Forecasting Task - Problem Set 3

Bernardo Calvente e Ricardo Castro

2024-09-05

1 Setup

The following packages were used:

```
library(tidyverse)
library(patchwork)
library(glue)

library(zoo)
library(sandwich)
library(strucchange)
library(lmtest)

library(forecast)
library(varr) #devtools::install_github("ricardo-semiao/varr")
library(opera)

#library(writexl)
library(stargazer)
library(knitr)
library(kableExtra)

theme_set(theme_bw())
```

Custom function for the ADF test:

```
output_dftest <- function(data,
  nlag = NULL, pval = TRUE, index = TRUE, ...) {
  aTSA::adf.test(data, nlag = nlag, output = FALSE) %>%
    imap_dfr(~ tibble(type = .y, as_tibble(.x))) %>%
    mutate(
      p.value = if (pval) {glue("({round(p.value, 2)})")} else {""},
      ADF = round(ADF, 2)
    ) %>%
    unite(Statistic, ADF, p.value, sep = " ") %>%
    pivot_wider(names_from = type, values_from = "Statistic") %>%
    set_names("Lag", glue("Type {1:3}")[index]) %>%
    stargazer(summary = FALSE, header = FALSE, table.placement = "H")
}
```

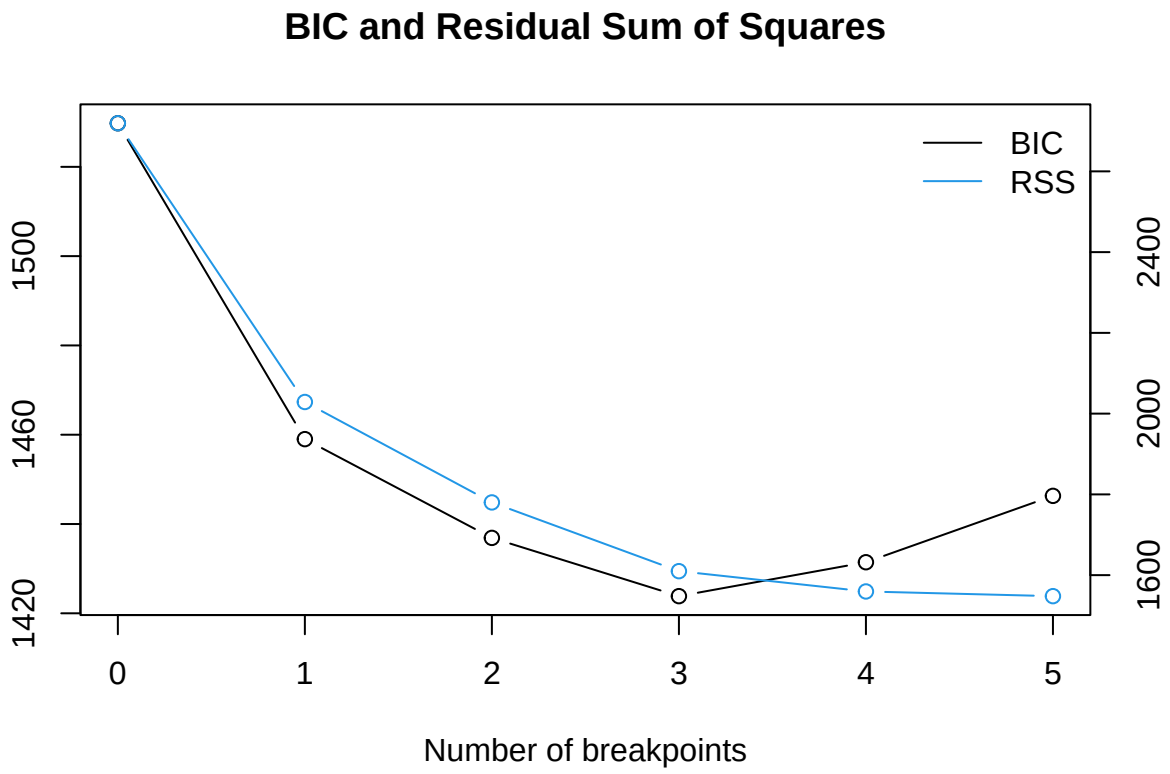
1.1 Adjusting the dataset

```r
# Load the CSV file
dados <- read.csv("../ps_data.csv", header = TRUE, sep = ",")

# Rename variables
dados$y0 <- dados$y_new1
dados$x0 <- dados$x_new

# Create lagged variables
dados$x1 <- dplyr::lag(dados$x0, 1)
dados$x2 <- dplyr::lag(dados$x0, 2)
dados$y1 <- dplyr::lag(dados$y0, 1)
dados$y2 <- dplyr::lag(dados$y0, 2)

# Create difference variables
dados$dif_x0 <- dados$x0 - dados$x1
dados$dif_x1 <- dados$x1 - dados$x2
dados$dif_y1 <- dados$y1 - dados$y2

# Remove the first 100 and the last 200 rows
dados_estimacao <- dados[-c(1:100, (nrow(dados)-199):nrow(dados)), ]

# Fit a simple linear model with breakpoints
bp_test <- breakpoints(y0 ~ x0, data = dados_estimacao)

# Summary of the results
summary(bp_test)
```

```
##
##   Optimal (m+1)-segment partition:
##
## Call:
## breakpoints.formula(formula = y0 ~ x0, data = dados_estimacao)
##
## Breakpoints at observation number:
##
## m = 1          153
## m = 2          154 210
## m = 3      101 166 211
## m = 4   46 101 166 211
## m = 5   46 101 165 210 255
##
## Corresponding to breakdates:
##
## m = 1                                          0.51
## m = 2                                          0.513333333333333 0.7
## m = 3                     0.336666666666667 0.553333333333333 0.703333333333333
## m = 4   0.153333333333333 0.336666666666667 0.553333333333333 0.703333333333333
## m = 5   0.153333333333333 0.336666666666667 0.55              0.7
##
## m = 1
## m = 2
## m = 3
```

```
## m = 4
## m = 5   0.85
##
## Fit:
##
## m   0    1    2    3    4    5
## RSS 2719 2029 1780 1610 1560 1548
## BIC 1530 1459 1437 1424 1431 1446
```

```r
# Plot the results
plot(bp_test)
```




```r
# Create dummies for breakpoints - problem does not specify which dummies to use. so we are using the s
dados_estimacao$dummy_101 <- ifelse(1:nrow(dados_estimacao) >= 101, 1, 0)
dados_estimacao$dummy_166 <- ifelse(1:nrow(dados_estimacao) >= 166, 1, 0)
dados_estimacao$dummy_211 <- ifelse(1:nrow(dados_estimacao) >= 211, 1, 0)

dados$dummy_101 <- ifelse(1:nrow(dados) >= 201, 1, 0)
dados$dummy_166 <- ifelse(1:nrow(dados) >= 266, 1, 0)
dados$dummy_211 <- ifelse(1:nrow(dados) >= 311, 1, 0)
```

2 Part I

2.1 1) Perform bias and efficiency tests

```
# Model 1: Linear regression
model_1 <- lm(
  y0 ~ x0,
  data = dados_estimacao
)

# Model 2: Model with dummy variables
model_2 <- lm(
  y0 ~ dummy_101 + dummy_166 + dummy_211 +
        x0 +
        dummy_101 * x0 +
        dummy_166 * x0 +
        dummy_211 * x0,
  data = dados_estimacao
)

# Model 3: Dynamic model with lagged variables + dummy variables
model_3 <- lm(
  y0 ~ dummy_101 + dummy_166 + dummy_211 +
        x0 +
        dummy_101 * x0 +
        dummy_166 * x0 +
        dummy_211 * x0 +
        y1 +
        x1,
  data = dados_estimacao
)
```

Considering the forcast error:

