



PROVA 2 DE PROCESSO ESTOCÁSTICOS

Rafael de Acypreste (200060023) e Rafael Lira (190115858)

Professor Felipe Quintino

1Questão 1

1.1 Aplicação ao modelo empírico

Trata-se de um modelo para avaliar as probabilidades de transição entre os estados de precipitação de chuvas.

```
# Importing data
dados <-
  read.delim("dados.txt",
    header = TRUE,
    sep = ";"
  ) |>
  select(-X) |>
  filter(!is.na(Precipitacao))

# Summary statistics
dados$Precipitacao |> summary()
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.000	0.000	0.000	4.173	2.400	131.000

O primeiro passo é discretizar a variável de precipitação, que é feita com a função `cut()` do pacote `base`. Para esse exemplo, a variável será dividida em 3 categorias: sem chuva (precipitação até 0, 1), chuva fraca (precipitação maior que 0, 1 e menor que 10) e chuva forte.

```
# Discretization of the variable
quantiles <- quantile(dados$Precipitacao,
  probs = seq(0.7, 0.9, length.out = 3)
)

breaks <- c(-Inf, 0.001, quantiles, Inf)

state_labels <- factor(
  c(
    "sem chuva",
    "garoa",
    "chuva fraca",
    "chuva moderada",
    "chuva forte"
  ),
  levels = c(
    "sem chuva",
    "garoa",
    "chuva fraca",
    "chuva moderada",
    "chuva forte"
  )
)
```

```
# Discretization
dados <-
  dados |>
  # Discretization
  mutate(rain_status = cut(Precipitacao,
    breaks = breaks,
    labels = state_labels
  ))
```

Para esse exemplo, serão separadas as 10 últimas observações para avaliar as estimativas.

```
dados_teste <- tail(dados, 10)
dados_treinamento <- dados[1:(nrow(dados) - 10), ]
```

Para estimar as transições de estado, é necessário criar uma variável que identifique o estado atual e o estado seguinte. Para isso, é necessário criar uma variável defasada, que pode ser feita com a função `lag()` do pacote `dplyr`. Depois disso, basta avaliar as proporções das transições de estado.

```
# Creating the lagged variable
transicoes_chuva <-
  dados_treinamento |>
  # Lag variable
  mutate(rain_status_lag = lag(rain_status)) |>
  # Exclude the last state
  filter(!is.na(rain_status_lag)) |>
  # Count the transitions
  count(rain_status, rain_status_lag) |>
  # Calculates the estimator
  mutate(
    Prop = round(n / sum(n), digits = 3),
    .by = rain_status
  )
```

A matriz de transição estimada entre os estados sem chuva, garoa, chuva fraca, chuva moderada, chuva forte, nesta ordem, é dada por:

$$P = \begin{pmatrix} 0.822 & 0.041 & 0.051 & 0.045 & 0.04 \\ 0.334 & 0.12 & 0.186 & 0.177 & 0.183 \\ 0.306 & 0.133 & 0.187 & 0.184 & 0.19 \\ 0.306 & 0.121 & 0.171 & 0.193 & 0.21 \\ 0.282 & 0.118 & 0.174 & 0.204 & 0.223 \end{pmatrix} \quad (1.1)$$

Agora, pode-se recuperar a matriz de transição para fazer as estimativas de transição de estado.

```
# Transition matrix
matriz_transicao <-
  transicoes_chuva |>
  select(-n) |>
  pivot_wider(
```

```

names_from = rain_status_lag,
values_from = Prop
) |>
column_to_rownames("rain_status") |>
as.matrix()

ultimo_estado <- dados_treinamento |>
tail(1) |>
pull(rain_status)

```

Com a matriz de transição, basta considerar o último estado dos dados (garoa) — consequência da propriedade de Markov — de treinamento para fazer as estimativas de transição de estado.

```

simula_cadeia_markov <- function(n = 10,
                                valor_inicial,
                                matriz_transicao,
                                estados) {

  P <- matriz_transicao
  y <- valor_inicial

  # Simulation of the stochastic process
  for (i in 1:n) {
    # Sample of the next state
    y[i + 1] <- sample(estados, size = 1, prob = P[y[i], ])
  }

  return(y[-1])
}

# Execution of the function
previsoes <- simula_cadeia_markov(
  valor_inicial = ultimo_estado,
  matriz_transicao = matriz_transicao,
  estados = state_labels,
  n = 10
)

```

E, então, pode-se comparar as previsões com os dados de teste. Para o gráfico, os acertos são indicados pela linha tracejada vermelha.

```

comparacao <-
  data.frame(
    observado = dados_teste$rain_status,
    previsao = previsoes
  )

# Imprime a tabela
comparacao

```

	observado	previsao
1	chuva moderada	chuva moderada
2	chuva moderada	chuva fraca
3	chuva forte	chuva moderada

```

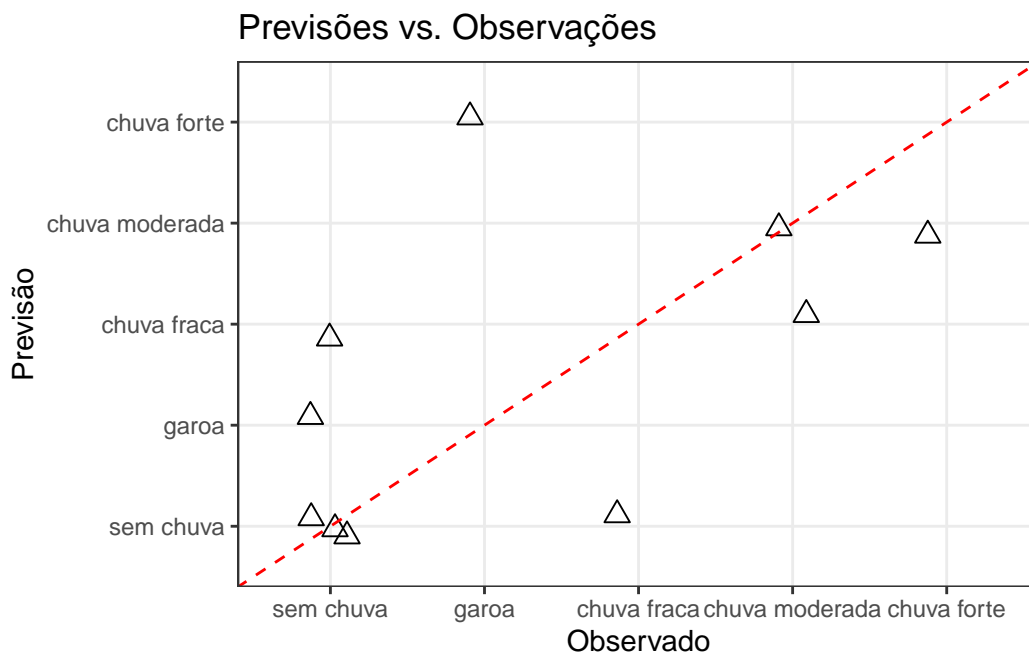
4      sem chuva      garoa
5      sem chuva      chuva fraca
6      sem chuva      sem chuva
7      sem chuva      sem chuva
8      sem chuva      sem chuva
9      chuva fraca    sem chuva
10     garoa          chuva forte

```

