PROVA 2 DE PROCESSO ESTOCÁSTICOS

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

Professor Felipe Quintino

UnB

Questão 1

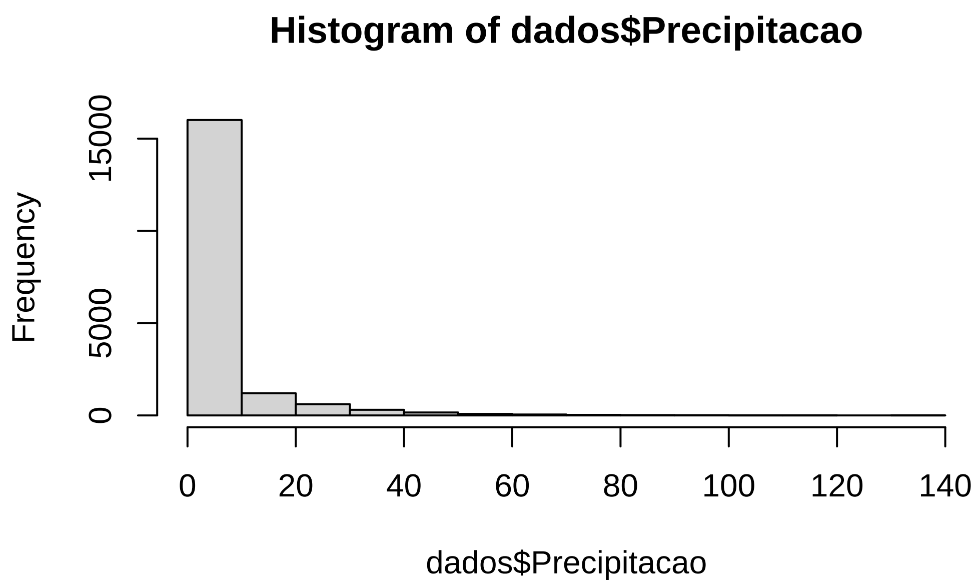
Aplicação ao modelo empírico

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 
```

```
hist(dados$Precipitacao)
```



a)

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 5 categorias: sem chuva (precipitação até $0,001$), garoa (precipitação até 1), chuva fraca (precipitação até $4,8$), chuva moderada (precipitação até $14,2$) e chuva forte (precipitação acima de $14,2$).

```r
# 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
    ))
```

b)

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

```r
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.

```r
# Creating the lagged variable
transicoes_chuva <-
    dados_treinamento |>
    # Lag variable
    mutate(rain_status_lag = lag(rain_status)) |>
```

```r
    # 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:

$$P = \begin{pmatrix} 0.822 & 0.041 & 0.051 & 0.045 & 0.04 \\ 0.334 & 0.12 & 0.186 & 0.177 & 0.183 \\ 0.306 & 0.133 & 0.187 & 0.184 & 0.19 \\ 0.306 & 0.121 & 0.171 & 0.193 & 0.21 \\ 0.282 & 0.118 & 0.174 & 0.204 & 0.223 \end{pmatrix} \tag{0.1}$$

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

```r
# 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()

matriz_transicao
```

```
               sem chuva garoa chuva fraca chuva moderada chuva forte
sem chuva          0.822 0.041       0.051          0.045       0.040
garoa              0.334 0.120       0.186          0.177       0.183
chuva fraca        0.306 0.133       0.187          0.184       0.190
chuva moderada     0.306 0.121       0.171          0.193       0.210
chuva forte        0.282 0.118       0.174          0.204       0.223
```

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

ultimo_estado
```

```
[1] garoa
Levels: sem chuva garoa chuva fraca chuva moderada chuva forte
```

c)

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

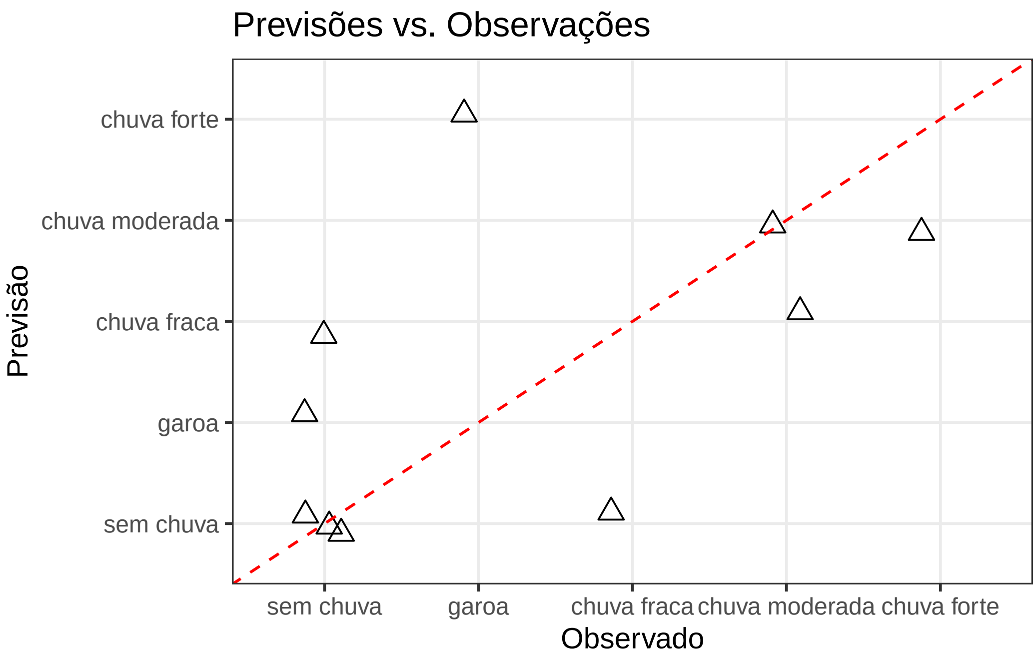
```r
simula_cadeia_markov <- function(n = 10,
                                 valor_inicial,
                                 matriz_transicao,
                                 estados) {
    P <- matriz_transicao
    y <- valor_inicial

    # 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])
}
# Excecution 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.