$$\epsilon_{t+h} = y_{t+h} - \hat{y}_{t+h} = \alpha + e_{t+h}$$

the bias test evaluates whether $\alpha = 0$ (the null hypothesis). If the null hypothesis holds, it indicates that the forecast is unbiased. We will conduct this test in the context of non-recursive forecasts, using the last 200 periods of our sample as the forecast evaluation window.

```
# Forecasting one-step ahead (non-recursive)
forcast_model_1 <- model_1$coefficients[1] + model_1$coefficients[2]*dados$x0
forcast_model_2 <- model_2$coefficients[1] +
                   model_2$coefficients[2] * dados$dummy_101 +
                   model_2$coefficients[3] * dados$dummy_166 +
                   model_2$coefficients[4] * dados$dummy_211 +
                   model_2$coefficients[5] * dados$x0 +
                   model_2$coefficients[6] * dados$dummy_101 * dados$x0 +
                   model_2$coefficients[7] * dados$dummy_166 * dados$x0 +
                   model_2$coefficients[8] * dados$dummy_211 * dados$x0
forcast_model_3 <- model_3$coefficients[1] +
                   model_3$coefficients[2] * dados$dummy_101 +
                   model_3$coefficients[3] * dados$dummy_166 +
                   model_3$coefficients[4] * dados$dummy_211 +
```

```
                  model_3$coefficients[5] * dados$x0 +
                  model_3$coefficients[6] * dados$y1  +
                  model_3$coefficients[7] * dados$x1  +
                  model_3$coefficients[8] * dados$dummy_101 * dados$x0 +
                  model_3$coefficients[9] *  dados$dummy_166 * dados$x0 +
                  model_3$coefficients[10] * dados$dummy_211 * dados$x0

# Forecasting one-step ahead (recursive)
forcast_model_1_rec <- rep(0,200)
forcast_model_2_rec <- rep(0,200)
forcast_model_3_rec <- rep(0,200)

for (i in 1:200) {
  # Estimation
    dados_estimacao_rec <- dados[101:(nrow(dados)-i),]
    # Model 1: Linear regression
    model_1 <- lm(
      y0 ~ x0,
      data = dados_estimacao_rec
    )

    # Model 2: Model with dummy variables
    model_2 <- lm(
      y0 ~ dummy_101 + dummy_166 + dummy_211 +
            x0 +
            dummy_101 * x0 +
            dummy_166 * x0 +
            dummy_211 * x0,
      data = dados_estimacao_rec
    )

    # Model 3: Dynamic model with lagged variables + dummy variables
    model_3 <- lm(
      y0 ~ dummy_101 + dummy_166 + dummy_211 +
            x0 +
            dummy_101 * x0 +
            dummy_166 * x0 +
            dummy_211 * x0 +
            y1 +
            x1,
      data = dados_estimacao_rec
    )

  # Forecasts
  forcast_model_1_rec_help <- model_1$coefficients[1] + model_1$coefficients[2]*dados$x0
  forcast_model_2_rec_help <- model_2$coefficients[1] +
                    model_2$coefficients[2] * dados$dummy_101 +
                    model_2$coefficients[3] * dados$dummy_166 +
                    model_2$coefficients[4] * dados$dummy_211 +
                    model_2$coefficients[5] * dados$x0 +
                    model_2$coefficients[6] * dados$dummy_101 * dados$x0 +
                    model_2$coefficients[7] * dados$dummy_166 * dados$x0 +
                    model_2$coefficients[8] * dados$dummy_211 * dados$x0
```

```
  forcast_model_3_rec_help <- model_3$coefficients[1] +
                      model_3$coefficients[2] * dados$dummy_101 +
                      model_3$coefficients[3] * dados$dummy_166 +
                      model_3$coefficients[4] * dados$dummy_211 +
                      model_3$coefficients[5] * dados$x0 +
                      model_3$coefficients[6] * dados$y1 +
                      model_3$coefficients[7] * dados$x1  +
                      model_3$coefficients[8] * dados$dummy_101 * dados$x0 +
                      model_3$coefficients[9] *  dados$dummy_166 * dados$x0 +
                      model_3$coefficients[10] * dados$dummy_211 * dados$x0
  forcast_model_1_rec[200-i+1] <- forcast_model_1_rec_help[nrow(dados)-i+1]
  forcast_model_2_rec[200-i+1] <- forcast_model_2_rec_help[nrow(dados)-i+1]
  forcast_model_3_rec[200-i+1] <- forcast_model_3_rec_help[nrow(dados)-i+1]
}

# Forcast errors
ferror_model1 <- tail(dados$y0, 200)-tail(forcast_model_1, 200)
ferror_model2 <- tail(dados$y0, 200)-tail(forcast_model_2, 200)
ferror_model3 <- tail(dados$y0, 200)-tail(forcast_model_3, 200)
ferror_model1_rec <- tail(dados$y0, 200)-forcast_model_1_rec
ferror_model2_rec <- tail(dados$y0, 200)-forcast_model_2_rec
ferror_model3_rec <- tail(dados$y0, 200)-forcast_model_3_rec

# Bias test
bias_test_model_1 <- coeftest(lm(ferror_model1 ~ 1), vcov. = NeweyWest)
p_value_1 <- bias_test_model_1[1, "Pr(>|t|)"]
t_stat_1 <- bias_test_model_1[1, "t value"]

bias_test_model_2 <- coeftest(lm(ferror_model2 ~ 1), vcov. = NeweyWest)
p_value_2 <- bias_test_model_2[1, "Pr(>|t|)"]
t_stat_2 <- bias_test_model_2[1, "t value"]

bias_test_model_3 <- coeftest(lm(ferror_model3 ~ 1), vcov. = NeweyWest)
p_value_3 <- bias_test_model_3[1, "Pr(>|t|)"]
t_stat_3 <- bias_test_model_3[1, "t value"]

bias_test_model_1_rec <- coeftest(lm(ferror_model1_rec ~ 1), vcov. = NeweyWest)
p_value_1_rec <- bias_test_model_1_rec[1, "Pr(>|t|)"]
t_stat_1_rec <- bias_test_model_1_rec[1, "t value"]

bias_test_model_2_rec <- coeftest(lm(ferror_model2_rec ~ 1), vcov. = NeweyWest)
p_value_2_rec <- bias_test_model_2_rec[1, "Pr(>|t|)"]
t_stat_2_rec <- bias_test_model_2_rec[1, "t value"]

bias_test_model_3_rec <- coeftest(lm(ferror_model3_rec ~ 1), vcov. = NeweyWest)
p_value_3_rec <- bias_test_model_3_rec[1, "Pr(>|t|)"]
t_stat_3_rec <- bias_test_model_3_rec[1, "t value"]


# Montando a tabela com os valores dos seis modelos (incluindo os recursivos)
results_table <- data.frame(
  Model = c("Model 1", "Model 2", "Model 3", "Model 1 (Rec)", "Model 2 (Rec)", "Model 3 (Rec)"),
  `T-Statistic` = c(t_stat_1, t_stat_2, t_stat_3, t_stat_1_rec, t_stat_2_rec, t_stat_3_rec),
```

```
  `P-Value` = c(p_value_1, p_value_2, p_value_3, p_value_1_rec, p_value_2_rec, p_value_3_rec)
)

# Gerar a tabela em formato LaTeX
kable(results_table, format = "latex", caption = "Bias Test Results for Each Model", booktabs = TRUE) %>
  kable_styling(latex_options = c("hold_position", "striped"))
```

Table 1: Bias Test Results for Each Model

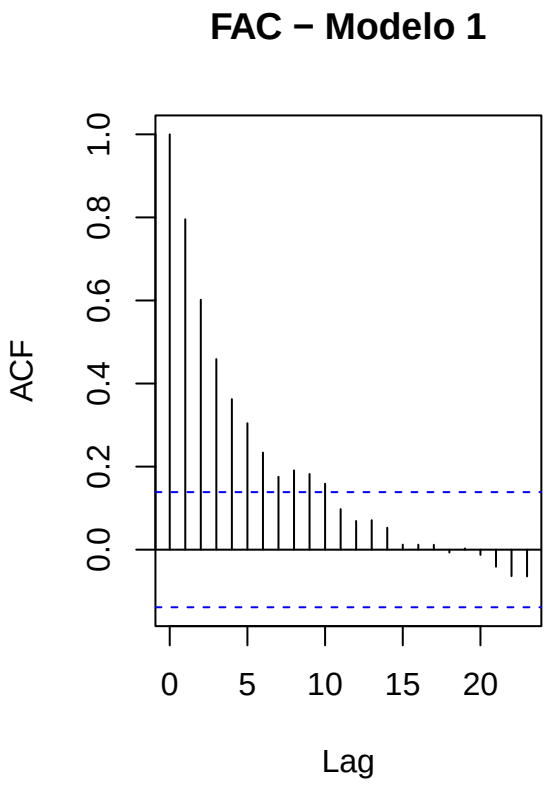
Model	T.Statistic	P.Value
Model 1	1.3748498	0.1707240
Model 2	2.1086974	0.0362217
Model 3	0.2474865	0.8047869
Model 1 (Rec)	0.7325183	0.4647142
Model 2 (Rec)	0.5416769	0.5886471
Model 3 (Rec)	-0.3807313	0.7038088

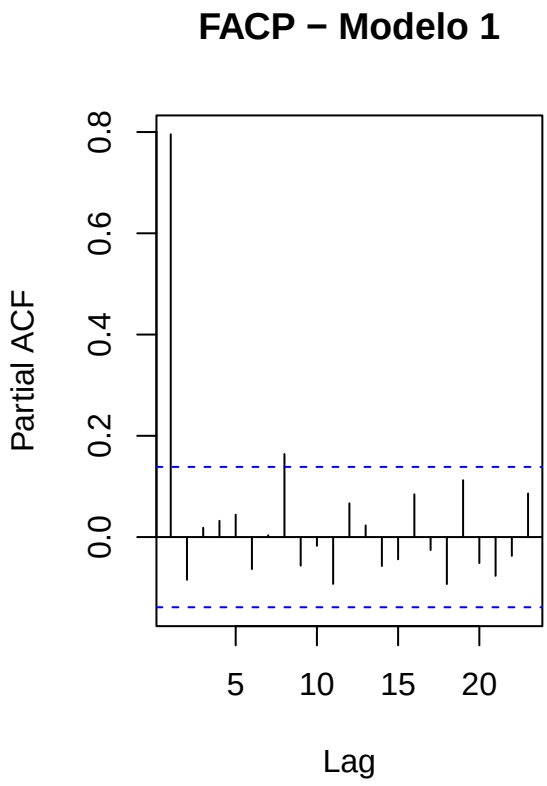
At a 95% confidence level, only non-recursive Model 2 (the static regression with dummies) shows evidence of bias in the forecasts.

An efficient h-step ahead forecast error should exhibit correlations only up to lag $h-1$ and should be uncorrelated with information available at the time of the forecast.