```

# Constroi o gráfico
comparacao |>
  ggplot(aes(x = observado, y = previsao)) +
  geom_jitter(
    size = 3, shape = 2,
    width = 0.15, height = 0.15
  ) +
  geom_abline(
    intercept = 0,
    slope = 1,
    color = "red",
    linetype = "dashed"
  ) +
  theme_bw() +
  labs(
    x = "Observado",
    y = "Previsão",
    title = "Previsões vs. Observações"
  )

```



Questão 2

Aplicação ao modelo empírico

Trata-se de um modelo para avaliar o comportamento dos preços de fechamentos dos valores das ações do BBAS3 no ano de 2023.

```
# Define the stock symbol and specify the start and end dates
stock_symbol <- "BBAS3.SA"
start_date <- "2023-01-01"
end_date <- "2024-01-01"

# Use getSymbols to fetch historical stock data
getSymbols(stock_symbol,
  src = "yahoo",
  from = start_date,
  to = end_date
)
```

```
[1] "BBAS3.SA"
```

```
# Check the loaded data and get the closing values
stock_values <- as.vector(Cl(get(stock_symbol)))

# Summary statistics
summary(stock_values)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
32.64	40.78	47.00	45.49	49.07	55.39

Obtendo o número de observações no vetor de valores da ação, é possível gerar simulações do movimento Browniano e do processo de Poisson com a mesma quantidade de pontos que a base de dados. Considerando um intervalo de 0 a 1, em anos, é gerado um vetor t relativo ao tempo decorrido do início da contagem ao momento de cada observação. Para simular o movimento Browniano, basta fazer a soma cumulativa de n valores da distribuição Normal padrão, enquanto para o processo de Poisson é feita a soma de valores da distribuição Poisson com parâmetro $\lambda = 1/n$.

```
n <- length(stock_values) - 1
t <- seq(0, 1, length.out = n + 1)
B <- c(0, cumsum(rnorm(n, mean = 0, sd = 1)))
N <- c(0, cumsum(rpois(n, lambda = 1 / n)))
```

Em seguida, é criada uma função para prever a k -ésima observação do modelo, usando os tempos, o histórico do processo, o parâmetro θ e o valor do processo $\xi(t_k)$

```
simulate_Xtk <- function(t, X, theta, csi) {
  timeline <- as.vector(t)
  history <- as.vector(X)
  csi <- as.vector(csi)
```

```
stop

  if (length(timeline) != length(history) ||
      length(timeline) != length(csi) ||
      length(history) != length(csi)) {
    stop("The timeline, the history and the csi vector must have the same length!")
  }

  n <- length(timeline)

  tj <- timeline[-1]
  tj_1 <- timeline[-n]
  Xtj_1 <- history[-n]
  fatork <- Xtj_1 * (tj - tj_1)
  sumk <- cumsum(fatork)
  Xtk <- Xtj_1 - theta * sumk + csi[-1]
  return(Xtk)
}
```

Para estimar o parâmetro θ por meio do método dos mínimos quadrados, é criada uma função que recebe os mesmos *inputs* da função de simulação, porém retornando a soma de quadrados do resíduo.

```
least_squares <- function(t, X, theta, csi) {
  observed_values <- X[-1]
  predicted_values <- simulate_Xtk(t, X, theta, csi)

  return(sum((observed_values - predicted_values)^2))
}
```

Utilizando a função `optim`, e escolhendo um valor inicial para θ , é possível encontrar o ponto onde a soma de quadrados é mínima. Assim, são gerados os estimadores para cada o movimento Browniano e para o processo de Poisson.

```
initial_theta <- 100

(estim_theta_browniano <- optim(
  par = initial_theta,
  fn = least_squares,
  X = stock_values,
  t = t,
  csi = B ## Trajetória do movimento Browniano
)$par)
```

```
[1] 0.8203125
```

```
(estim_theta_poisson <- optim(
  par = initial_theta,
  fn = least_squares,
  X = stock_values,
  t = t,
```

```
csi = N ## Trajetória do processo de Poisson
)$par)
```

```
[1] 0
```

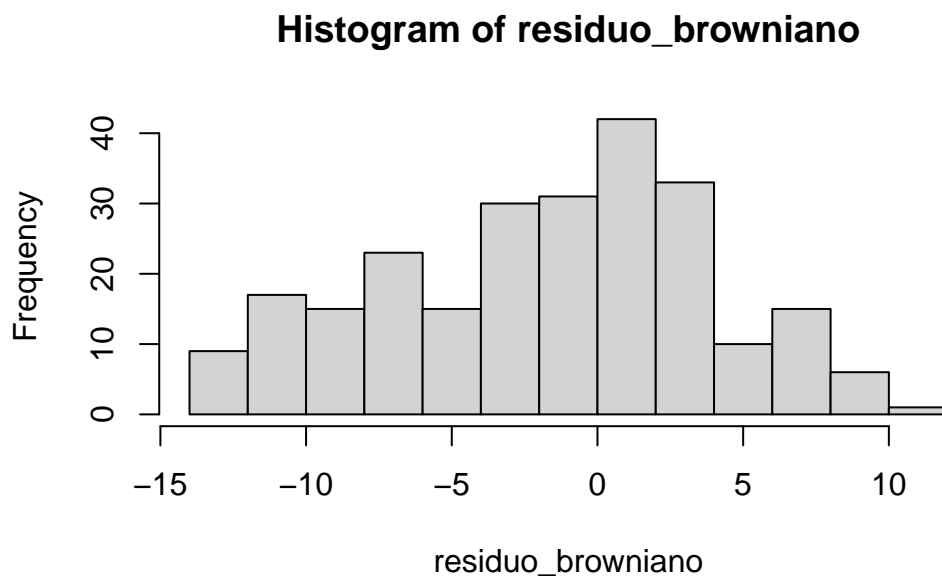
```
X_prev_browniano <- simulate_Xtk(t, stock_values, estim_theta_browniano, B)
X_prev_poisson <- simulate_Xtk(t, stock_values, estim_theta_poisson, N)

residuo_browniano <- stock_values[-1] - X_prev_browniano
shapiro.test(residuo_browniano)
```

