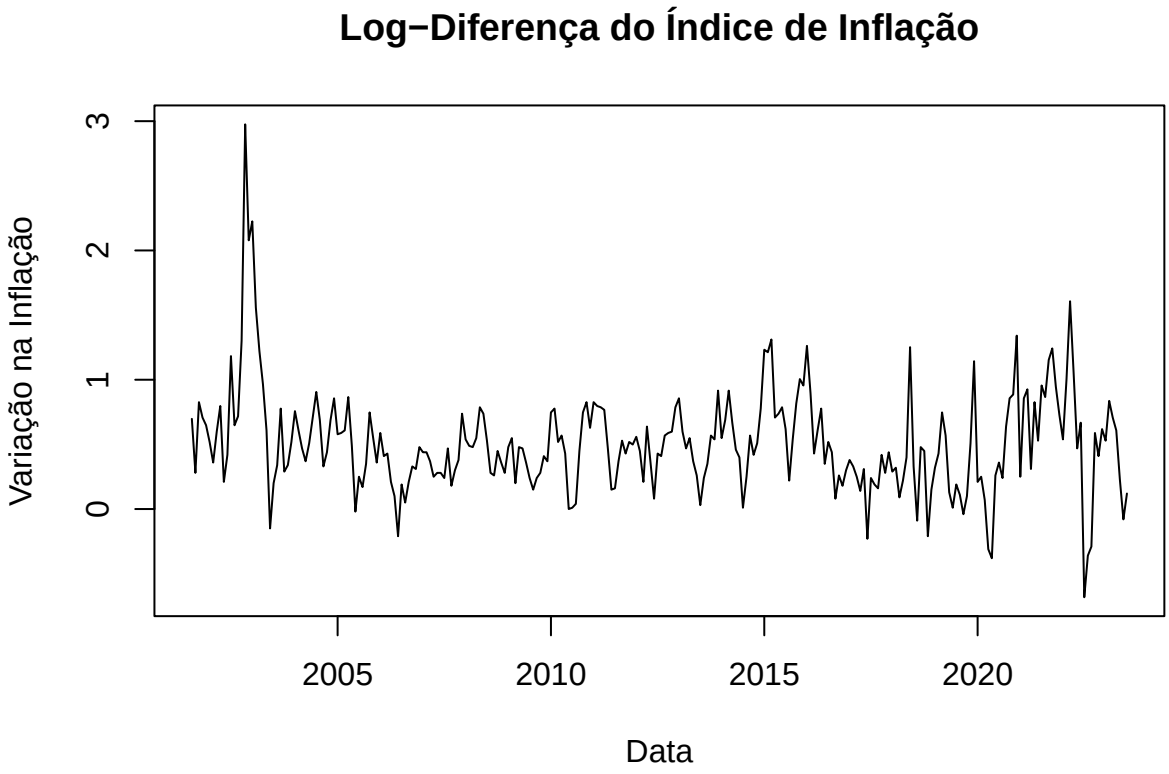
Repare que ao realizar as transformações indicadas no enunciado, obtivemos exatamente o mesmo dado inicial, afinal a diferença de log é uma aproximação para uma variação percentual. Ou seja, apenas perdemos uma observação e pioramos a qualidade do dado porque a aproximação pela primeira diferença dos logs introduz um erro.

Agora vamos criar uma variável no formato de série temporal.

```
IndexLogDiff_ts <- ts(ipca$IndexLogDiff,start = c(2001,7), frequency = 12)
IndexLogDiff_ts <- na.omit(IndexLogDiff_ts)
```

Vamos visualizar a série temporal.

```
plot(
  IndexLogDiff_ts*100,
  xlab = "Data",
  ylab = "Variação na Inflação",
  main = "Log-Diferença do Índice de Inflação"
    )
```



Teste Phillips Perron

```
pp_test <- pp.test(IndexLogDiff_ts)
```

```
tidy(pp_test) %>%
  kable(
    col.names = c(
      "Statistic: Dickey-Fuller Z (alpha)",
      "P Value",
      "Parameter: Truncation lag",
      "Method",
      "Alternative Hypothesis"
    ),
    caption = "Teste Philips Perron"
  )
```

Table 1: Teste Philips Perron

Statistic: Dickey-Fuller Z (alpha)	P Value	Parameter: Truncation lag	Method	Alternative Hypothesis
-95.03949	0.01	5	Phillips-Perron Unit Root Test	stationary

Ao analisarmos o p-value do teste de raiz unitária de Phillips-Perron, podemos rejeitar a hipótese nula e concluir que a série é estacionária.