```r
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
```

```r
# 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.

```r
# 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"
```

```r
# 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 
```

a)

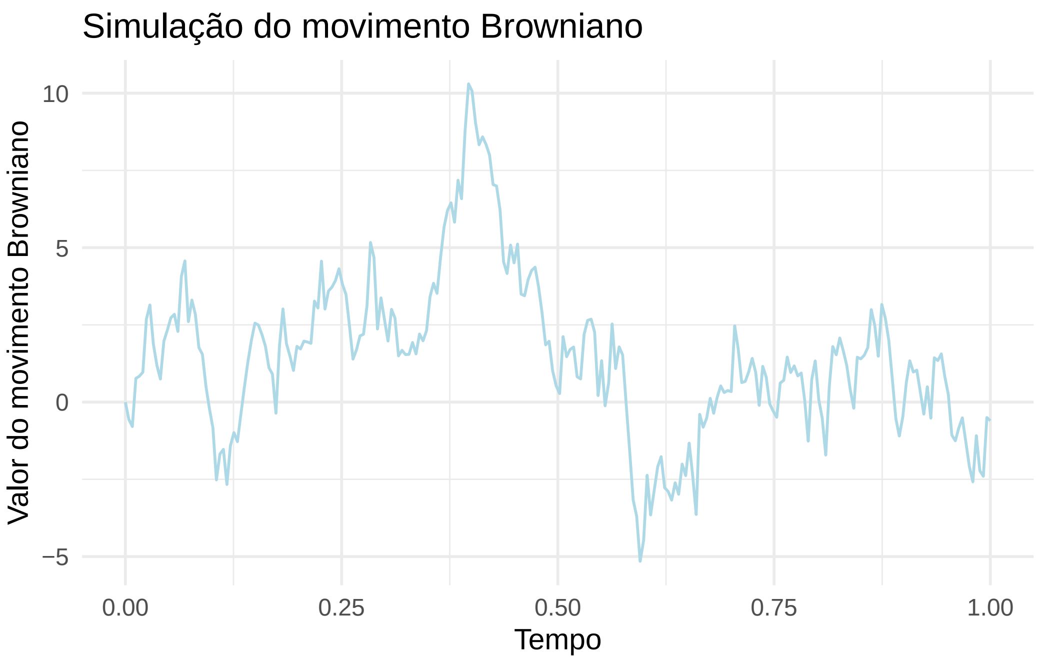
Obtendo o número de observações no vetor de valores da ação, é possível gerar simulações do movimento Browniano e do processo de Poisson (padrão e compensado) com a mesma quantidade de pontos que a base de dados. Considerando um intervalo de 0 a 1, em anos, é gerado um vetor t relativo ao tempo decorrido 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. Para o processo de Poisson é feita a soma de valores da distribuição Poisson com parâmetro $\lambda = 1$. Por fim, para o processo de Poisson compensado, é feita a soma de valores da distribuição Poisson com parâmetro $\lambda = 1$ subtraídos de $\lambda t_k = t_k$, onde t_k é o tempo decorrido do início da contagem ao momento de cada observação.

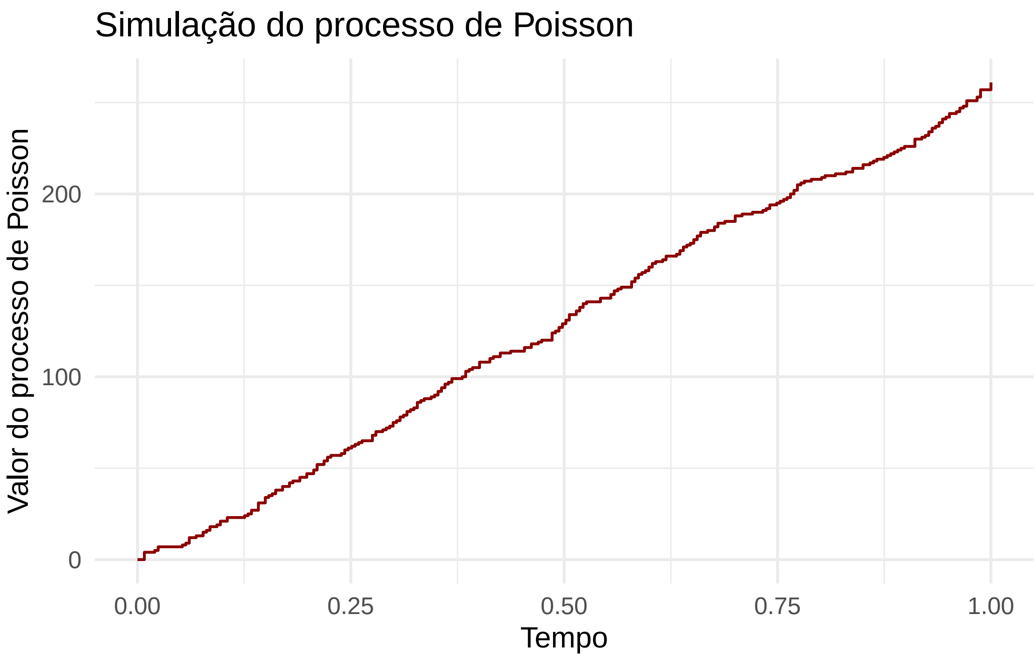
```r
n <- length(stock_values) - 1
t <- seq(0, 1, length.out = n + 1)
B <- c(0, cumsum(rnorm(n, mean = 0, sd = 1)))
N <- c(0, cumsum(rpois(n, lambda = 1)))
```

```
N_compensated <- c(0, cumsum(rpois(n, lambda = 1)) - seq(1, n, by = 1))
```