Shapiro-Wilk normality test

```
data: residuo_browniano
W = 0.97536, p-value = 0.0002736
```

```
hist(residuo_browniano)
```

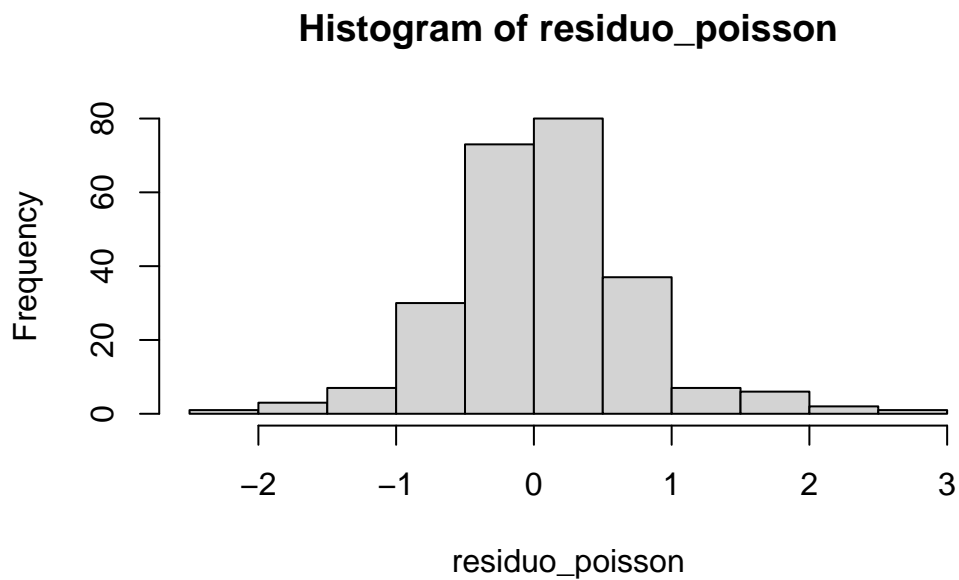


```
residuo_poisson <- stock_values[-1] - X_prev_poisson
shapiro.test(residuo_poisson)
```

Shapiro-Wilk normality test

```
data: residuo_poisson
W = 0.9795, p-value = 0.001227
```

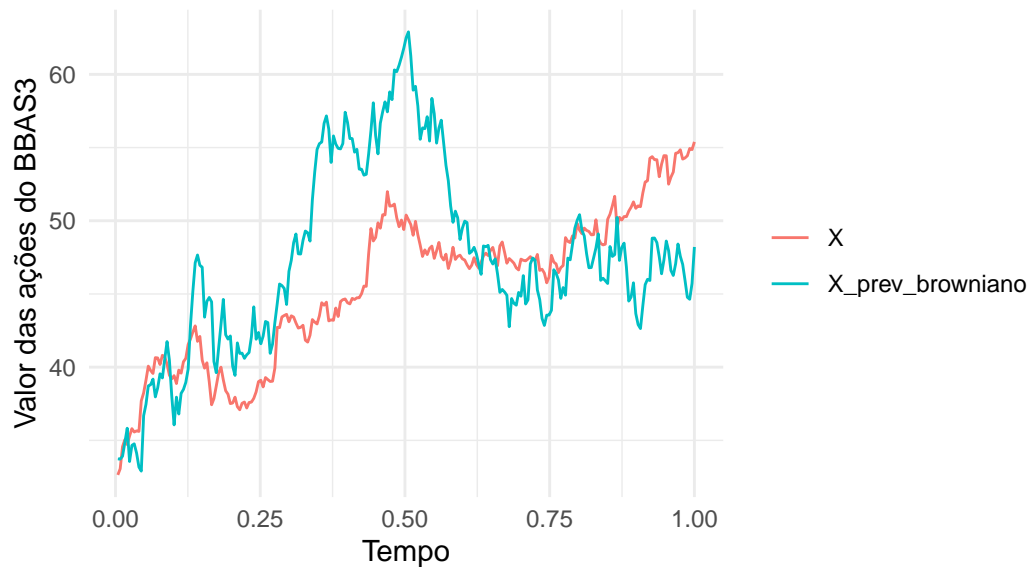
```
hist(residuo_poisson)
```

```
dados_sim <- data.frame(  
  t = t[-1],  
  X = stock_values[-1],  
  X_prev_browniano = X_prev_browniano,  
  X_prev_poisson = X_prev_poisson  
) |>  
pivot_longer(cols = c(X, X_prev_browniano, X_prev_poisson),  
             names_to = "Variavel",  
             values_to = "Valor")
```