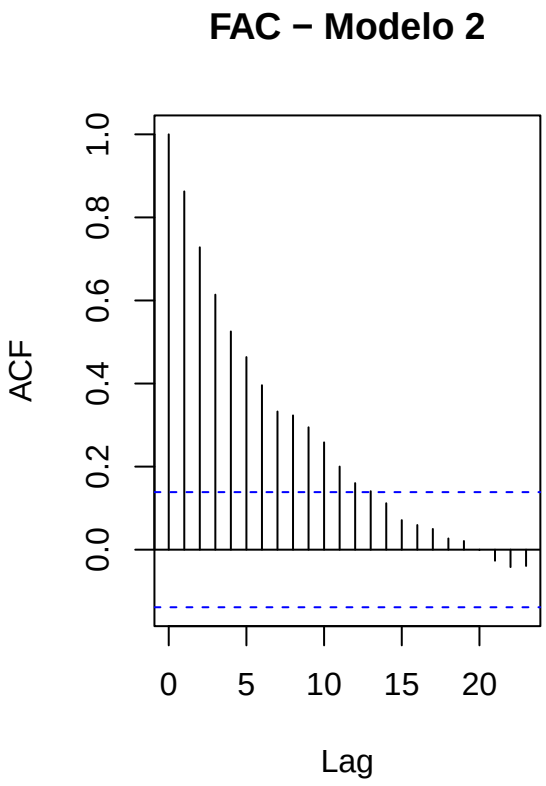
We will test for weak efficiency, we check if the forecast error ϵ_{t+h} is correlated over time only up to lag $h-1$. This implies that no information beyond lag $h-1$ should explain the forecast errors. This property can be tested by fitting a moving average model MA($h-1$) to the h-step ahead forecast error and ensuring that the residuals from this model are white noise.

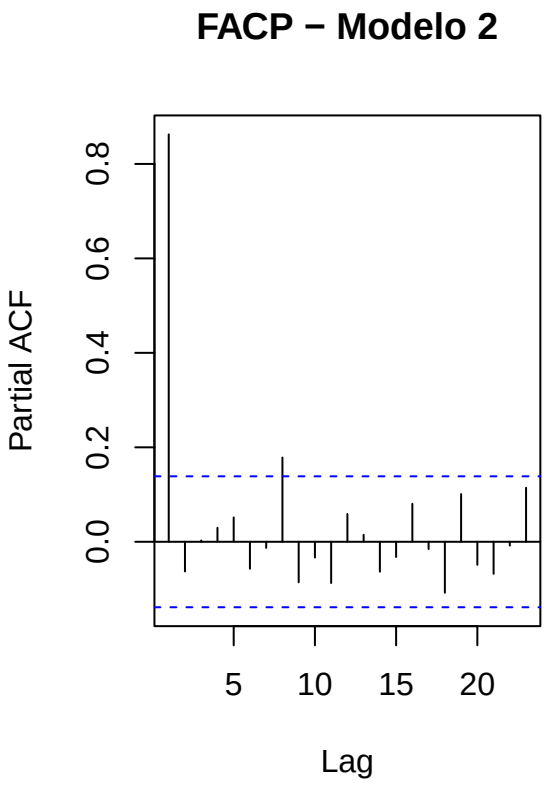
```
# Função para ajustar um modelo MA(1) e gerar os gráficos de FAC e FACP
plot_acf_pacf_ma1 <- function(errors, model_name) {
  # Ajustar o modelo MA(1)
  fit_ma1 <- Arima(errors, order = c(0, 0, 0))

  # Extrair os resíduos
  residuals_ma1 <- residuals(fit_ma1)

  # Configurar a janela de gráficos para 1 linha e 2 colunas
  par(mfrow = c(1, 2))

  # Gráfico da Função de Autocorrelação (FAC)
  acf(residuals_ma1, main = paste("FAC -", model_name))

  # Gráfico da Função de Autocorrelação Parcial (FACP)
  pacf(residuals_ma1, main = paste("FACP -", model_name))
}

# Ajustar o modelo MA(1) e gerar os gráficos para os seis modelos

# Modelos originais
plot_acf_pacf_ma1(ferror_model1, "Modelo 1")
```



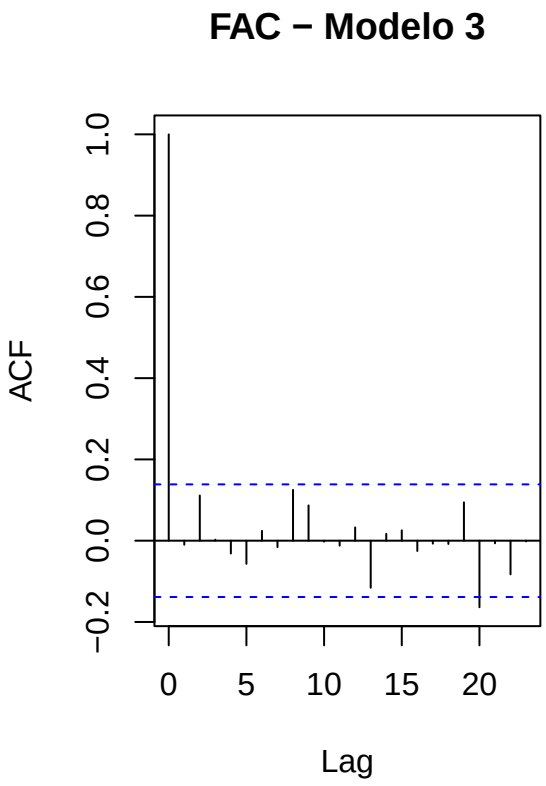


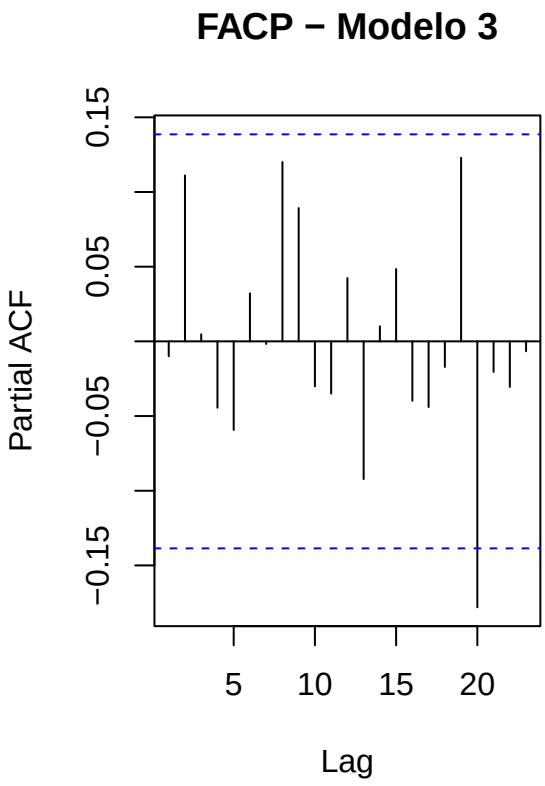

```
plot_acf_pacf_ma1(ferror_model2, "Modelo 2")
```



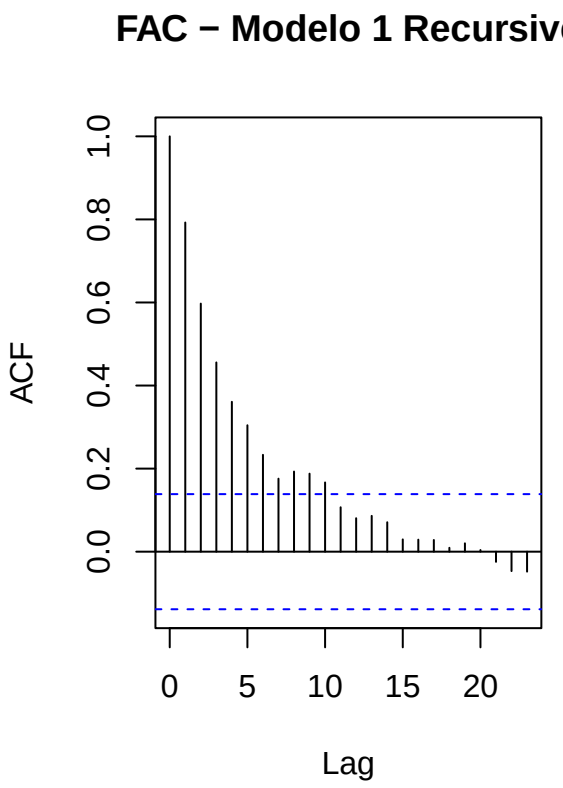


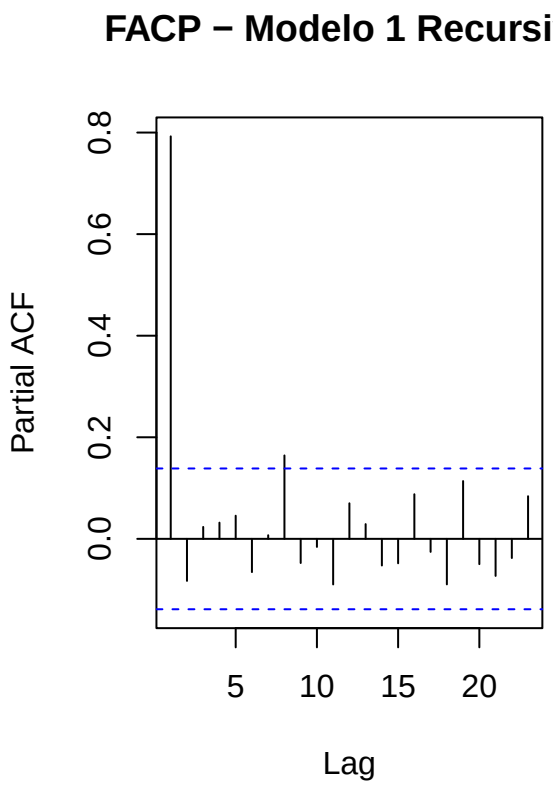

```
plot_acf_pacf_ma1(ferror_model3, "Modelo 3")
```