Teste de Dickey-Fuller aumentado

```
adf_test <- adf.test(IndexLogDiff_ts, alternative = "stationary", k = 12)
```

```
tidy(adf_test) %>%
  kable(
    col.names = c(
      "Statistic: Dickey-Fuller",
      "P Value",
      "Parameter: Lag order",
      "Method",
      "Alternative Hypothesis"
    ),
    caption = "Teste Dickey-Fuller Aumentado"
  )
```

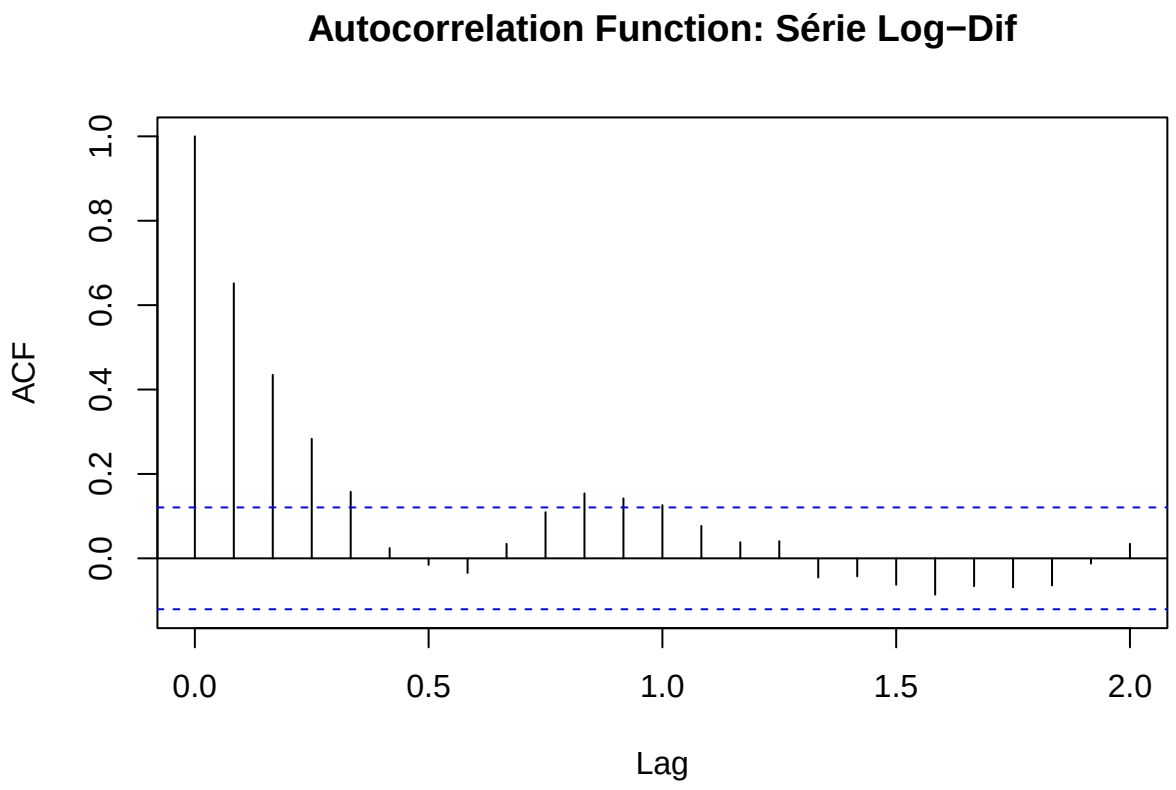
Table 2: Teste Dickey-Fuller Aumentado

Statistic: Dickey-Fuller	P Value	Parameter: Lag order	Method	Alternative Hypothesis
-3.563361	0.0371268	12	Augmented Dickey-Fuller Test	stationary