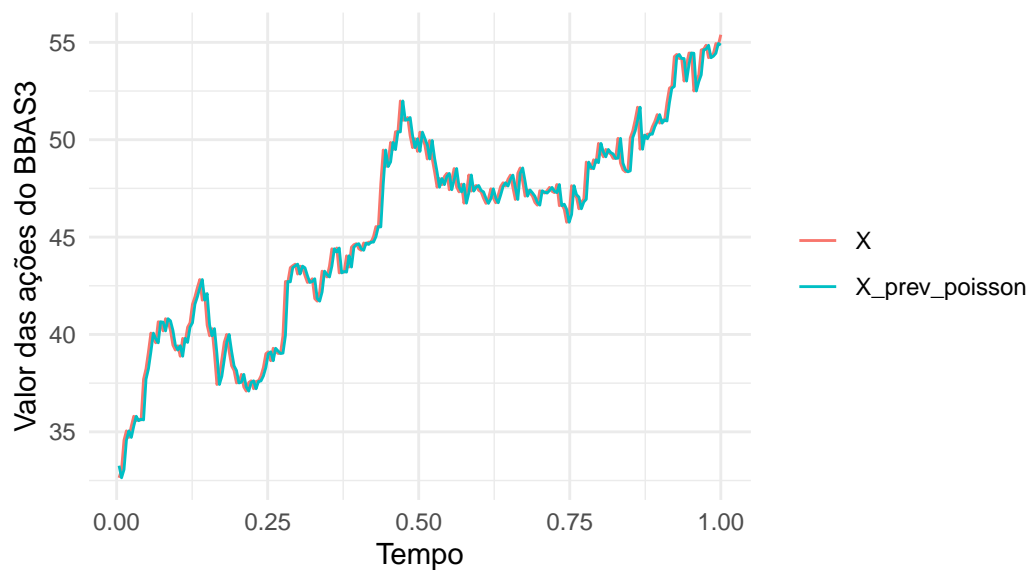
```
dados_sim |>  
  filter(Variavel != "X_prev_poisson") |>  
  ggplot() +  
  geom_line(aes(x = t, y = Valor, color = Variavel)) +  
  labs(x = "Tempo",  
       y = "Valor das ações do BBAS3",  
       color = NULL,  
       title = "Ajuste do modelo de movimento Browniano  
               aos dados de ações do BBAS3 em 2023") +  
  theme_minimal()
```

Ajuste do modelo de movimento Browniano aos dados de ações do BBAS3 em 2023



```
dados_sim |>
  filter(Variavel != "X_prev_browniano") |>
  ggplot() +
  geom_line(aes(x = t, y = Valor, color = Variavel)) +
  labs(x = "Tempo",
       y = "Valor das ações do BBAS3",
       color = NULL,
       title = "Ajuste do modelo de processos de Poisson
               aos dados de ações do BBAS3 em 2023") +
  theme_minimal()
```

Ajuste do modelo de processos de Poisson aos dados de ações do BBAS3 em 2023



Previsão de valores de 2024

```
# Define the stock symbol and specify the start and end dates
start_date_valid <- "2024-01-01"
end_date_valid <- "2024-01-15"

# Use getSymbols to fetch historical stock data
getSymbols(stock_symbol,
  src = "yahoo",
  from = start_date_valid,
  to = end_date_valid
)
```

```
[1] "BBAS3.SA"
```

```
# Check the loaded data and get the closing values
stock_values_valid <- as.vector(Cl(get(stock_symbol)))

# Summary statistics
summary(stock_values)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
32.64	40.78	47.00	45.49	49.07	55.39

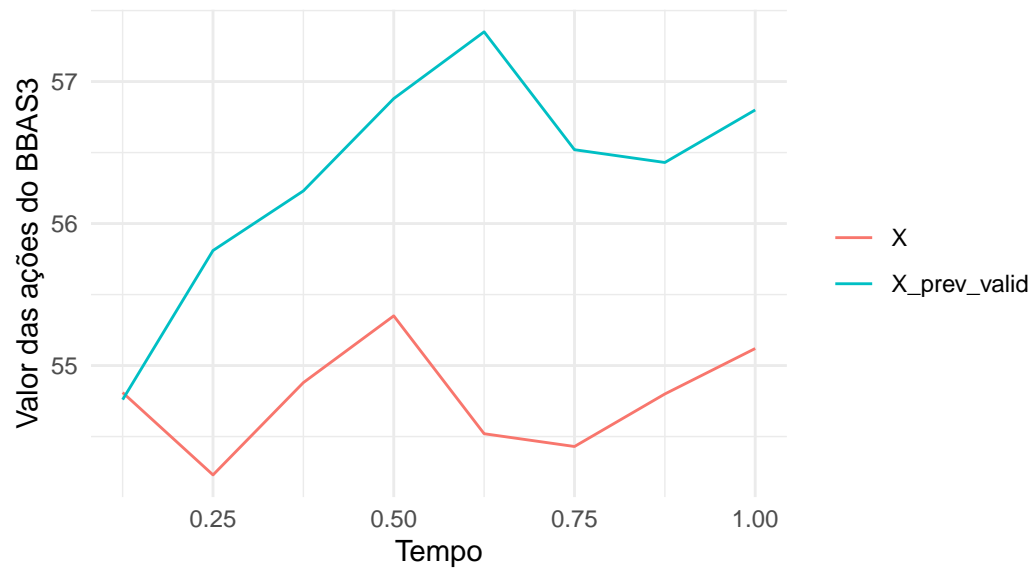
```
n_valid <- length(stock_values_valid) - 1
t_valid <- seq(0, 1, length.out = n_valid + 1)
N_valid <- c(0, cumsum(rpois(n_valid, 1 / n_valid)))
```

```
X_prev_valid <- simulate_Xtk(t_valid, stock_values_valid, estim_theta_poisson, N_valid)
```

```
dados_valid <- data.frame(
  t = t_valid[-1],
  X = stock_values_valid[-1],
  X_prev_valid = X_prev_valid
) |>
pivot_longer(cols = c(X, X_prev_valid),
  names_to = "Variavel",
  values_to = "Valor")
```

```
dados_valid |>
  ggplot() +
  geom_line(aes(x = t, y = Valor, color = Variavel)) +
  labs(x = "Tempo",
    y = "Valor das ações do BBAS3",
    color = NULL,
    title = "Ajuste do modelo de processos de Poisson aos dados
      de ações do BBAS3 nas primeiras semanas de 2024") +
  theme_minimal()
```

Ajuste do modelo de processos de Poisson aos dados de ações do BBAS3 nas primeiras semanas de 2024



Questão 2 Acypreste

Aplicação ao modelo empírico

Trata-se de um modelo para avaliar o comportamento dos preços de fechamentos dos valores das ações do BBAS3 no ano de 2023.