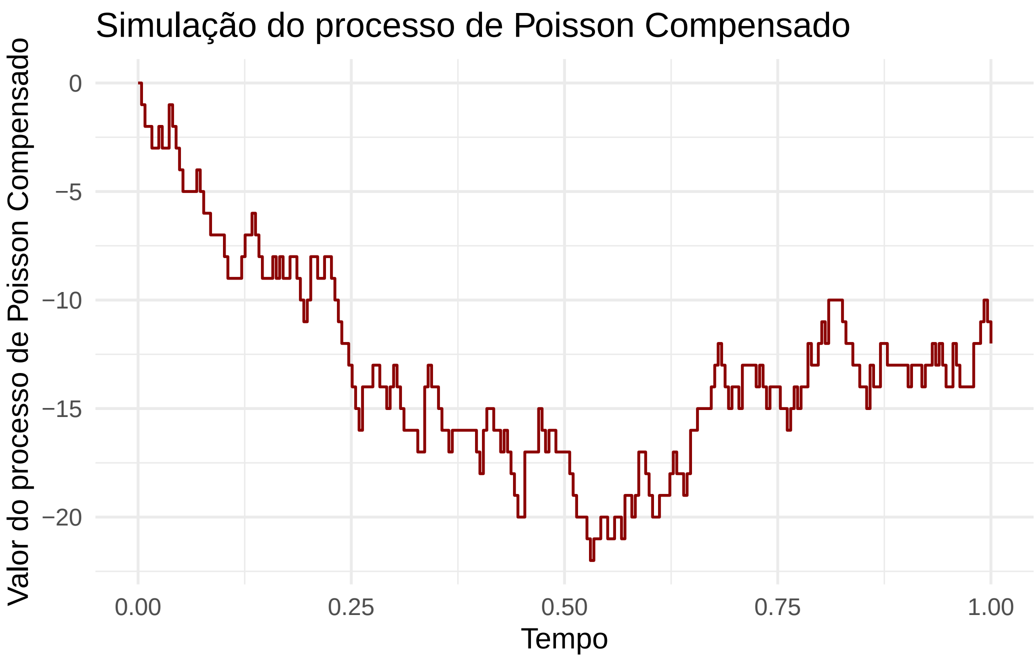
```
data.frame(time = t, Browniano = B) |>
    ggplot(aes(x = time, y = Browniano)) +
    geom_line(color = "lightblue") +
    labs(
        x = "Tempo",
        y = "Valor do movimento Browniano",
        title = "Simulação do movimento Browniano"
    ) +
    theme_minimal()
```




```
data.frame(time = t, Poisson = N) |>
    ggplot(aes(x = time, y = Poisson)) +
    geom_step(color = "darkred") +
    labs(
        x = "Tempo",
        y = "Valor do processo de Poisson",
        title = "Simulação do processo de Poisson"
    ) +
    theme_minimal()
```




```r
data.frame(time = t, Compensated = N_compensated) |>
    ggplot(aes(x = time, y = Compensated)) +
    geom_step(color = "darkred") +
    labs(
        x = "Tempo",
        y = "Valor do processo de Poisson Compensado",
        title = "Simulação do processo de Poisson Compensado"
    ) +
    theme_minimal()
```



b)

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)
}
```

c)

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 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] 5.9375
```

```
(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
```

d)

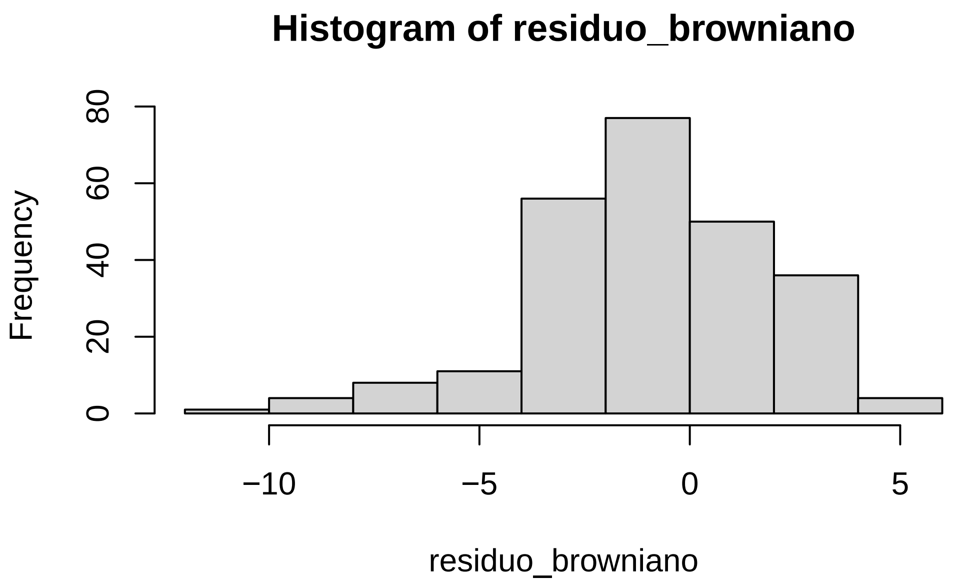
Os testes dos resíduos são feitos a seguir. O que se percebe é que os resíduos gerados pelos três modelos não seguem uma distribuição normal, o que pode ser observado pelos valores-p do teste de Shapiro-Wilk e pelos gráficos de histograma e de quantis.

Uma das possíveis razões pode ser a presença de autocorrelação nos resíduos, o que é confirmado pelo gráfico de autocorrelação.