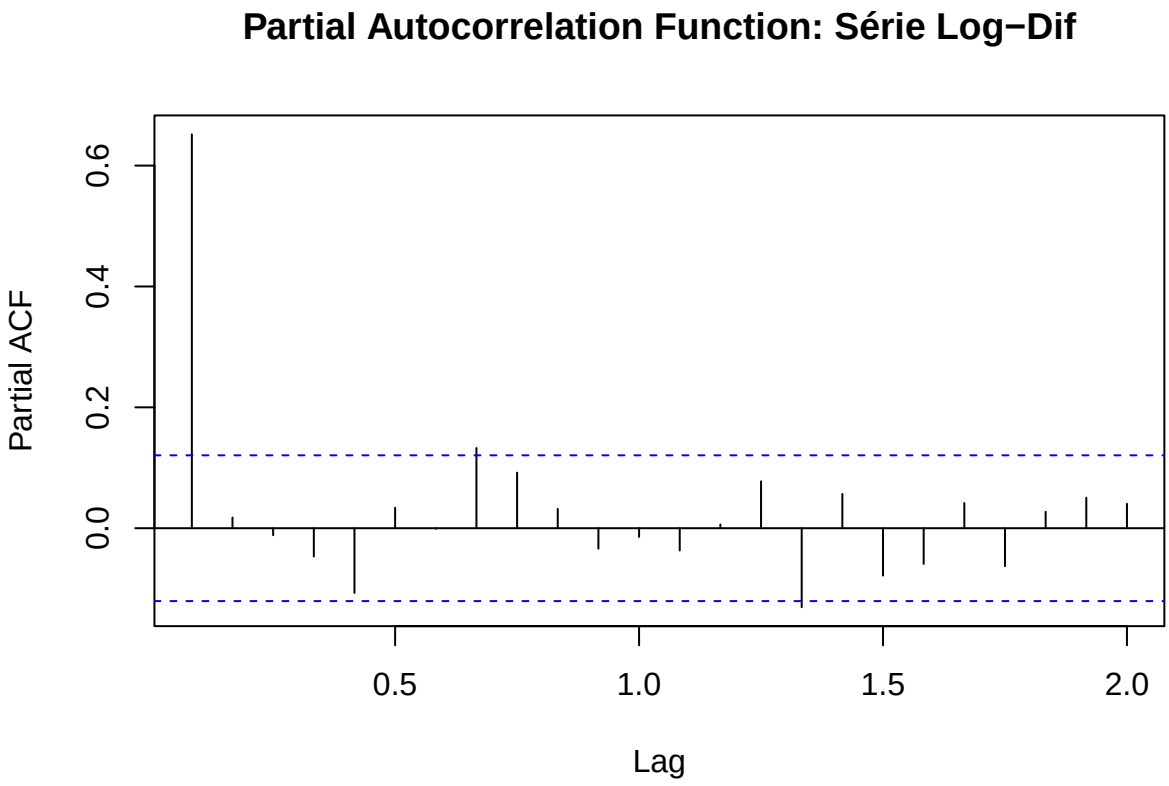
O teste de Dickey-Fuller aumentado que aplicamos à série temporal indica que a série é estacionária. Isso é sustentado pelo valor-p, que é menor do que o nível de significância comum. Portanto, podemos rejeitar a hipótese de que a série possui uma raiz unitária.

Funções de Autocorrelação

```
acf(IndexLogDiff_ts,
    main = "Autocorrelation Function: Série Log-Dif")
```




```
pacf(IndexLogDiff_ts,
    main = "Partial Autocorrelation Function: Série Log-Dif")
```