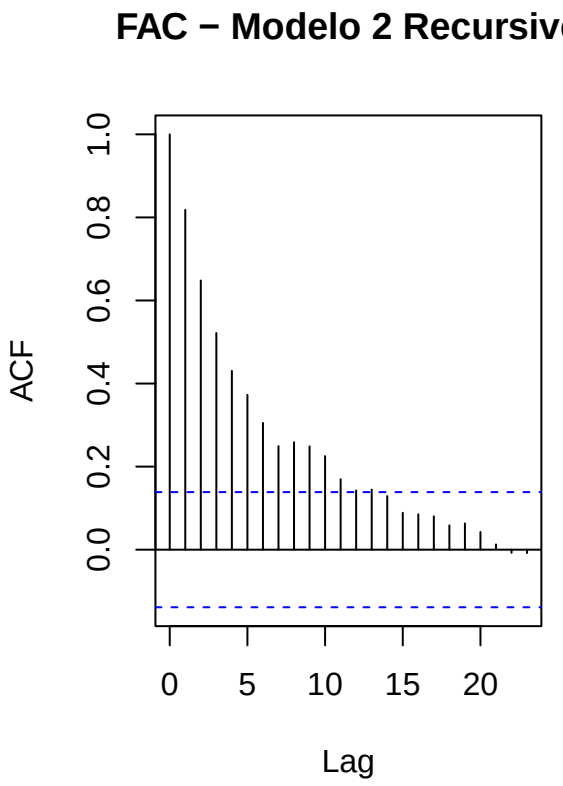


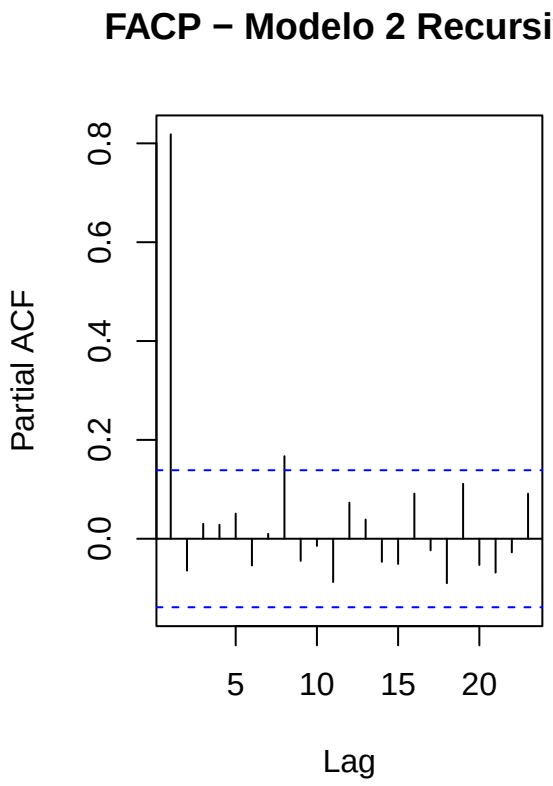

```r
# Modelos ajustados
plot_acf_pacf_ma1(ferror_model1_rec, "Modelo 1 Recursive")
```



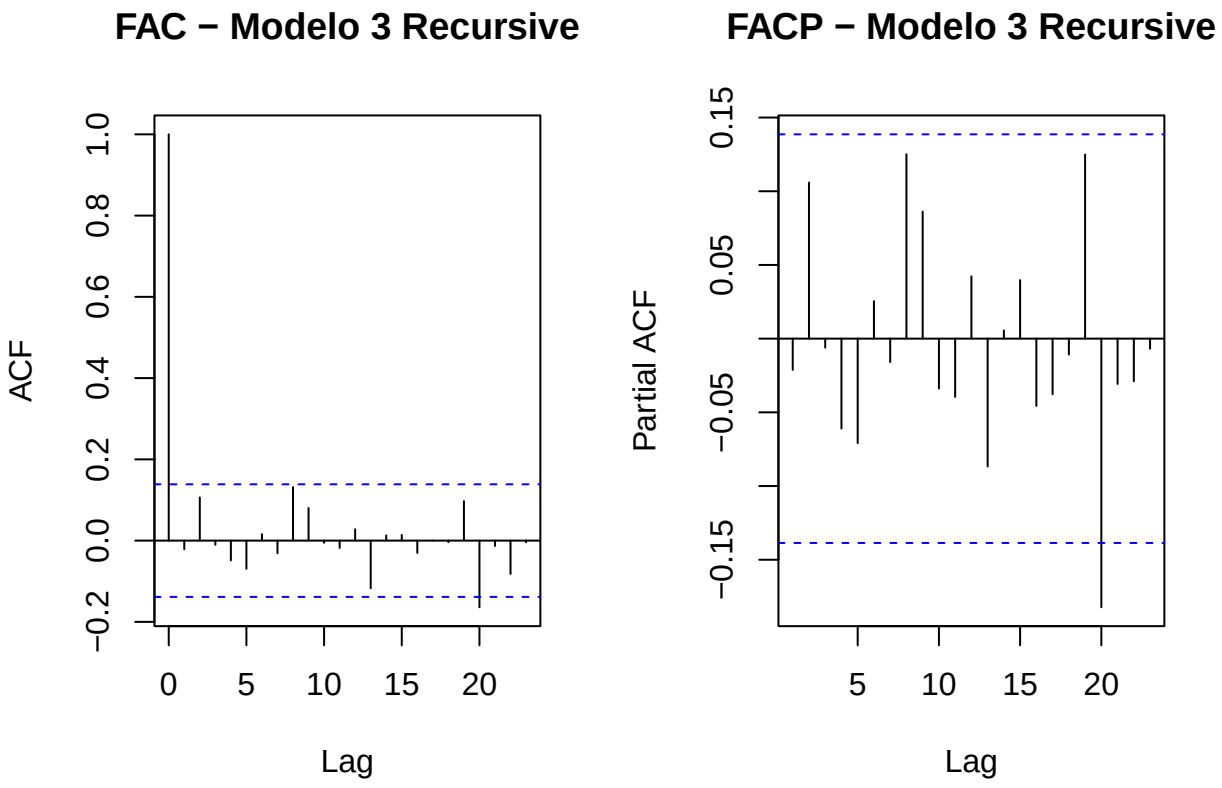



```
plot_acf_pacf_ma1(ferror_model2_rec, "Modelo 2 Recursive")
```






```
plot_acf_pacf_ma1(ferror_model3_rec, "Modelo 3 Recursive")
```



Since the forecasts are one-step ahead, $h = 1$. The ACF and PACF do not appear consistent with the residuals being white noise for most models, suggesting that the forecasts may not be efficient. Only in model 3 do the residuals resemble white noise, indicating a weak efficiency of the forecasts.

2.2 2) Compare the forecasts using "Pairs" tests

First, we will proceed with the Diebold-Mariano test:

```
# Realizar os testes Diebold-Mariano
dm_test_12 <- dm.test(ferror_model1, ferror_model2, h = 1)
dm_test_13 <- dm.test(ferror_model1, ferror_model3, h = 1)
dm_test_23 <- dm.test(ferror_model2, ferror_model3, h = 1)

# Criar uma tabela com os resultados
dm_results <- data.frame(
  Comparisons = c("Model 1 vs Model 2", "Model 1 vs Model 3", "Model 2 vs Model 3"),
  DM_Statistic = c(dm_test_12$statistic, dm_test_13$statistic, dm_test_23$statistic),
  P_Value = c(dm_test_12$p.value, dm_test_13$p.value, dm_test_23$p.value)
)

# Gerar a tabela LaTeX com kable e fixar no local usando [H]
kable(dm_results,
      caption = "Diebold-Mariano Test Statistics and P-values for Model Comparisons",
      col.names = c("Comparison", "DM Statistic", "P-value"),
      format = "latex", booktabs = TRUE) %>%
  kable_styling(latex_options = c("hold_position"))
```

Table 2: Diebold-Mariano Test Statistics and P-values for Model Comparisons

Comparison	DM Statistic	P-value
Model 1 vs Model 2	-5.105502	8e-07
Model 1 vs Model 3	10.992908	0e+00
Model 2 vs Model 3	12.133412	0e+00

A low p-value in the Diebold-Mariano (DM) test indicates that there is a statistically significant difference in the forecast errors between the compared models. This means that one model provides more accurate forecasts than the other. In this case, the results suggest that model 3 performs better than both model 2 and model 1, as it has smaller forecast errors. Additionally, the comparison between model 1 and model 2 shows that model 1 is better than model 2, as it also has lower forecast errors. Thus, model 3 is the most accurate, followed by model 1, with model 2 being the least accurate.

```
# Realizar os testes Diebold-Mariano para os modelos com projeção recursiva
dm_test_12_rec <- dm.test(ferror_model1_rec, ferror_model2_rec, h = 1)
dm_test_13_rec <- dm.test(ferror_model1_rec, ferror_model3_rec, h = 1)
dm_test_23_rec <- dm.test(ferror_model2_rec, ferror_model3_rec, h = 1)

# Criar uma tabela com os resultados para os modelos com projeção recursiva
dm_results_rec <- data.frame(
  Comparisons = c("Model 1 Rec vs Model 2 Rec", "Model 1 Rec vs Model 3 Rec", "Model 2 Rec vs Model 3 R
  DM_Statistic = c(dm_test_12_rec$statistic, dm_test_13_rec$statistic, dm_test_23_rec$statistic),
  P_Value = c(dm_test_12_rec$p.value, dm_test_13_rec$p.value, dm_test_23_rec$p.value)
)

# Gerar a tabela LaTeX com kable e fixar no local usando [H]
kable(dm_results_rec,
      caption = "Diebold-Mariano Test Statistics and P-values for Recursive Projection Model Comparisons
      col.names = c("Comparison", "DM Statistic", "P-value"),
      format = "latex", booktabs = TRUE) %>%
  kable_styling(latex_options = c("hold_position"))
```

Table 3: Diebold-Mariano Test Statistics and P-values for Recursive Projection Model Comparisons

Comparison	DM Statistic	P-value
Model 1 Rec vs Model 2 Rec	-3.263101	0.001297
Model 1 Rec vs Model 3 Rec	10.454218	0.000000
Model 2 Rec vs Model 3 Rec	10.737803	0.000000

The same can be said about the recursive forecasts.