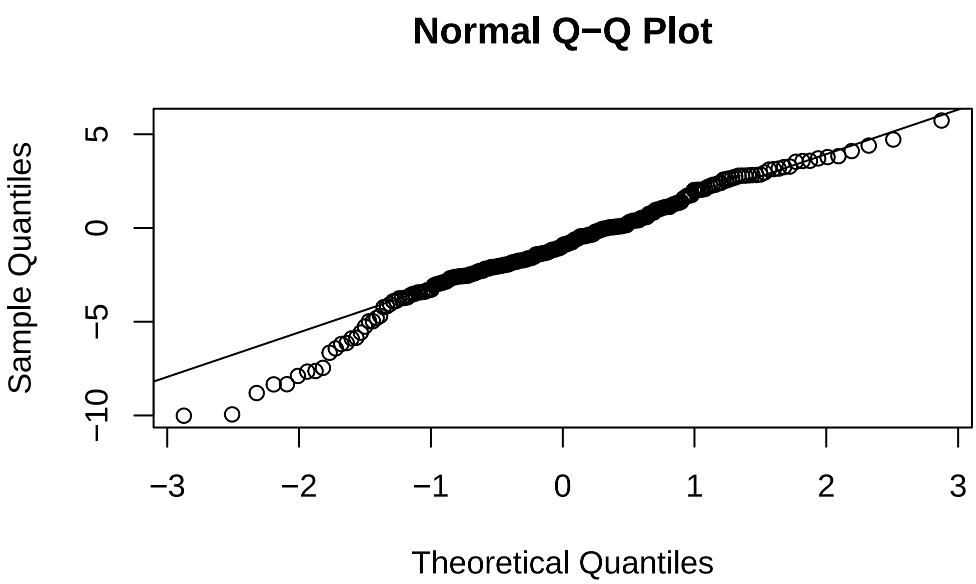
```
X_prev_browniano <-
    simulate_Xtk(t, stock_values, estim_theta_browniano, B)
X_prev_poisson <-
    simulate_Xtk(t, stock_values, estim_theta_poisson, N)
X_prev_poisson_compensated <-
    simulate_Xtk(t, stock_values, estim_theta_poisson_compensated, N_compensated)

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

```
    Shapiro-Wilk normality test

data:  residuo_browniano
W = 0.97457, p-value = 0.000208
```

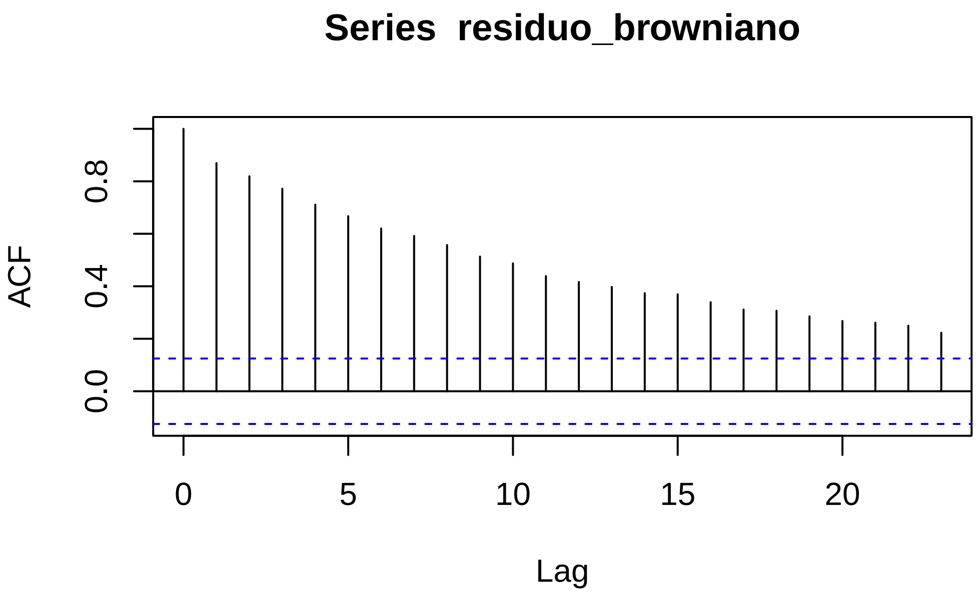
```r
hist(residuo_browniano)
```




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



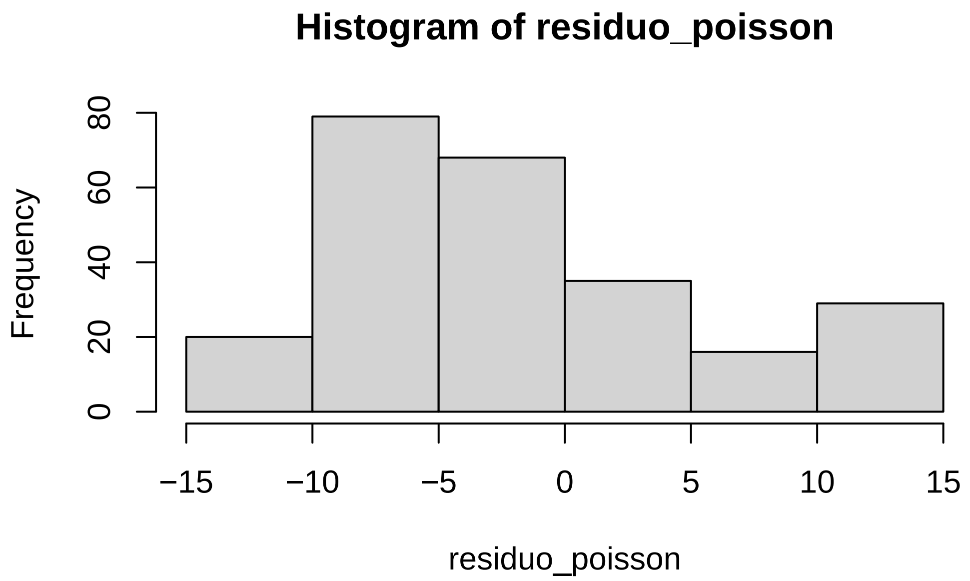