Modelo ARMA

Decisão pela FAC/FACP

Pelo FAC, identificamos que a ordem do MA é 4 e, pela FACP, que a ordem do AR é 1. No entanto, vamos fazer modelos alternativos para escolher o melhor pelo critério da informação.

```
# Modelo 1
modelo_arma4 <- arima(IndexLogDiff_ts, order = c(1, 0, 4))

# Modelo 2
modelo_arma1 <- arima(IndexLogDiff_ts, order = c(1, 0, 1))

# Modelo 3
modelo_arma2 <- arima(IndexLogDiff_ts, order = c(1, 0, 2))

# Modelo 4
modelo_arma3 <- arima(IndexLogDiff_ts, order = c(1, 0, 3))
```

```
stargazer(
  modelo_arma1,
  modelo_arma2,
  modelo_arma3,
  modelo_arma4,
```

```
  header = FALSE,
  float = FALSE,
  dep.var.caption = "Models",
  dep.var.labels = "",
  column.labels = c(
      "ARIMA(1,0,1)",
      "ARIMA(1,0,2)",
      "ARIMA(1,0,3)",
      "ARIMA(1,0,4)"),
  title = "Resultados dos Modelos ARMA"
)
```

	Models			
	ARIMA(1,0,1)	ARIMA(1,0,2)	ARIMA(1,0,3)	ARIMA(1,0,4)
	(1)	(2)	(3)	(4)
ar1	0.669***	0.658***	0.602***	0.367**
	(0.069)	(0.092)	(0.119)	(0.171)
ma1	−0.029	−0.019	0.028	0.265
	(0.092)	(0.107)	(0.126)	(0.171)
ma2		0.015	0.053	0.193*
		(0.077)	(0.087)	(0.104)
ma3			0.063	0.189**
			(0.077)	(0.081)
ma4				0.181**
				(0.077)
intercept	0.005***	0.005***	0.005***	0.005***
	(0.001)	(0.001)	(0.001)	(0.001)
Observations	264	264	264	264
Log Likelihood	1,158.117	1,158.136	1,158.449	1,160.557
σ^2	0.00001	0.00001	0.00001	0.00001
Akaike Inf. Crit.	−2,308.234	−2,306.271	−2,304.898	−2,307.113
Note:				*p<0.1; **p<0.05; ***p<0.01

Auto Arima (critérios de informação)

Utilizaremos a função `auto.arima`, que, de forma automática, calcula todos os modelos dentro de limites dados, calcula seus critérios de informação e escolhe o modelo que minimiza. Aqui, escolhemos os limites de 5 lags para cada parte do modelo (AR e MA).

```
modelo_arma_auto <- auto.arima(
  max.p = 5,
  max.q = 5,
  IndexLogDiff_ts,
```

```
  seasonal = FALSE,
  stepwise = FALSE,
  approximation = FALSE,
  ic = "bic"
  )
```

```
tidy(modelo_arma_auto) %>%
  kable(
    col.names = c(
      "Term",
      "Estimate",
      "Standard Error"
    ),
    caption = "Information Criteria Minimizer Model"
  )
```

Table 3: Information Criteria Minimizer Model

Term	Estimate	Standard Error
ar1	0.6520244	0.0464532
intercept	0.0050400	0.0005318

Ele sugere que o melhor modelo é, na realidade, o AR(1). Isso ocorre utilizando qualquer dos três criétios como base (AIC, BIC ou AICc). Vamos visualizar, aqui, os modelos pelos critério da informação. Salvaremos esse modelo como `modelo_arma5`.

```
modelo_arma5 <- arima(IndexLogDiff_ts, order = c(1, 0, 0))
```

Critério da Informação

Vamos olhar agora, todos os modelos juntos:

```
stargazer(m5,m1,m2,m3,m4,
  header = FALSE,
  float = FALSE,
  dep.var.caption = "Models",
  dep.var.labels = "",
  column.labels = c(
    "AR(1)",
    "ARIMA(1,0,1)",
    "ARIMA(1,0,2)",
    "ARIMA(1,0,3)",
    "ARIMA(1,0,4)"),
  title = "Resultados de todos os modelos realizados"
  )
```