Now we proceed to Morgan-Granger-Newbold test:

```
# Função para calcular a estatística MGN e o p-valor
mgn_test <- function(erro_1, erro_2) {

  # Somar e subtrair os resíduos
  soma_erros <- erro_1 + erro_2
  diferenca_erros <- erro_1 - erro_2

  r <- sum(soma_erros * diferenca_erros) / sqrt(sum(soma_erros^2) * sum(diferenca_erros^2))
```

```r
  # Calcular o t-valor para o teste MGN
  t_stat <- r / sqrt((1 - r^2) / (length(erro_1) - 2))

  # Calcular o p-valor
  p_val <- 2 * pt(abs(t_stat), df = (length(erro_1) - 2), lower.tail = FALSE)

  # Retornar a estatística de teste e o p-valor
  return(list(t_stat = t_stat, p_value = p_val))
}

# Aplicar o teste MGN para cada par de modelos
result_12 <- mgn_test(ferror_model1, ferror_model2)
result_13 <- mgn_test(ferror_model1, ferror_model3)
result_23 <- mgn_test(ferror_model2, ferror_model3)

# Criar um DataFrame com os resultados
mgn_results <- data.frame(
  Comparison = c("Model 1 vs Model 2", "Model 1 vs Model 3", "Model 2 vs Model 3"),
  t_stat = c(result_12$t_stat, result_13$t_stat, result_23$t_stat),
  p_value = c(result_12$p_value, result_13$p_value, result_23$p_value)
)

# Gerar a tabela LaTeX com kable e fixar no local usando [H]
kable(mgn_results,
      caption = "Morgan-Granger-Newbold Test Results for Model Comparisons",
      col.names = c("Comparison", "t-statistic", "P-value"),
      format = "latex", booktabs = TRUE) %>%
  kable_styling(latex_options = c("hold_position"))
```

Table 4: Morgan-Granger-Newbold Test Results for Model Comparisons

Comparison	t-statistic	P-value
Model 1 vs Model 2	-5.290572	3e-07
Model 1 vs Model 3	28.730154	0e+00
Model 2 vs Model 3	32.576732	0e+00

The null hypothesis of the Morgan-Granger-Newbold (MGN) test is that there is no significant difference in the accuracy of the two competing forecast models, meaning their forecast errors are uncorrelated. If the null hypothesis is rejected, it indicates that one model outperforms the other.

In this case, similar to the results from the Diebold-Mariano test, the MGN test also suggests that model 3 provides more accurate forecasts compared to models 1 and 2. Additionally, the results show that model 1 performs better than model 2. Therefore, the conclusions drawn from the MGN test are consistent with those of the Diebold-Mariano test.

```r
# Aplicar o teste MGN para cada par de modelos com projeção recursiva
result_12_rec <- mgn_test(ferror_model1_rec, ferror_model2_rec)
result_13_rec <- mgn_test(ferror_model1_rec, ferror_model3_rec)
result_23_rec <- mgn_test(ferror_model2_rec, ferror_model3_rec)

# Criar um DataFrame com os resultados para as projeções recursivas
mgn_results_rec <- data.frame(
  Comparison = c("Model 1 Rec vs Model 2 Rec", "Model 1 Rec vs Model 3 Rec", "Model 2 Rec vs Model 3 Re
```

```
  t_stat = c(result_12_rec$t_stat, result_13_rec$t_stat, result_23_rec$t_stat),
  p_value = c(result_12_rec$p_value, result_13_rec$p_value, result_23_rec$p_value)
)

# Gerar a tabela LaTeX com kable e fixar no local usando [H]
kable(mgn_results_rec,
      caption = "Morgan-Granger-Newbold Test Results for Recursive Projection Model Comparisons",
      col.names = c("Comparison", "t-statistic", "P-value"),
      format = "latex", booktabs = TRUE) %>%
  kable_styling(latex_options = c("hold_position"))
```

Table 5: Morgan-Granger-Newbold Test Results for Recursive Projection Model Comparisons

Comparison	t-statistic	P-value
Model 1 Rec vs Model 2 Rec	-2.543686	0.0117325
Model 1 Rec vs Model 3 Rec	28.207438	0.0000000
Model 2 Rec vs Model 3 Rec	28.993129	0.0000000

The same can be said about the recursive forecasts.

2.2.1 Compare the models using MCS

```
# Estimating the models that are missing (as in the example)
model_4 <- lm(
  y0 ~  x0 +
        y1 +
        x1,
  data = dados_estimacao_rec
)
model_5 <- lm(
  y0 ~  x0 +
        y1 +
        x1 +
        y2 +
        x2,
  data = dados_estimacao_rec
)
model_6 <- lm(
      y0 ~ dummy_101 + dummy_166 + dummy_211 +
            x0 +
            dummy_101 * x0 +
            dummy_166 * x0 +
            dummy_211 * x0 +
            y1 +
            x1 +
            y2 +
            x2,
      data = dados_estimacao_rec
)

# Forecast para o model 4
forcast_model_4 <- model_4$coefficients[1] +
```

```
                  model_4$coefficients[2] * dados$x0 +
                  model_4$coefficients[3] * dados$y1 +
                  model_4$coefficients[4] * dados$x1

# Forecast para o model 5
forcast_model_5 <- model_5$coefficients[1] +
                  model_5$coefficients[2] * dados$x0 +
                  model_5$coefficients[3] * dados$y1 +
                  model_5$coefficients[4] * dados$x1 +
                  model_5$coefficients[5] * dados$y2 +
                  model_5$coefficients[6] * dados$x2

# Forecast para o model 6
forcast_model_6 <- model_6$coefficients[1] +
                  model_6$coefficients[2] * dados$dummy_101 +
                  model_6$coefficients[3] * dados$dummy_166 +
                  model_6$coefficients[4] * dados$dummy_211+
                  model_6$coefficients[5] * dados$x0 +
                  model_6$coefficients[6] * dados$y1 +
                  model_6$coefficients[7] * dados$x1 +
                  model_6$coefficients[8] * dados$y2  +
                  model_6$coefficients[9] *  dados$x2 +
                  model_6$coefficients[10] * dados$dummy_101 * dados$x0 +
                  model_6$coefficients[11] * dados$dummy_166 * dados$x0 +
                  model_6$coefficients[12] * dados$dummy_211 * dados$x0

# Cálculo dos erros de previsão
ferror_model4 <- tail(dados$y0, 200) - tail(forcast_model_4, 200)
ferror_model5 <- tail(dados$y0, 200) - tail(forcast_model_5, 200)
ferror_model6 <- tail(dados$y0, 200) - tail(forcast_model_6, 200)

# Criar o DataFrame com os erros de previsão dos 6 modelos
df_errors <- data.frame(
  Model_1 = ferror_model1,
  Model_2 = ferror_model2,
  Model_3 = ferror_model3,
  Model_4 = ferror_model4,
  Model_5 = ferror_model5,
  Model_6 = ferror_model6
)

# Exportar o DataFrame para um arquivo Excel
# write_xlsx(df_errors, "forecast_errors_models_1_to_6.xlsx")
```

Apologies for the inconvenience, but the rest is addressed in the Jupyter notebook that was sent.