```r
acf(residuo_browniano)
```



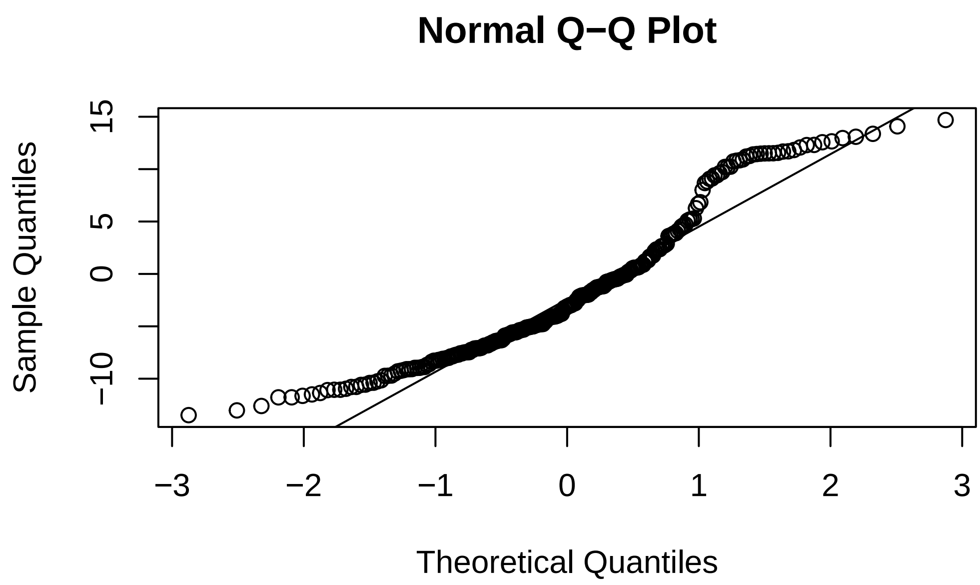

```
residuo_poisson <- stock_values[-1] - X_prev_poisson
shapiro.test(residuo_poisson)
```

```
	Shapiro-Wilk normality test

data:  residuo_poisson
W = 0.93173, p-value = 2.835e-09
```

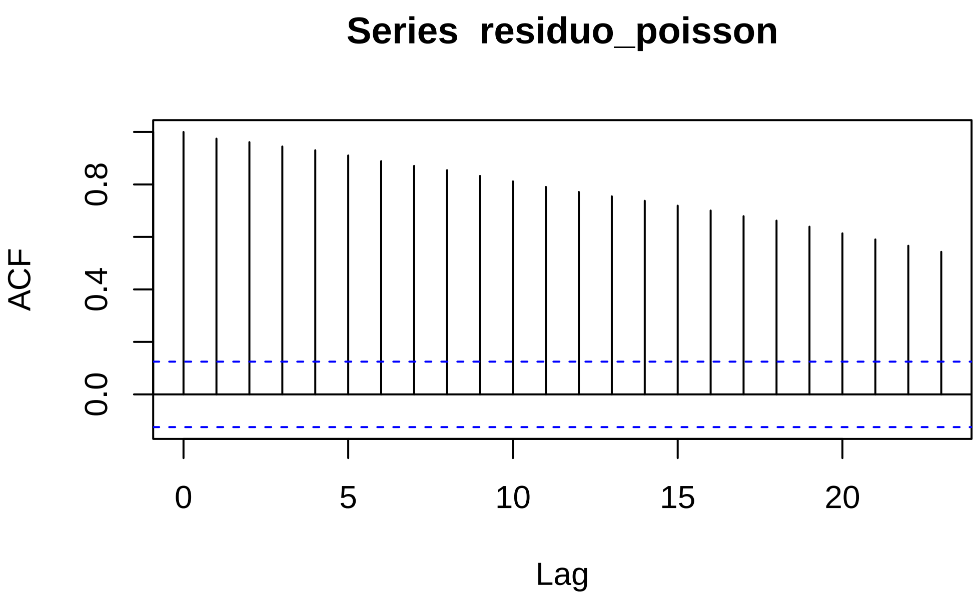
```
hist(residuo_poisson)
```




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



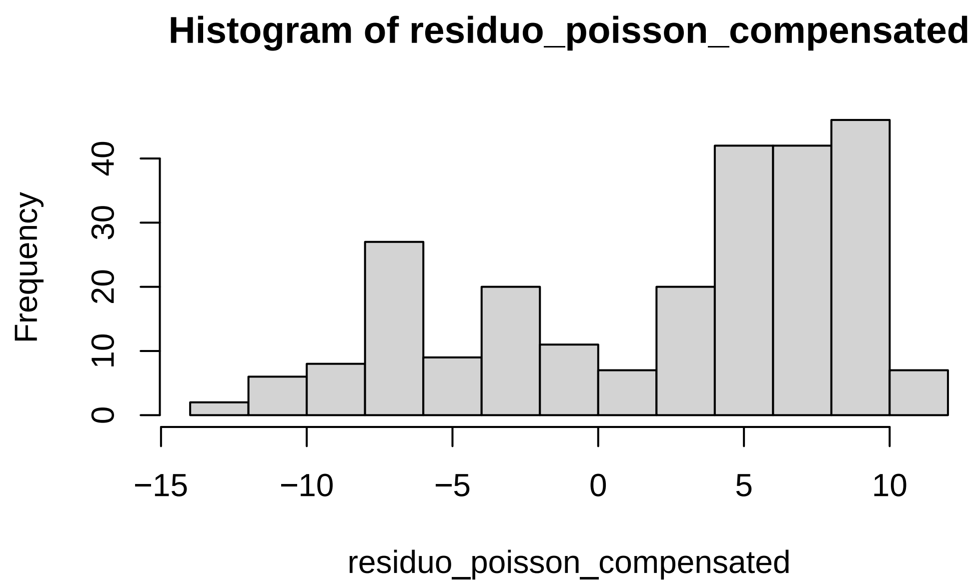

```r
acf(residuo_poisson)
```



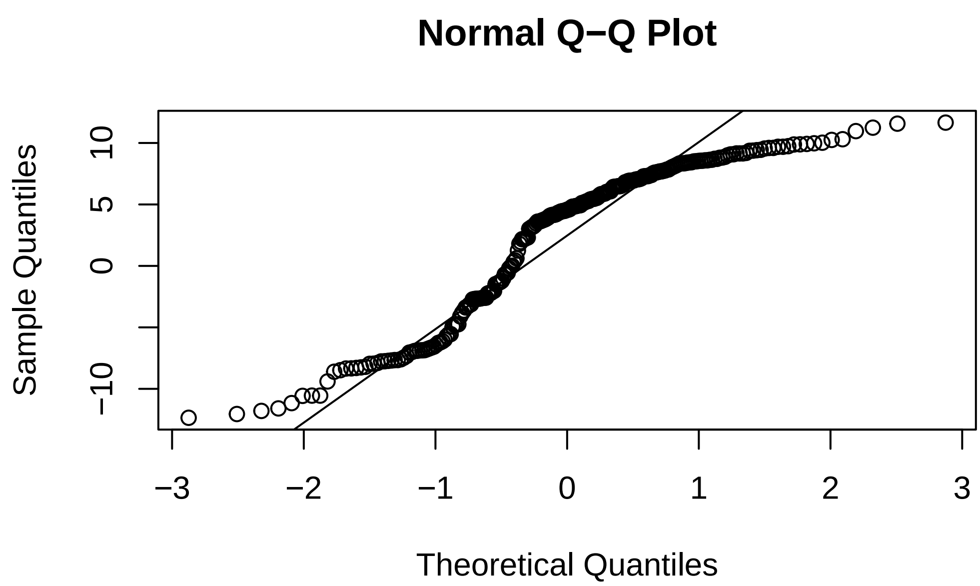

```r
residuo_poisson_compensated <- stock_values[-1] - X_prev_poisson_compensated
shapiro.test(residuo_poisson_compensated)
```