```
# Define the stock symbol and specify the start and end dates
stock_symbol <- "BBAS3.SA"
start_date <- "2023-01-01"
end_date <- "2024-01-01"

# Use getSymbols to fetch historical stock data
getSymbols(stock_symbol,
  src = "yahoo",
  from = start_date,
  to = end_date
)
```

```
[1] "BBAS3.SA"
```

```
# Check the loaded data and get the closing values
stock_values <- as.vector(Cl(get(stock_symbol)))

# Summary statistics
summary(stock_values)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
32.64	40.78	47.00	45.49	49.07	55.39

Obtendo o número de observações no vetor de valores da ação, é possível gerar simulações do movimento Browniano e do processo de Poisson com a mesma quantidade de pontos que a base de dados. Considerando um intervalo de 0 a 1, em anos, é gerado um vetor t relativo ao tempo decorrido do início da contagem ao momento de cada observação. Para simular o movimento Browniano, basta fazer a soma cumulativa de n valores da distribuição Normal padrão, enquanto para o processo de Poisson é feita a soma de valores da distribuição Poisson com parâmetro $\lambda = 1/n$.

```
n <- length(stock_values) - 1
t <- seq(0, 1, length.out = n + 1)
B <- c(0, cumsum(rnorm(n, mean = 0, sd = 1)))
N <- c(0, cumsum(rpois(n, lambda = 1)))
N_compensated <- c(0, cumsum(rpois(n, lambda = 1)) - seq(1, n, by = 1))
```

Em seguida, é criada uma função para prever a k -ésima observação do modelo, usando os tempos, o histórico do processo, o parâmetro θ e o valor do processo $\xi(t_k)$

```
simulate_Xtk <- function(t, X, theta, csi) {
  timeline <- as.vector(t)
  history <- as.vector(X)
```

```

csi <- as.vector(csi)
stop

if (length(timeline) != length(history) ||
    length(timeline) != length(csi) ||
    length(history) != length(csi)) {
  stop("The timeline, the history and the csi vector must have the same length!")
}

n <- length(timeline)

tj <- timeline[-1]
tj_1 <- timeline[-n]
Xtj_1 <- history[-n]

fatork <- rep(0, n - 1)
Xtk <- rep(0, n - 1)

for(j in 1:(n-1)){

  fatork[j] <- Xtj_1[j] * (tj[j] - tj_1[j])

  sumk <- cumsum(fatork[1:j])[j]

  Xtk[j] <- Xtj_1[j] - theta * sumk + csi[j]

}

return(Xtk)
}

```

Para estimar o parâmetro θ por meio do método dos mínimos quadrados, é criada uma função que recebe os mesmos *inputs* da função de simulação, porém retornando a soma de quadrados do resíduo.

```

least_squares <- function(t, X, theta, csi) {
  observed_values <- X[-1]
  predicted_values <- simulate_Xtk(t, X, theta, csi)

  return(sum((observed_values - predicted_values)^2))
}

```

Utilizando a função `optim`, e escolhendo um valor inicial para θ , é possível encontrar o ponto onde a soma de quadrados é mínima. Assim, são gerados os estimadores para cada o movimento Browniano e para o processo de Poisson.

```
initial_theta <- 100

(estim_theta_browniano <- optim(
  par = initial_theta,
  fn = least_squares,
  X = stock_values,
  t = t,
  csi = B ## Trajetória do movimento Browniano
)$par)
```

[1] 0.01953125

```
(estim_theta_poisson <- optim(
  par = initial_theta,
  fn = least_squares,
  X = stock_values,
  t = t,
  csi = N ## Trajetória do processo de Poisson
)$par)
```

[1] 5.859375

```
(estim_theta_poisson_compensated <- optim(
  par = initial_theta,
  fn = least_squares,
  X = stock_values,
  t = t,
  csi = N_compensated ## Trajetória do processo de Poisson
)$par)
```

[1] -0.5078125

```
X_prev_browniano <-
  simulate_Xtk(t, stock_values, estim_theta_browniano, B)
X_prev_poisson <-
  simulate_Xtk(t, stock_values, estim_theta_poisson, N)
X_prev_poisson_compensated <-
  simulate_Xtk(t, stock_values, estim_theta_poisson_compensated, N_compensated)

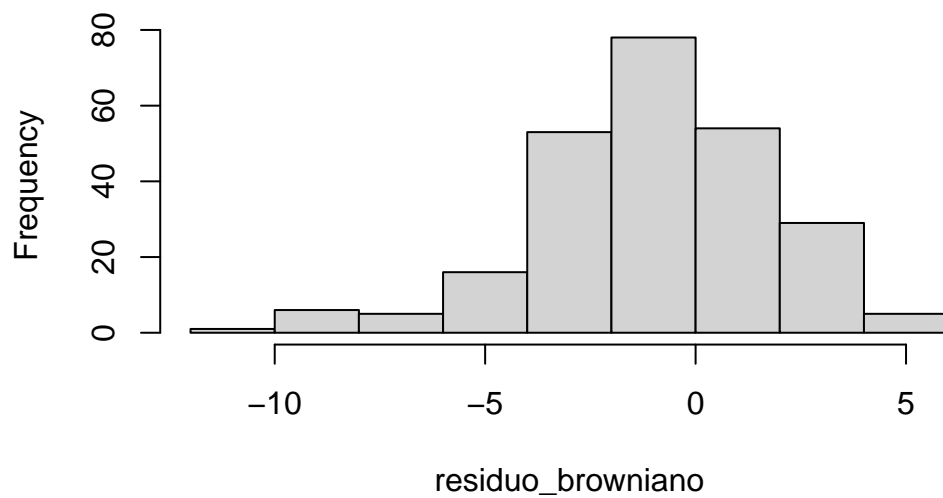
residuo_browniano <- stock_values[-1] - X_prev_browniano
shapiro.test(residuo_browniano)
```

Shapiro-Wilk normality test

data: residuo_browniano
W = 0.98166, p-value = 0.0028

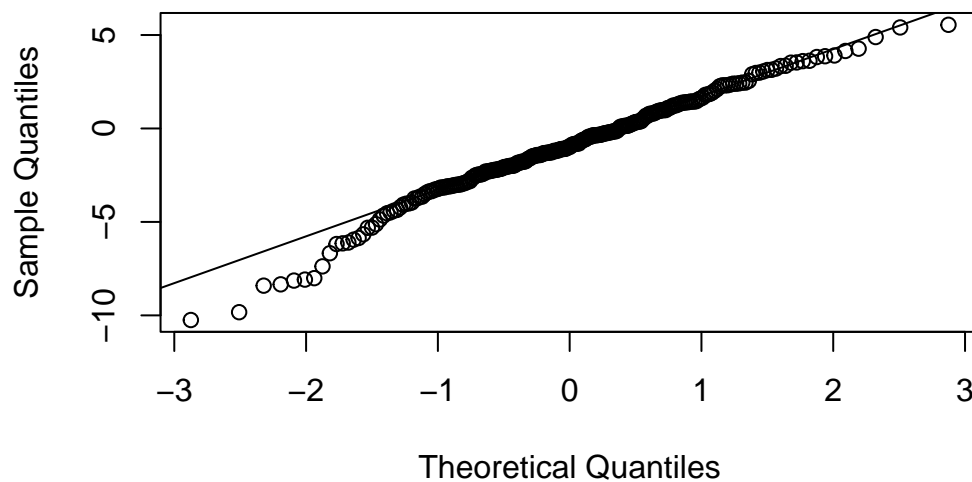
```
hist(residuo_browniano)
```

Histogram of residuo_browniano



```
qqnorm(residuo_browniano)
qqline(residuo_browniano)
```