	Models				
	AR(1)	ARIMA(1,0,1)	ARIMA(1,0,2)	ARIMA(1,0,3)	ARIMA(1,0,4)
	(1)	(2)	(3)	(4)	(5)
ar1	0.652***	0.669***	0.658***	0.602***	0.367**
	(0.046)	(0.069)	(0.092)	(0.119)	(0.171)
ma1		−0.029	−0.019	0.028	0.265
		(0.092)	(0.107)	(0.126)	(0.171)
ma2			0.015	0.053	0.193*
			(0.077)	(0.087)	(0.104)
ma3				0.063	0.189**
				(0.077)	(0.081)
ma4					0.181**
					(0.077)
intercept	0.005***	0.005***	0.005***	0.005***	0.005***
	(0.001)	(0.001)	(0.001)	(0.001)	(0.001)
Observations	264	264	264	264	264
Log Likelihood	1,158.069	1,158.117	1,158.136	1,158.449	1,160.557
σ^2	0.00001	0.00001	0.00001	0.00001	0.00001
Akaike Inf. Crit.	−2,310.138	−2,308.234	−2,306.271	−2,304.898	−2,307.113
Note:					*p<0.1; **p<0.05; ***p<0.01

Note que o modelo que escolhemos originalmente, tem a maior log verossimilhança! Este seria o melhor modelo, portanto, se o critério de decisão fosse esse. Mas ao utilizarmos o critério de informação, punimos o modelo pelo aumento na quantidade de parâmetros.

Decidindo pela minimização dos critérios de informação, o modelo 5, um AR(1), é de fato o melhor modelo. Embora a análise da FAC e FACP tenha nos levado ao modelo ARMA(1,4), o modelo AR(1) é mais parcimonioso e prosseguiremos com ele.

Podemos ver isso mais claramente na tabela seguinte, apenas com os critérios.

```
bind_rows(
  glance(modelo_arma1),
  glance(modelo_arma2),
  glance(modelo_arma3),
  glance(modelo_arma4),
  glance(modelo_arma5)
  ) %>%
  mutate(
    Model = c(
      "ARIMA(1,0,1)",
      "ARIMA(1,0,2)",
      "ARIMA(1,0,3)",
      "ARIMA(1,0,4)",
      "AR(1)"),
    .before = 1
```

```
  ) %>%
  select(
    c(1,3,4,5)
  ) %>%
  kable(
    col.names = c(
      "Model",
      "Log Likelihood",
      "AIC",
      "BIC"
    ),
    caption = "Decision Criteria"
  )
```