```
	Shapiro-Wilk normality test

data:  residuo_poisson_compensated
W = 0.9024, p-value = 1.395e-11
```

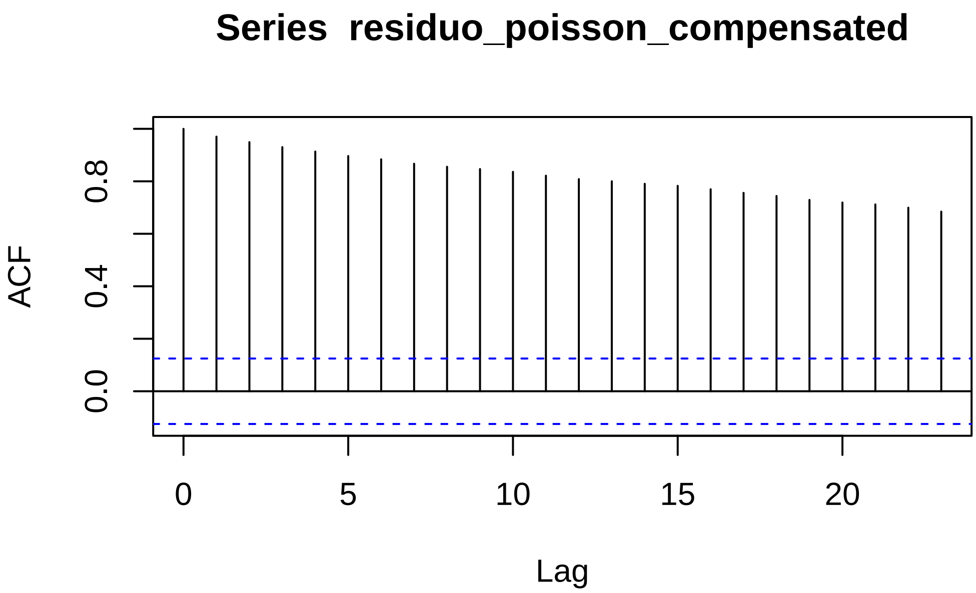
```r
hist(residuo_poisson_compensated)
```




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




```r
acf(residuo_poisson_compensated)
```



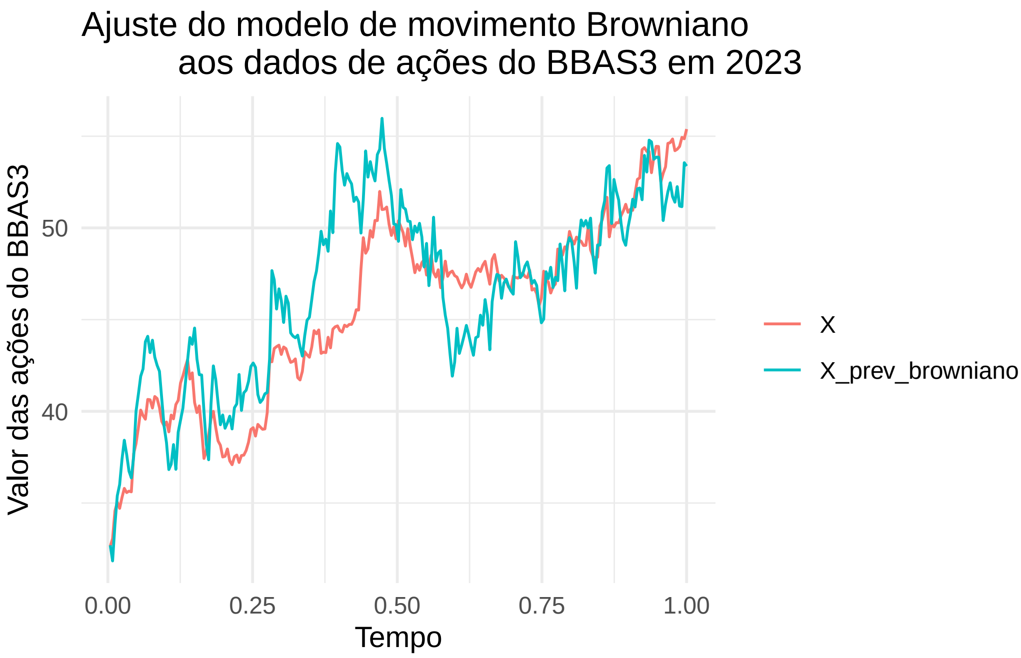
e)

Os modelos ajustados estão representados nos gráficos a seguir. Em todos eles, a série original está representada em vermelho. Visualmente, o modelo que melhor se ajusta aos dados é o modelo de movimento Browniano. Para testar isso, são realizados testes no item f.

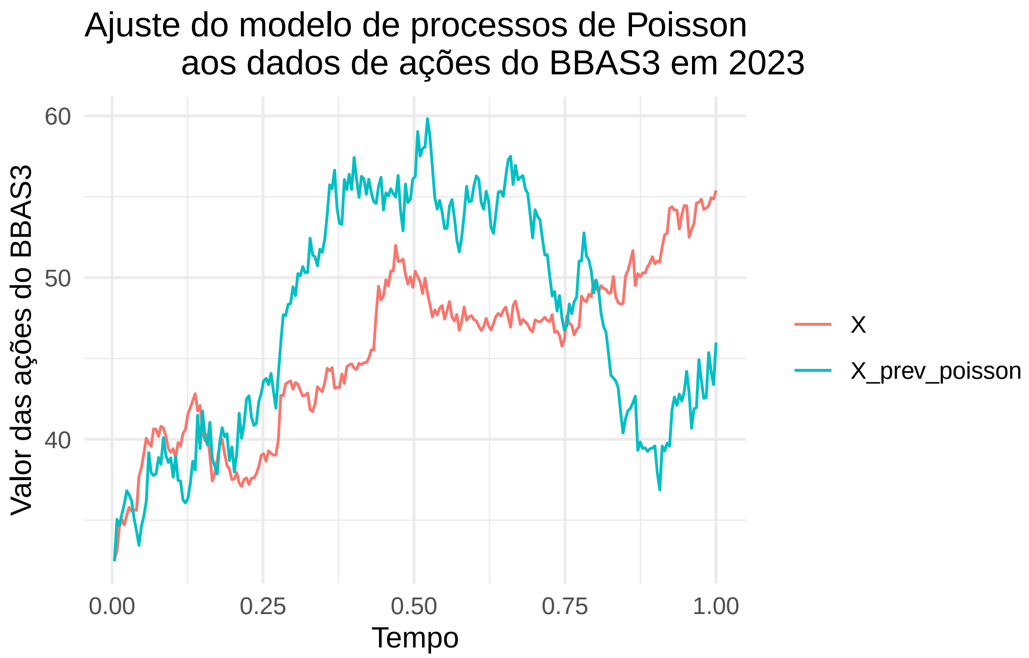
```r
dados_mod <- data.frame(
    t = t[-1],
    X = stock_values[-1],
    X_prev_browniano = X_prev_browniano,
    X_prev_poisson = X_prev_poisson,
    X_prev_poisson_compensated = X_prev_poisson_compensated
)
dados_sim <- dados_mod |>
    pivot_longer(
        cols = c(X,
                 X_prev_browniano,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

```r
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"
```