2.3 4) Perform forecast combinations

```
# Ponderação simples das previsões dos três modelos
forecast_w <- (tail(forcast_model_1, 200) + tail(forcast_model_2, 200)
              + tail(forcast_model_3, 200))/3

# Ponderação simples das previsões recursivas dos três modelos
```

```r
forecast_w_rec <- (forcast_model_1_rec + forcast_model_2_rec + forcast_model_3_rec)/3

# Cálculo do RMSE para os modelos normais
rmse_model1 <- sqrt(mean((tail(forcast_model_1, 200) - tail(dados$y0, 200))^2))
rmse_model2 <- sqrt(mean((tail(forcast_model_2, 200) - tail(dados$y0, 200))^2))
rmse_model3 <- sqrt(mean((tail(forcast_model_3, 200) - tail(dados$y0, 200))^2))

# Cálculo do RMSE para os modelos recursivos
rmse_model1_rec <- sqrt(mean((forcast_model_1_rec - tail(dados$y0, 200))^2))
rmse_model2_rec <- sqrt(mean((forcast_model_2_rec - tail(dados$y0, 200))^2))
rmse_model3_rec <- sqrt(mean((forcast_model_3_rec - tail(dados$y0, 200))^2))

# Cálculo do RMSE para a ponderação simples dos modelos
rmse_forecast_w <- sqrt(mean((forecast_w - tail(dados$y0, 200))^2))
rmse_forecast_w_rec <- sqrt(mean((forecast_w_rec - tail(dados$y0, 200))^2))

# Parte 1: Modelos sem _rec

# Erros de previsão para os modelos sem _rec
ferror_model1 <- tail(dados$y0, 200) - tail(forcast_model_1, 200)
ferror_model2 <- tail(dados$y0, 200) - tail(forcast_model_2, 200)
ferror_model3 <- tail(dados$y0, 200) - tail(forcast_model_3, 200)

# Organizar as previsões dos modelos sem _rec em uma matriz
forecasts_non_rec <- cbind(tail(forcast_model_1, 200),
                           tail(forcast_model_2, 200),
                           tail(forcast_model_3, 200))

# Valores observados
actuals <- tail(dados$y0, 200)

# Inicializar o modelo de combinação para modelos sem _rec
model_non_rec <- mixture(model = "MLpol")

# Atualizar o modelo com os dados e calcular os pesos dinâmicos
model_non_rec <- predict(model_non_rec, forecasts_non_rec, actuals)

# Extrair os pesos dinâmicos para os modelos sem _rec
weights_dynamic_non_rec <- model_non_rec$weights

# Calcular o forecast ponderado dinamicamente para cada período (modelos sem _rec)
forecast_weighted_dynamic_non_rec <- rowSums(forecasts_non_rec * weights_dynamic_non_rec)

# Calcular o RMSE para os modelos sem _rec
rmse_dynamic_non_rec <- sqrt(mean((actuals - forecast_weighted_dynamic_non_rec)^2))

# Parte 2: Modelos com _rec

# Erros de previsão para os modelos com _rec
ferror_model1_rec <- tail(dados$y0, 200) - forcast_model_1_rec
ferror_model2_rec <- tail(dados$y0, 200) - forcast_model_2_rec
ferror_model3_rec <- tail(dados$y0, 200) - forcast_model_3_rec
```

```r
# Organizar as previsões dos modelos com _rec em uma matriz
forecasts_rec <- cbind(forcast_model_1_rec,
                       forcast_model_2_rec,
                       forcast_model_3_rec)

# Inicializar o modelo de combinação para modelos com _rec
model_rec <- mixture(model = "MLpol")

# Atualizar o modelo com os dados e calcular os pesos dinâmicos
model_rec <- predict(model_rec, forecasts_rec, actuals)

# Extrair os pesos dinâmicos para os modelos com _rec
weights_dynamic_rec <- model_rec$weights

# Calcular o forecast ponderado dinamicamente para cada período (modelos com _rec)
forecast_weighted_dynamic_rec <- rowSums(forecasts_rec * weights_dynamic_rec)

# Calcular o RMSE para os modelos com _rec
rmse_dynamic_rec <- sqrt(mean((actuals - forecast_weighted_dynamic_rec)^2))

# Criar uma tabela com os resultados de RMSE (incluindo o RMSE ponderado com pesos ótimos)
rmse_results_table <- data.frame(
  Model = c("Model 1", "Model 2", "Model 3",
            "Model 1 Rec", "Model 2 Rec", "Model 3 Rec",
            "Weighted Forecast (Non-Rec)", "Weighted Forecast (Rec)",
            "Optimal Weighted Forecast (Non-Rec)",
            "Optimal Weighted Forecast (Rec)"),
  RMSE = c(rmse_model1, rmse_model2, rmse_model3,
           rmse_model1_rec, rmse_model2_rec, rmse_model3_rec,
           rmse_forecast_w, rmse_forecast_w_rec,
           rmse_dynamic_non_rec, rmse_dynamic_rec)
)

# Gerar a tabela LaTeX com kable
library(knitr)
library(kableExtra)

kable(rmse_results_table,
      caption = "RMSE Results for Model Comparisons",
      col.names = c("Model", "RMSE"),
      format = "latex", booktabs = TRUE) %>%
  kable_styling(latex_options = c("hold_position"))
```

After attempts to combine the forecasts, we were unable to outperform model 3 (both recursive and non-recursive) based on RMSE. While the combined forecasts showed reasonable accuracy, they couldn't surpass the performance of model 3, which remained the most accurate.

2.3.1 Compare the models using MCS

```python
import numpy as np
import pandas as pd
from model_confidence_set import ModelConfidenceSet

# Passo 1: Carregar o arquivo Excel em um DataFrame
```

Table 6: RMSE Results for Model Comparisons

Model	RMSE
Model 1	2.7161550
Model 2	3.0506200
Model 3	0.6531095
Model 1 Rec	2.6800277
Model 2 Rec	2.7415971
Model 3 Rec	0.6524572
Weighted Forecast (Non-Rec)	1.9705463
Weighted Forecast (Rec)	1.8741939
Optimal Weighted Forecast (Non-Rec)	0.7449397
Optimal Weighted Forecast (Rec)	0.7443822

```
file_path = r"..\ps3\forecast_errors_models_1_to_6.xlsx"
df_errors = pd.read_excel(file_path)

# Passo 2: Calcular os erros quadráticos (quadrado dos erros de projeção)
df_squared_errors = df_errors ** 2  # Isso eleva ao quadrado todos os erros no DataFrame

# Passo 3: Configurar e rodar o MCS usando os erros quadráticos

# Converter os erros para formato numpy array
losses = df_squared_errors.values

# Inicializar o procedimento MCS (5000 bootstraps, nível de confiança de 5%)
mcs = ModelConfidenceSet(losses, n_boot=5000, alpha=0.05, show_progress=True)

# Calcular o MCS
mcs.compute()
```

```
## Bootstrapping:   0%|          | 0/5000 [00:00<?, ?it/s]Bootstrapping:  55%|#####5    | 2757/5000 [00
## Computing MCS:   0%|          | 0/5 [00:00<?, ?model/s]Computing MCS: 100%|##########| 5/5 [00:00<00
```

```
# Recuperar os resultados como um DataFrame
results = mcs.results()

# Exibir os resultados
print(results)
```

```
##         pvalues    status
## models
## 1           0.0  excluded
## 2           0.0  excluded
## 6           0.0  excluded
## 5           0.0  excluded
## 4           0.0  excluded
## 3           1.0  included
```

The Model Confidence Set (MCS) procedure is a sequential testing method designed to compare multiple models and identify a subset of the best-performing ones based on their predictive accuracy. The procedure eliminates the worst model at each step until only models with similar predictive abilities remain. This is based on the null hypothesis of Equal Predictive Ability (EPA), which tests whether the difference in

performance between models is statistically significant or not.

In your case, models 1, 2, 4, 5, and 6 were excluded due to their low p-values, indicating they performed significantly worse than model 3, which was retained in the confidence set. Model 3, which includes dummies and dynamic terms, was the only one with a p-value of 1, showing that it is statistically indistinguishable from the best model and remains the top performer in your analysis.

3 Part II

Obs: in this part, we'll the "varr" package, an authorial package for time series graphs.

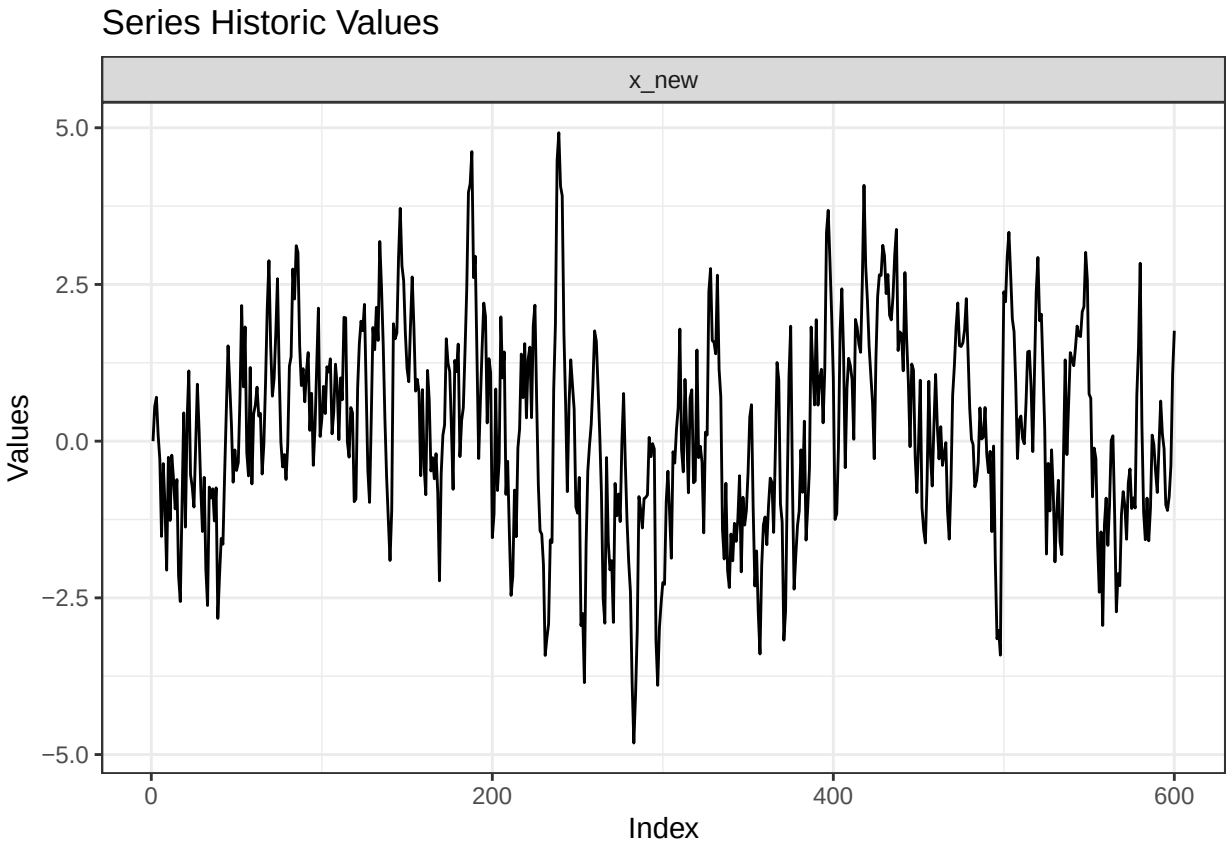
First, create the subset of data for estimation:

```
data_est <- slice(dados, 101:(nrow(dados) - 200))
n <- nrow(data_est)
```