Normal Q-Q Plot

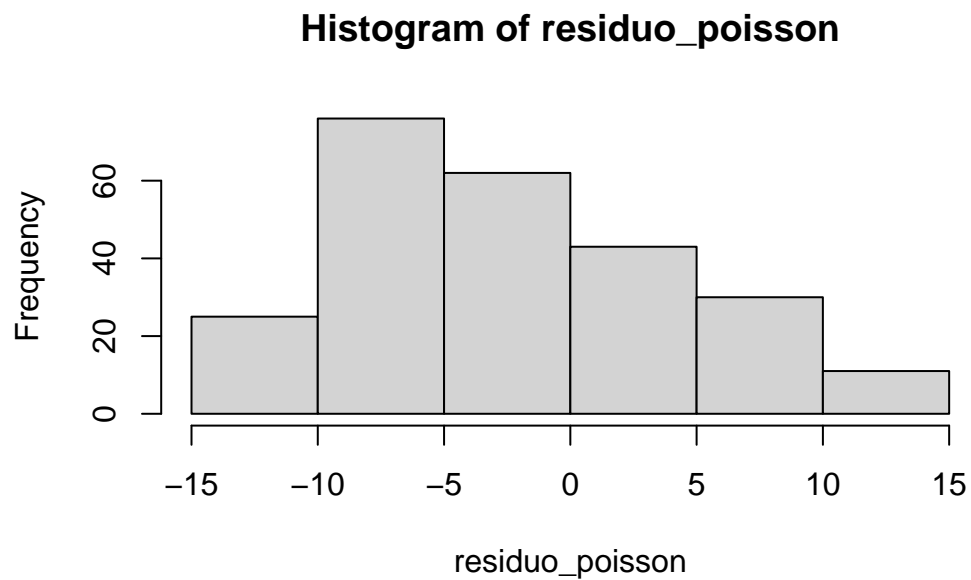


```
residuo_poisson <- stock_values[-1] - X_prev_poisson
shapiro.test(residuo_poisson)
```

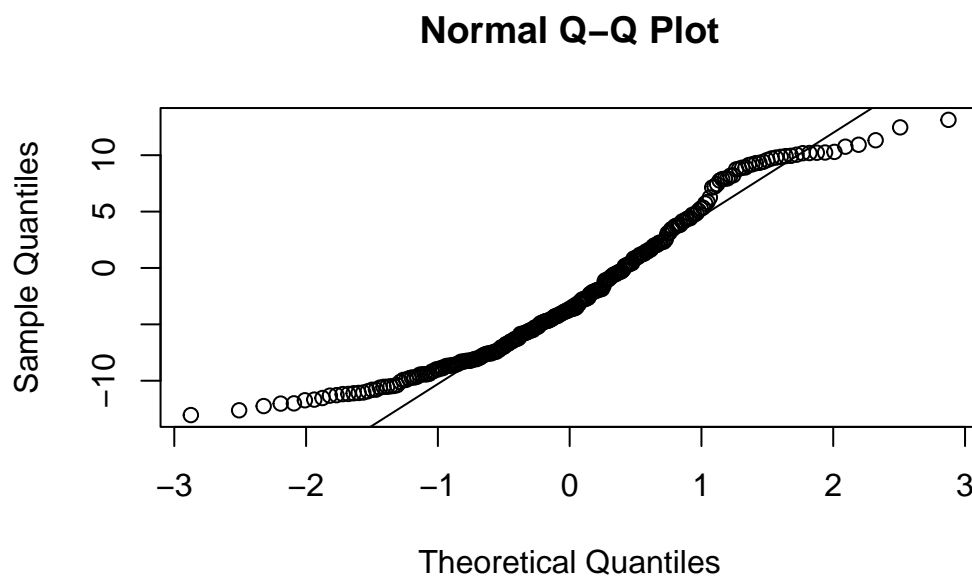
Shapiro-Wilk normality test

```
data: residuo_poisson
W = 0.94498, p-value = 5.071e-08
```

```
hist(residuo_poisson)
```

```
qqnorm(residuo_poisson)
qqline(residuo_poisson)
```



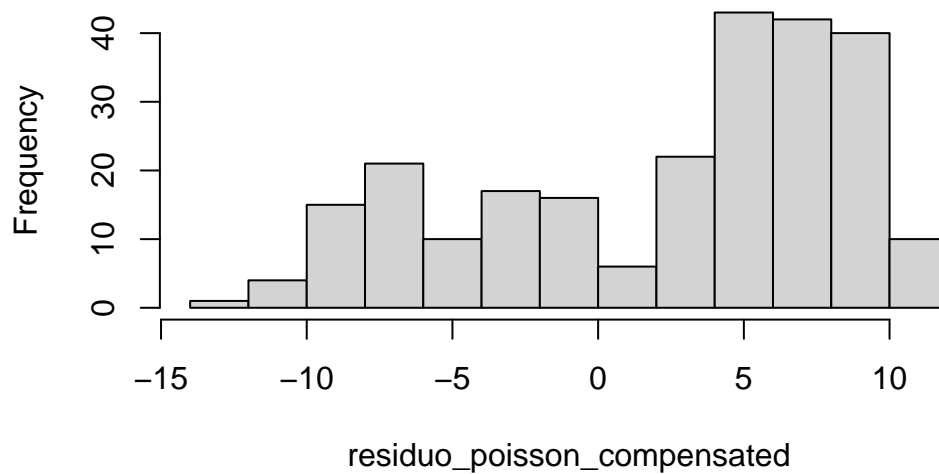
```
residuo_poisson_compensated <- stock_values[-1] - X_prev_poisson_compensated
shapiro.test(residuo_poisson_compensated)
```