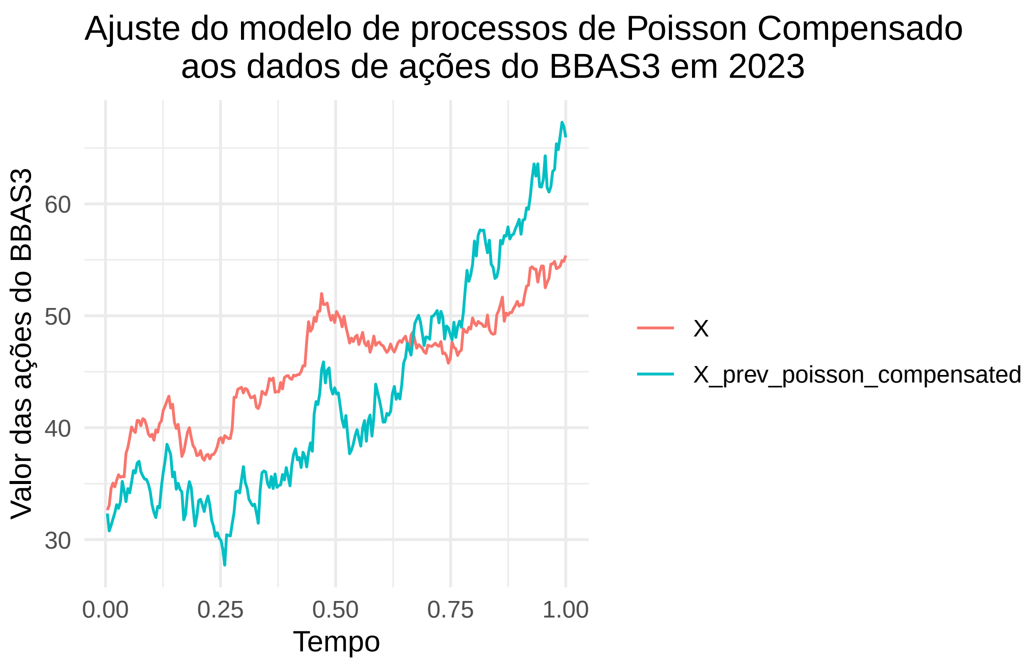
```
    ) +
    theme_minimal()
```




```
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()
```




```
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()
```



f) Definindo o melhor modelo

Para definir o melhor modelo, pode-se comparar as métricas mais comumento usadas:

```
mse_browniano <- mean(residuo_browniano**2)
modelo_browniano <- lm(data = dados_mod, X ~ X_prev_browniano)
r2_browniano <- summary(modelo_browniano)$r.squared
aic_browniano <- AIC(modelo_browniano)

mse_poisson <- mean(residuo_poisson**2)
modelo_poisson <- lm(data = dados_mod, X ~ X_prev_poisson)
r2_poisson <- summary(modelo_poisson)$r.squared
aic_poisson <- AIC(modelo_poisson)

mse_poisson_compensated <- mean(residuo_poisson_compensated**2)
modelo_poisson_compensated <- lm(
    data = dados_mod,
    X ~ X_prev_poisson_compensated
)
r2_poisson_compensated <- summary(modelo_poisson_compensated)$r.squared
aic_poisson_compensated <- AIC(modelo_poisson)
```

```
resultados <- data.frame(
    Modelo = c("Movimento Browniano", "Processo de Poisson", "Processo de Poisson Compensado
    MSE = c(mse_browniano, mse_poisson, mse_poisson_compensated),
    R2 = c(r2_browniano, r2_poisson, r2_poisson_compensated),
    AIC = c(aic_browniano, aic_poisson, aic_poisson_compensated)
)

resultados
```

```
                          Modelo       MSE        R2      AIC
1            Movimento Browniano  8.369757 0.7351166 1190.829
2            Processo de Poisson 51.263530 0.1688992 1473.264
3 Processo de Poisson Compensado 45.577426 0.7289872 1473.264
```

- **MSE**: o modelo com menor erro quadrático médio é o modelo de movimento Browniano.
- **R²**: o modelo com maior coeficiente de determinação é o modelo de movimento Browniano.
- **AIC**: o modelo com menor critério de informação de Akaike é o modelo de movimento Browniano.

g) Previsão de valores de 2024

Primeiro, foram baixados os dados das ações entre os dias 1 e 15 de janeiro de 2024. Em seguida, foram geradas as previsões dos três modelos para esses dias.