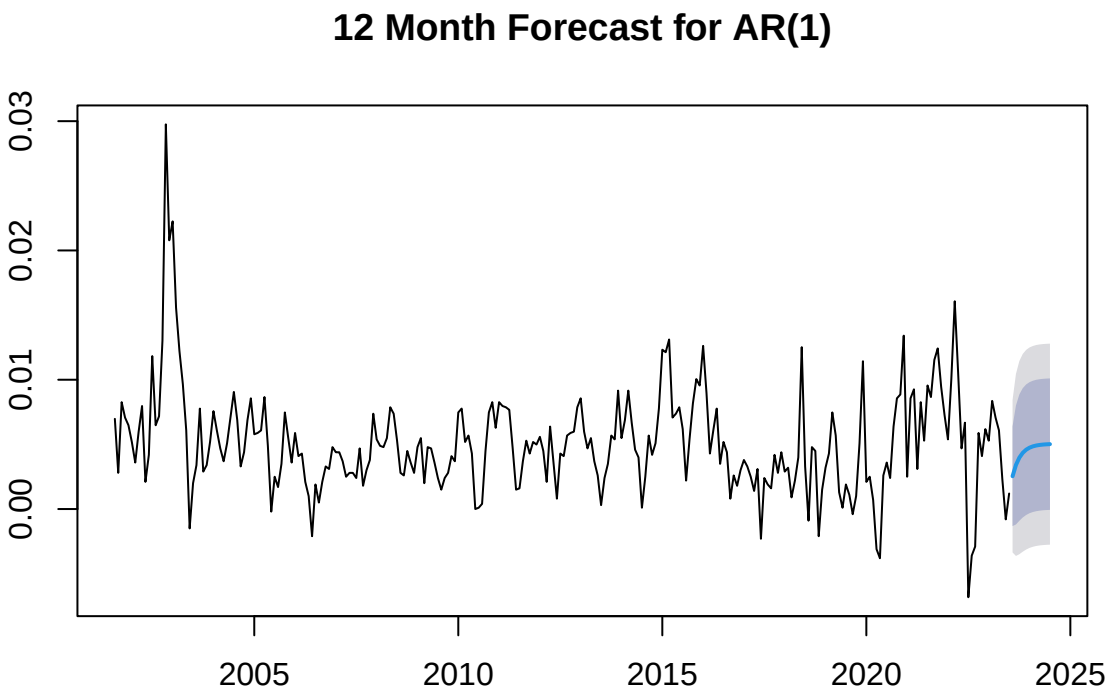
Table 4: Decision Criteria

Model	Log Likelihood	AIC	BIC
ARIMA(1,0,1)	1158.117	-2308.234	-2293.930
ARIMA(1,0,2)	1158.136	-2306.271	-2288.392
ARIMA(1,0,3)	1158.449	-2304.898	-2283.443
ARIMA(1,0,4)	1160.557	-2307.113	-2282.082
AR(1)	1158.069	-2310.138	-2299.410

Previsão

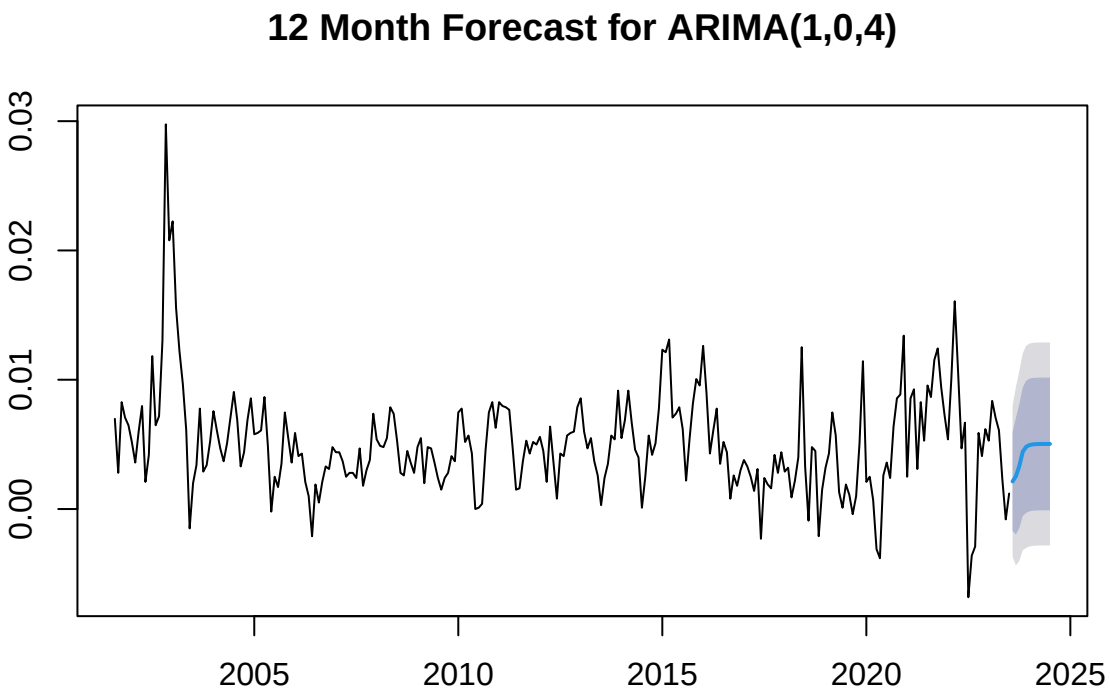
Plotando as previsões do modelo 5 um ano a frente.

```
plot(forecast(modelo_arma5, h=12),
     main = "12 Month Forecast for AR(1)")
```



Podemos também plotar as previsões para o modelo ARIMA(1,0,4), que foi obtido pela análise da FAC e FACP.

```
plot(forecast(modelo_arma4,h=12),
     main = "12 Month Forecast for ARIMA(1,0,4)")
```


12 Month Forecast for ARIMA(1,0,4)
0.00
0.01
0.02
0.03
2005
2010
2015
2020
2025