Shapiro-Wilk normality test

data: residuo_poisson_compensated
W = 0.9067, p-value = 2.829e-11

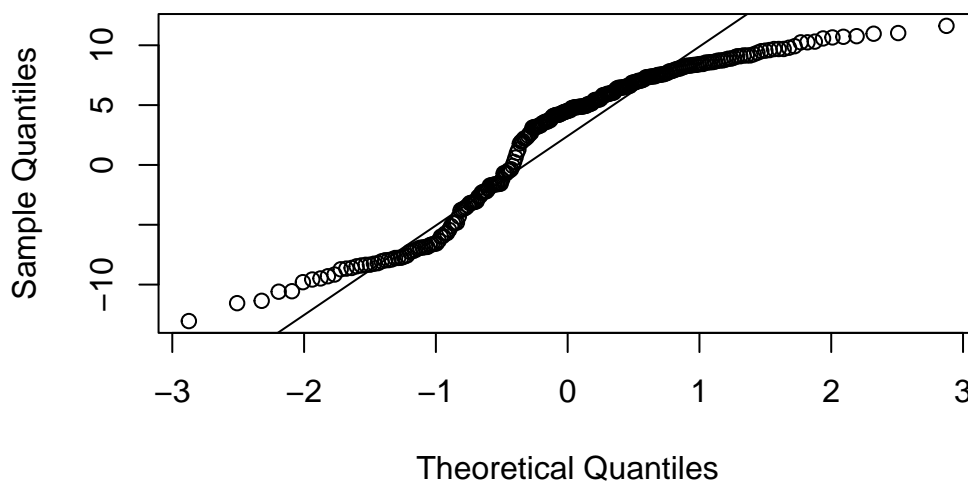
```
hist(residuo_poisson_compensated)
```

Histogram of residuo_poisson_compensated



```
qqnorm(residuo_poisson_compensated)
qqline(residuo_poisson_compensated)
```

Normal Q-Q Plot

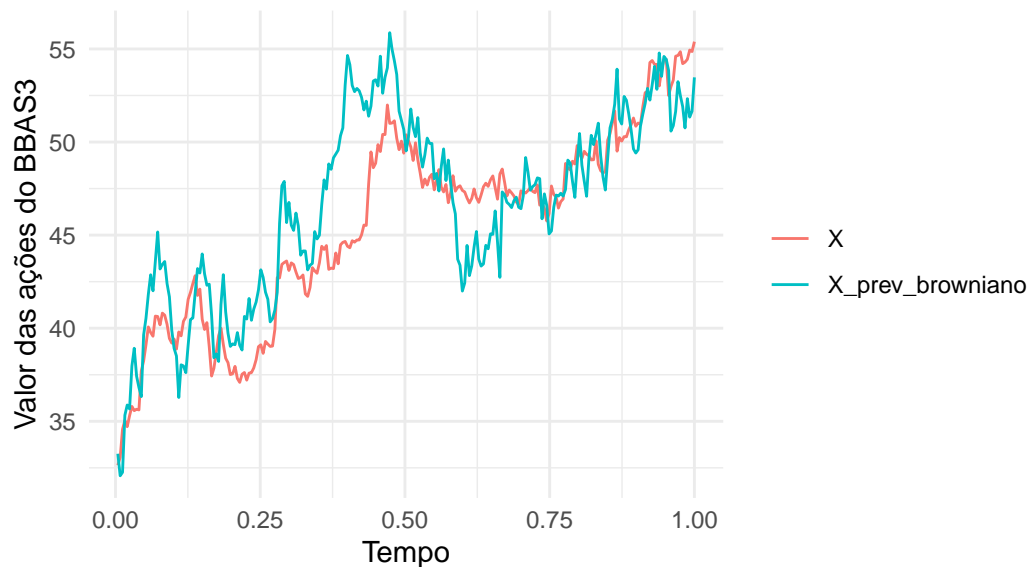


```
dados_sim <- data.frame(
  t = t[-1],
  X = stock_values[-1 ],
  X_prev_browniano = X_prev_browniano,
  X_prev_poisson = X_prev_poisson,
  X_prev_poisson_compensated = X_prev_poisson_compensated
) |>
pivot_longer(cols = c(X, X_prev_browniano, X_prev_poisson, X_prev_poisson_compensated),
  names_to = "Variavel",
  values_to = "Valor")
```

```
dados_sim |>
  filter(Variavel %in% c("X", "X_prev_browniano")) |>
  ggplot() +
```

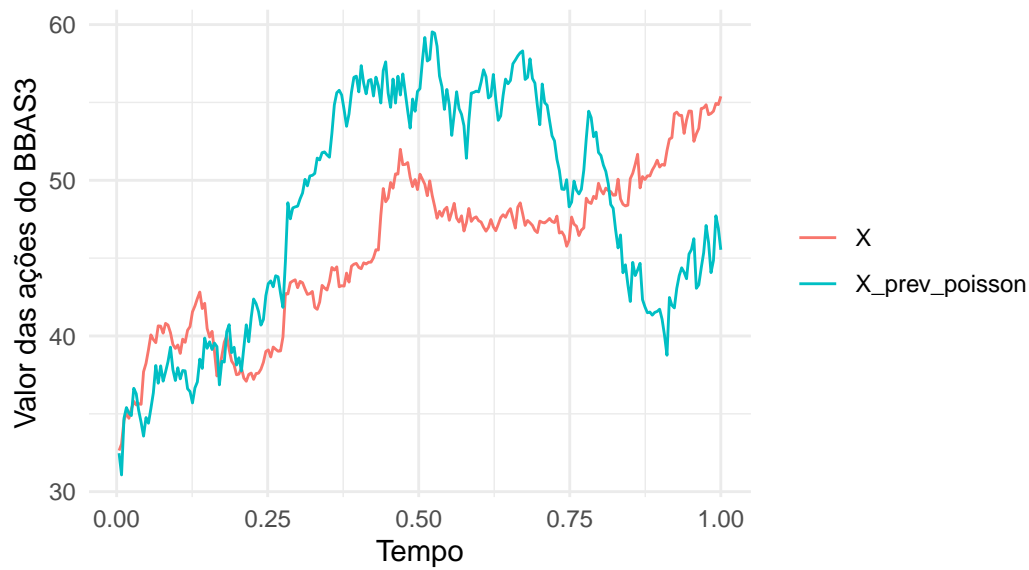
```
geom_line(aes(x = t, y = Valor, color = Variavel)) +
labs(x = "Tempo",
     y = "Valor das ações do BBAS3",
     color = NULL,
     title = "Ajuste do modelo de movimento Browniano
             aos dados de ações do BBAS3 em 2023") +
theme_minimal()
```