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

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

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

```r
# 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
```

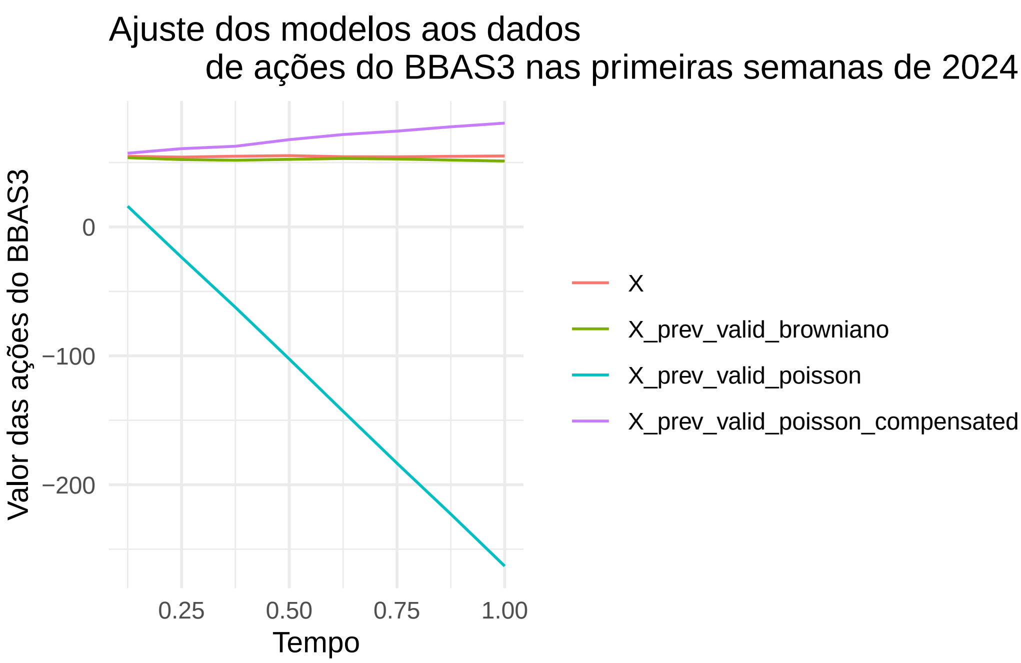
```r
n_valid <- length(stock_values_valid) - 1
t_valid <- seq(0, 1, length.out = n_valid + 1)
N_valid_browniano <- c(0, cumsum(rnorm(n_valid, mean = 0, sd = 1)))
N_valid_poisson <- c(0, cumsum(rpois(n_valid, 1)))
N_valid_poisson_compenstated <- c(0, cumsum(rpois(n_valid, 1)) -
    seq(1, n_valid, by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

```r
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"
    )
```

As previsões geradas pelos três modelos são:

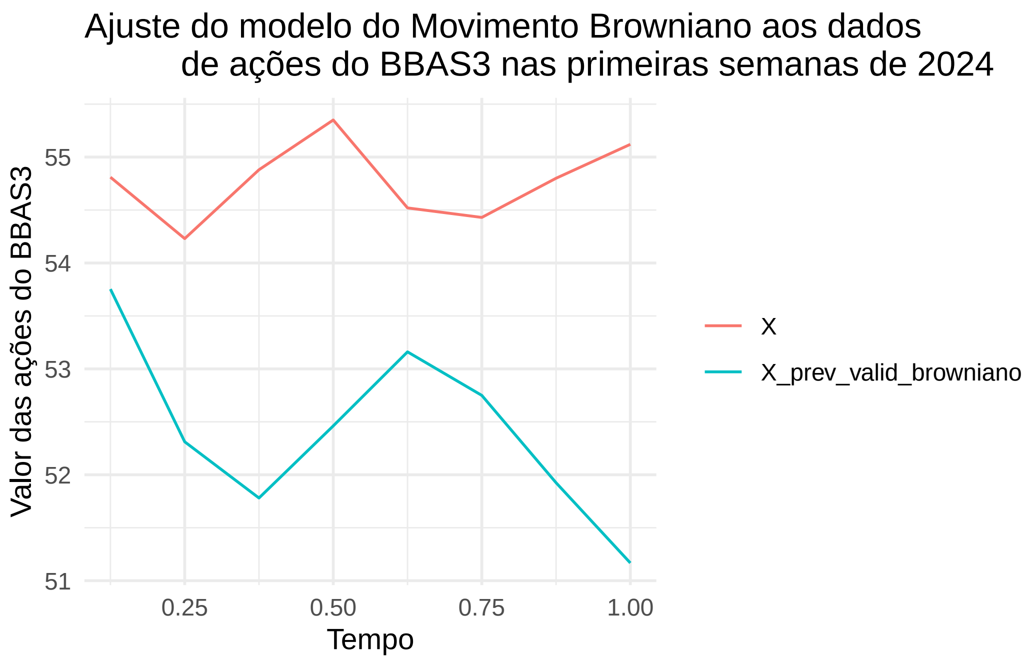
```
dados_valid |>
    ggplot() +
    geom_line(aes(x = t, y = Valor, color = Variavel)) +
    labs(
        x = "Tempo",
        y = "Valor das ações do BBAS3",
        color = NULL,
        title = "Ajuste dos modelos aos dados
          de ações do BBAS3 nas primeiras semanas de 2024"
    ) +
   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



Como percebido pelo item anterior, o modelo que ajustou melhor os dados foi o modelo estimado pelo movimento Browniano, cujo gráfico pode ser visto a seguir:

```
dados_valid |>
    filter(Variavel %in% c("X", "X_prev_valid_browniano")) |>
```

```r
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()
```



Nessa linha, o movimento Browniano parece ter melhores propriedades para estimar e prever valores futuros das ações, marcadamente conhecidos por alta volatilidade e comportamentos não-lineares e não-sistemáticos.