3.1 Stationarity Analisys

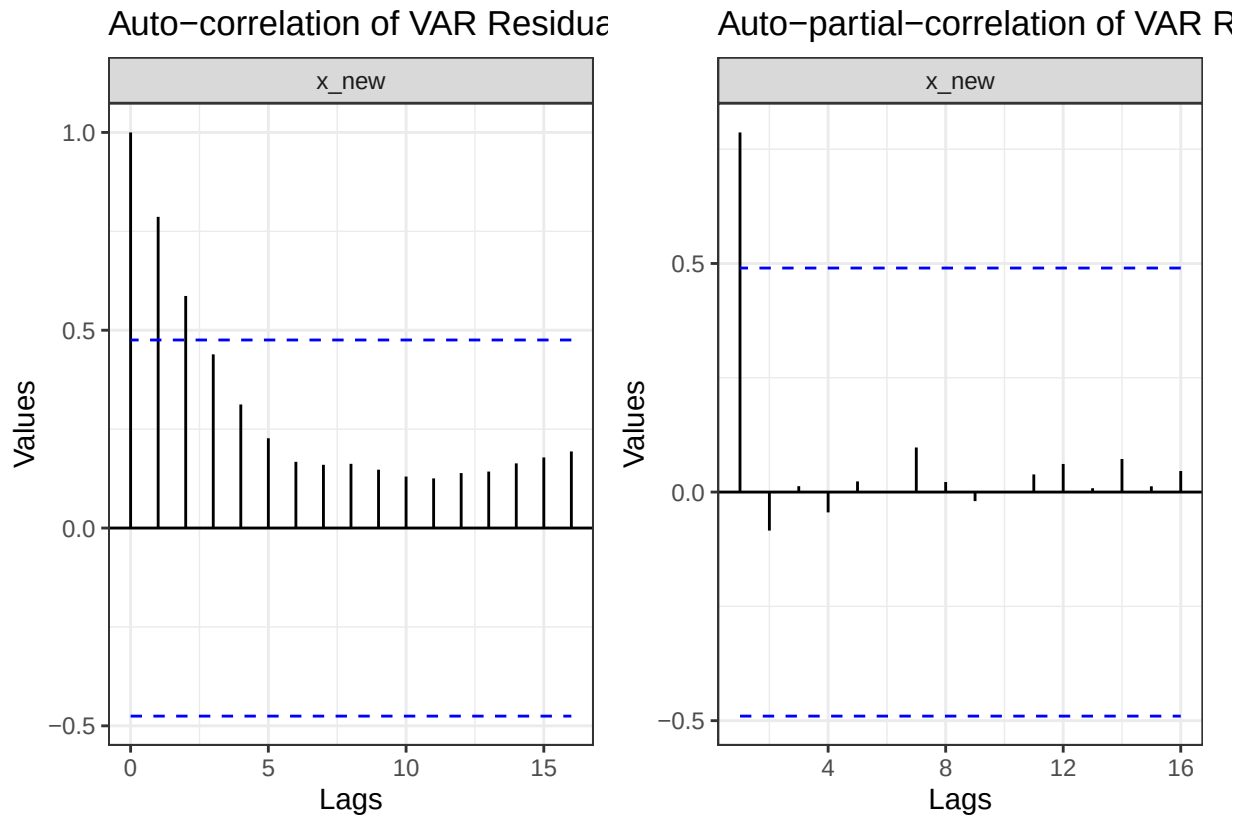
First, lets plot the data.

```
ggvar_history(dados, "x_new")
```



We need further evidence, but it does not seem to have a stochastic trend.

```
ggvar_acf(dados, "x_new") + ggvar_acf(dados, "x_new", type = "partial")
```



We can see that there is some autocorrelation, but it is small and quckly decreases. Also, no partial correlation. Lets use a ADF test to consolidate our hypothesis.

```
output_dftest(dados$x_new)
```

Table 7:

	Lag	Type 1	Type 2	Type 3
1	0	-8.34 (0.01)	-8.4 (0.01)	-8.39 (0.01)
2	1	-8.57 (0.01)	-8.63 (0.01)	-8.63 (0.01)
3	2	-7.98 (0.01)	-8.04 (0.01)	-8.04 (0.01)
4	3	-7.9 (0.01)	-7.98 (0.01)	-7.97 (0.01)
5	4	-7.32 (0.01)	-7.4 (0.01)	-7.4 (0.01)
6	5	-6.98 (0.01)	-7.07 (0.01)	-7.07 (0.01)

The test type that we are more interested is 1, the one without intercept nor trend, as it seems to match the data best. It, and all others, present statistical evidence of stationarity.

4 ARIMA Selection

Now, we can run a loop creating models for several values of AR and MA lag, p and q. The maximum lag was 4, as more that that seems to be overparametrization.

```r
orders <- list(p = 1:4, q = 1:4)
info_table <- list()

i <- 1
for (p in orders$p) {
  for (q in orders$q) {
    mod <- arima(data_est$x_new, c(p, 0, q))
    info_table[[i]] <- c(p = p, q = q, AIC = AIC(mod), BIC = BIC(mod))
    i <- i + 1
  }
}

bind_rows(info_table) %>%
  kable()
```

p	q	AIC	BIC
1	1	879.5289	894.3440
1	2	881.4908	900.0097
1	3	881.4618	903.6845
1	4	882.6542	908.5807
2	1	881.6133	900.1322
2	2	880.7392	902.9619
2	3	882.6894	908.6159
2	4	885.1446	914.7748
3	1	880.5935	902.8162
3	2	881.8133	907.7397
3	3	873.4093	903.0395
3	4	886.5983	919.9323
4	1	883.5140	909.4405
4	2	873.3471	902.9774
4	3	884.3695	917.7035
4	4	874.8268	911.8646

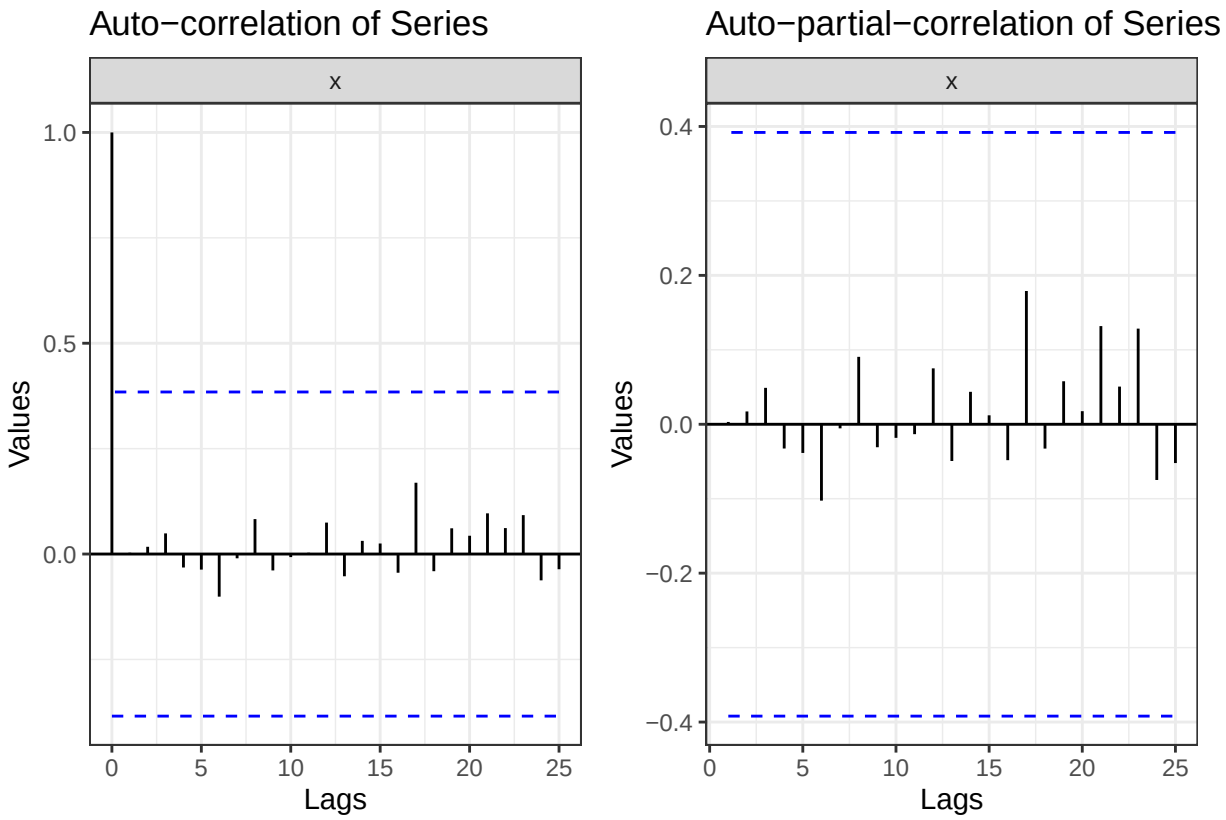
We can see that, by BIC, the best model was the simplest one, $(1, 1)$. By AIC, $(4, 2)$ is a strong contender, but this information criteria is famous for overparametrization. Thus, we choose the $(1, 1)$ model.

4.1 Best Model Diagnostics

Now, lets look at some diagnoses.

4.1.1 Autocorrelation of Residuals

```r
mod_arma <- arima(data_est$x_new, c(1, 0, 1))

acf_setup.Arima <- varr:::acf_setup.varest #creating a method for Arima class
ggvar_acf(mod_arma) + ggvar_acf(mod_arma, type = "partial")
```



We can see that there is no statistical evidence of autocorrelation, and both functions look very similar, which is a good sign for our specification. To get a more concise result, lets do the BG test of serial correlation:

```
diff_fill <- \(x, n = 1) c(rep(NA, n), diff(x, n))

mod_bg <- lm(x ~ diff_fill(x) + diff_fill(x, 2),
  as.data.frame(residuals(mod_arma))
)

stargazer(mod_bg)
```

% Table created by stargazer v.5.2.3 by Marek Hlavac, Social Policy Institute. E-mail: marek.hlavac at gmail.com % Date and time: qui, set 05, 2024 - 22:48:12

We find evidence to support our hypothesis.