Ajuste do modelo de movimento Browniano aos dados de ações do BBAS3 em 2023



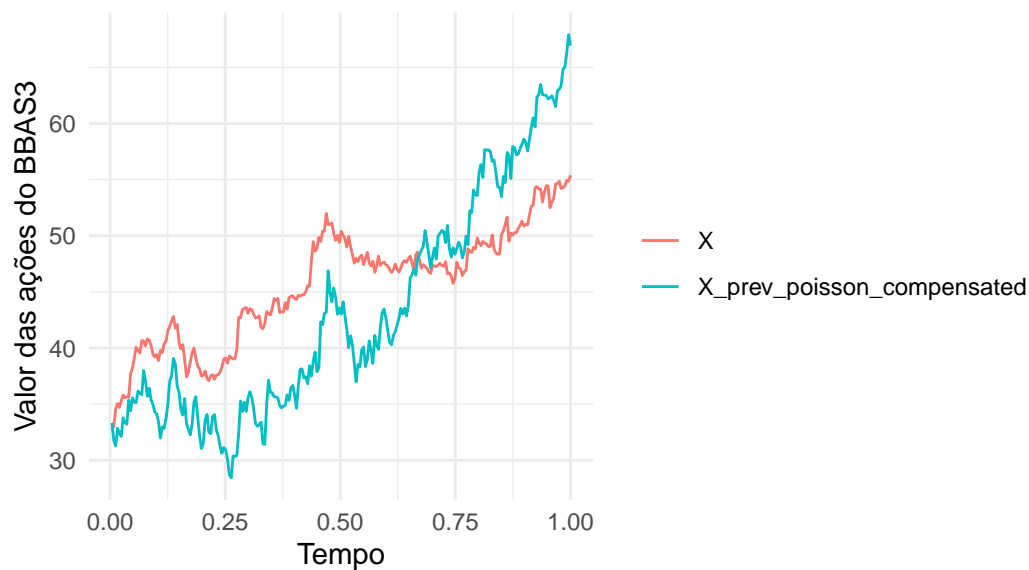
```
dados_sim |>
  filter(Variavel %in% c("X", "X_prev_poisson")) |>
  ggplot() +
  geom_line(aes(x = t, y = Valor, color = Variavel)) +
  labs(x = "Tempo",
       y = "Valor das ações do BBAS3",
       color = NULL,
       title = "Ajuste do modelo de processos de Poisson
               aos dados de ações do BBAS3 em 2023") +
  theme_minimal()
```

Ajuste do modelo de processos de Poisson aos dados de ações do BBAS3 em 2023



```
dados_sim |>
  filter(Variavel %in% c("X", "X_prev_poisson_compensated")) |>
  ggplot() +
  geom_line(aes(x = t, y = Valor, color = Variavel)) +
  labs(x = "Tempo",
       y = "Valor das ações do BBAS3",
       color = NULL,
       title = "Ajuste do modelo de processos de Poisson Compensado
               aos dados de ações do BBAS3 em 2023") +
  theme_minimal()
```

Ajuste do modelo de processos de Poisson Compensado aos dados de ações do BBAS3 em 2023



Previsão de valores de 2024

```
# Define the stock symbol and specify the start and end dates
start_date_valid <- "2024-01-01"
end_date_valid <- "2024-01-15"

# Use getSymbols to fetch historical stock data
getSymbols(stock_symbol,
  src = "yahoo",
  from = start_date_valid,
  to = end_date_valid
)
```

```
[1] "BBAS3.SA"
```

```
# Check the loaded data and get the closing values
stock_values_valid <- as.vector(Cl(get(stock_symbol)))

# Summary statistics
summary(stock_values)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
32.64	40.78	47.00	45.49	49.07	55.39

```
n_valid <- length(stock_values_valid) - 1
t_valid <- seq(0, 1, length.out = n_valid + 1)
N_valid_browniano <- c(0, cumsum(rnorm(n_valid, mean = 0, sd = 1)))
N_valid_poisson <- c(0, cumsum(rpois(n_valid, 1)))
N_valid_poisson_compenstated <- c(0, cumsum(rpois(n_valid, 1)) - seq(1, n_valid, by = 1))
```

```
X_prev_valid_browniano <- simulate_Xtk(t_valid,
  stock_values_valid,
  estim_theta_browniano,
  N_valid_browniano)
X_prev_valid_poisson <- simulate_Xtk(t_valid,
  stock_values_valid,
  estim_theta_poisson,
  N_valid_poisson)

X_prev_valid_poisson_compensated <- simulate_Xtk(t_valid,
  stock_values_valid,
  estim_theta_poisson_compensated,
  N_valid_poisson_compenstated)

dados_valid <- data.frame(
  t = t_valid[-1],
  X = stock_values_valid[-1],
  X_prev_valid_browniano = X_prev_valid_browniano,
  X_prev_valid_poisson = X_prev_valid_poisson,
```

```
X_prev_valid_poisson_compensated = X_prev_valid_poisson_compensated
) |>
pivot_longer(cols = c(X,
                      X_prev_valid_browniano,
                      X_prev_valid_poisson,
                      X_prev_valid_poisson_compensated),
             names_to = "Variavel",
             values_to = "Valor")
```

```
dados_valid |>
  filter(Variavel %in% c("X", "X_prev_valid_browniano")) |>
  ggplot() +
  geom_line(aes(x = t, y = Valor, color = Variavel)) +
  labs(x = "Tempo",
       y = "Valor das ações do BBAS3",
       color = NULL,
       title = "Ajuste do modelo do Movimento Browniano aos dados
               de ações do BBAS3 nas primeiras semanas de 2024") +
  theme_minimal()
```

Ajuste do modelo do Movimento Browniano aos dados
de ações do BBAS3 nas primeiras semanas de 2024