4.1.2 ARCH Test

Lets see if our variance presents either ARCH effects:

```
FinTS::ArchTest(residuals(mod_arma), lags = 1)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  residuals(mod_arma)
## Chi-squared = 3.0065, df = 1, p-value = 0.08293
```

At the 5% level, we reject the null, but at the 10% we would. While this is not too worrisome, any hypothesis

Table 9:

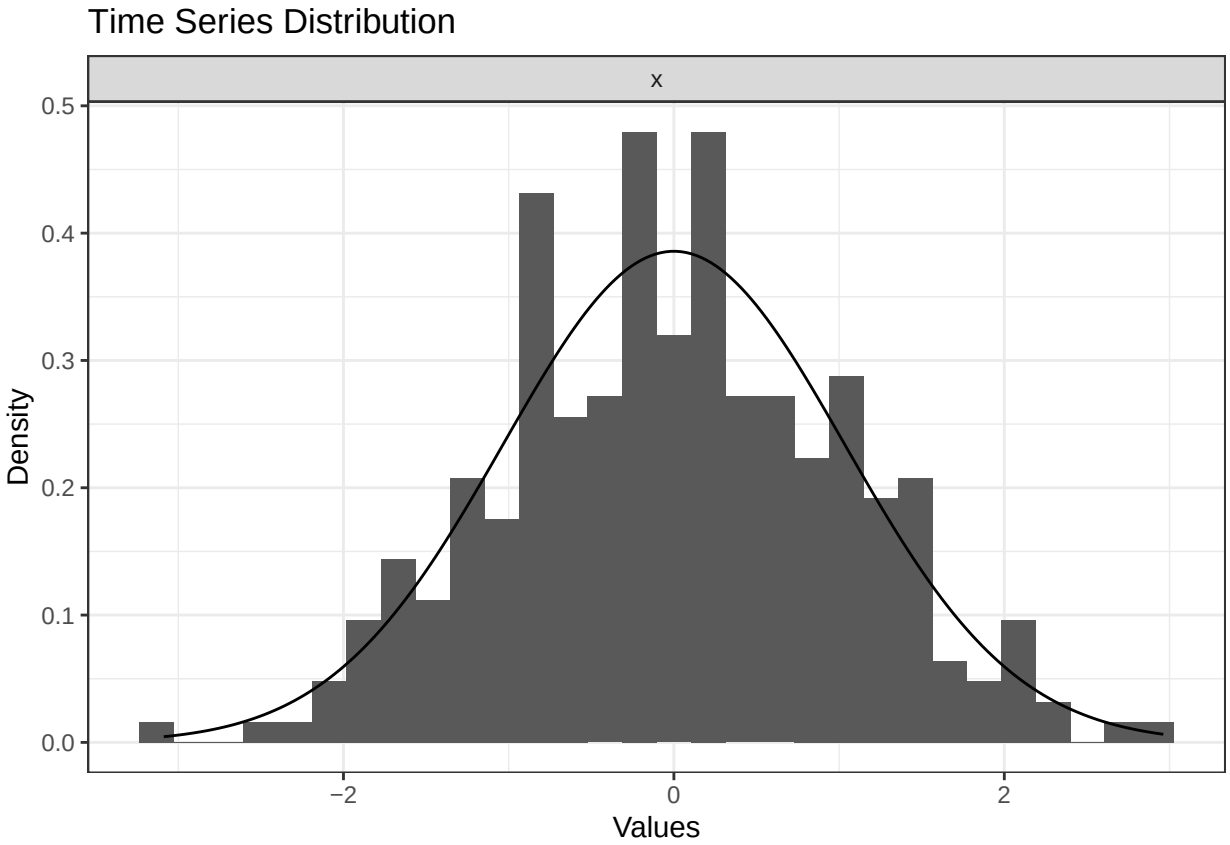
	Dependent variable:
	x
diff_fill(x)	0.337^{***}
	(0.028)
diff_fill(x, 2)	0.332^{***}
	(0.028)
Constant	-0.003
	(0.035)
Observations	298
R^2	0.661
Adjusted R^2	0.659
Residual Std. Error	0.605 (df = 295)
F Statistic	288.228^{***} (df = 2; 295)
Note:	*p<0.1; **p<0.05; ***p<0.01

tests should be taken with a grain of salt.

4.1.3 JB Test

For the test of the series distribution, we can plot its histogram against a normal curve:

```
ggvar_distribution(as.data.frame(residuals(mod_arma)))
```



It seems to follow a normal distribution, which can be confirmed in the JB test, as we dont reject the null:

```
tseries::jarque.bera.test(residuals(mod_arma))
```

```
##
##  Jarque Bera Test
##
## data:  residuals(mod_arma)
## X-squared = 0.67896, df = 2, p-value = 0.7121
```

4.1.4 White Test

Finally we don't have evidence to support non-linearity on the mean, as the White Test shows:

```
tseries::white.test(residuals(mod_arma))
```

```
White Neural Network Test
```

data: residuals(mod_arma) X-squared = 2.7514, df = 2, p-value = 0.2527

5 Forecasts

We ran the standard, and 1/2-step ahead recursive forecasts:

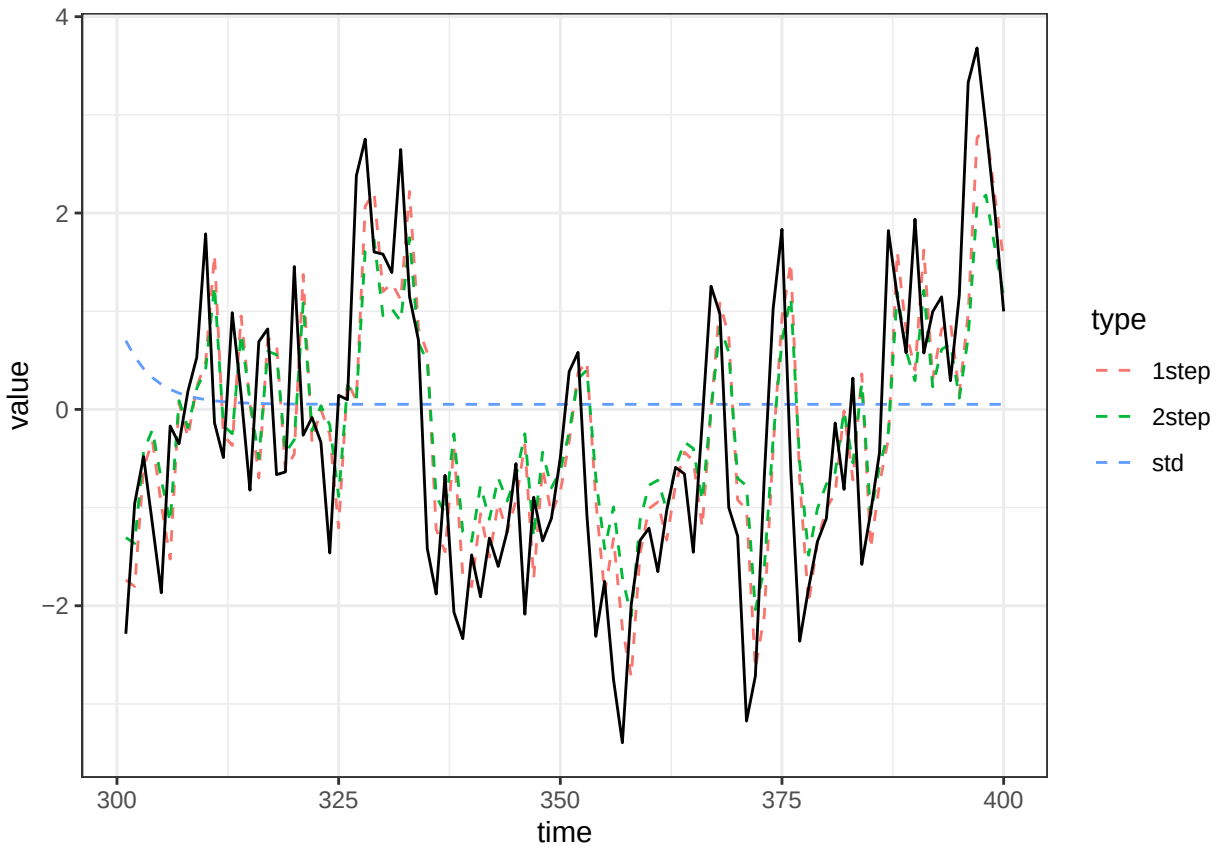
```
pred_std <- predict(mod_arma, n.ahead = 100)$pred

pred_rec1 <- c()
pred_rec2 <- c()
```

```r
for (i in 1:100) {
  mod_rec <- arima(data_est$x_new[1:(199 + i)], order = c(1, 0, 1))
  pred_rec <- predict(mod_rec, n.ahead = 2)$pred
  pred_rec1[i] <- pred_rec[1]
  pred_rec2[i] <- pred_rec[2]
}
```

Now, we can analyze their results

5.1 Plots

```r
data_g <- tibble(
  time = rep(301:400, 3),
  x_new = rep(data_est[201:300,]$x_new, 3),
  value = c(pred_std, pred_rec1, pred_rec2),
  type = rep(c("std", "1step", "2step"), each = 100)
)

ggplot() +
  geom_line(aes(time, value, color = type), data_g, linetype = 2) +
  geom_line(aes(X, x_new), data_est[201:300,])
```



5.2 RMSE & MAE

```r
data_g %>%
  group_by(type) %>%
  summarise(
    RMSE = mean((value - x_new)^2),
    MAE = mean(abs(value - x_new))
  ) %>%
  kable()
```

type	RMSE	MAE
1step	1.063552	0.8211135
2step	1.088350	0.8294157
std	2.364934	1.2928582

We can see that the std. prediction have very less information, and is not capable of such a long prediction window. As expected, the 2-step forecast yields a slightly worse result than the 1-step.