Lista 4 - Series Temporais

Davi Wentrick Feijó - 200016806

2024-01-08

Dados - Consumo de Energia

Formatacao do Banco e Variaveis

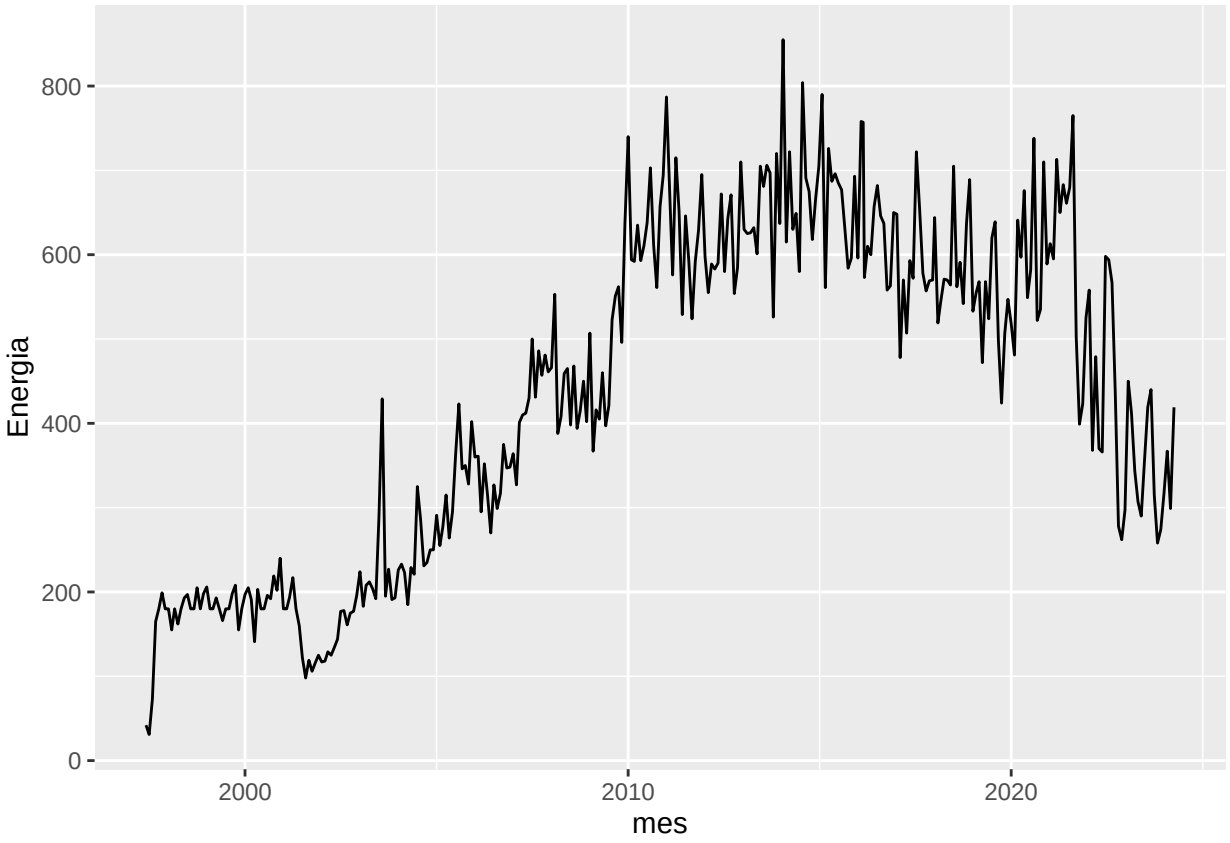
```
dados_energia <- read_excel("ConsumoEnergiaEAgua.xlsx") %>%
  select(-c(2,5:11))

dados_energia = rbind(dados_energia, list("2024-04-01",419,33)) #Adicionando a nova observacao
```

```
dados_energia = dados_energia %>%
  mutate(consumo = Energia/Dias)
```

```
dados_energia
```

```
## # A tibble: 322 x 4
##    mes                 Energia  Dias consumo
##    <dttm>                <dbl> <dbl>   <dbl>
##  1 1997-06-01 00:00:00      42    42    1
##  2 1997-07-01 00:00:00      31    30    1.03
##  3 1997-08-01 00:00:00      73    33    2.21
##  4 1997-09-01 00:00:00     165    29    5.69
##  5 1997-10-01 00:00:00     180    30    6
##  6 1997-11-01 00:00:00     199    33    6.03
##  7 1997-12-01 00:00:00     180    29    6.21
##  8 1998-01-01 00:00:00     180    29    6.21
##  9 1998-02-01 00:00:00     155    33    4.70
## 10 1998-03-01 00:00:00     180    29    6.21
## # i 312 more rows
```

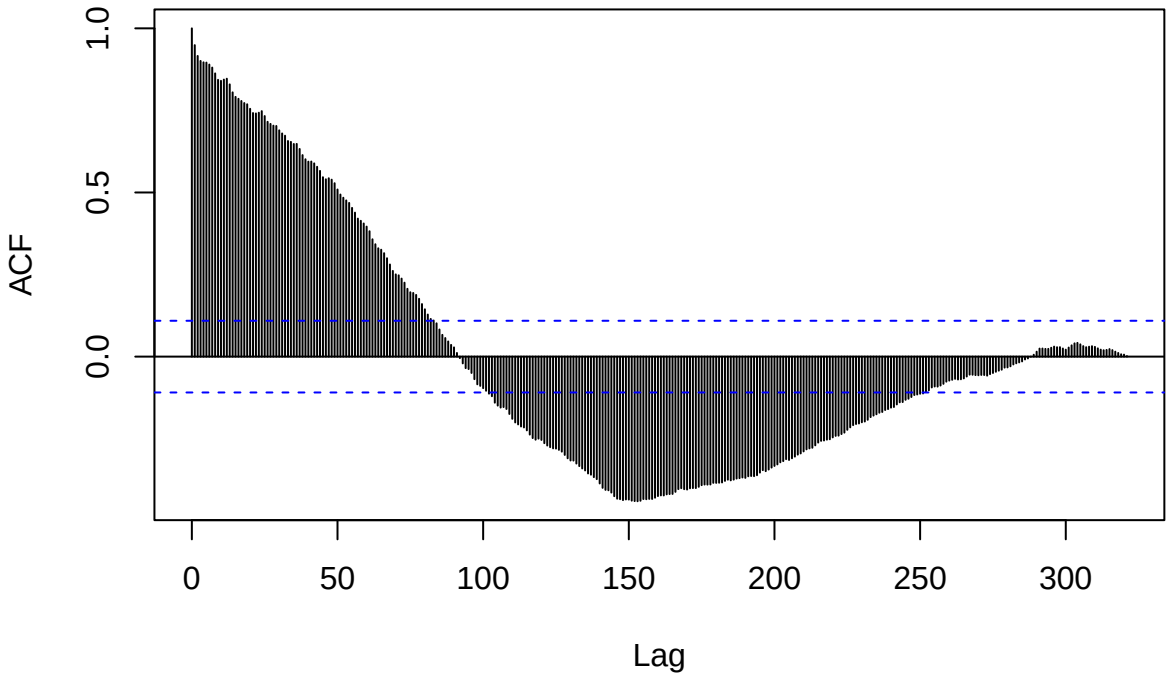
Grafico 1



Evolução temporal do consumo elétrico médio mensal de uma residência (kWh), totalizando 321 observações de jun/1997 a fev/2024.

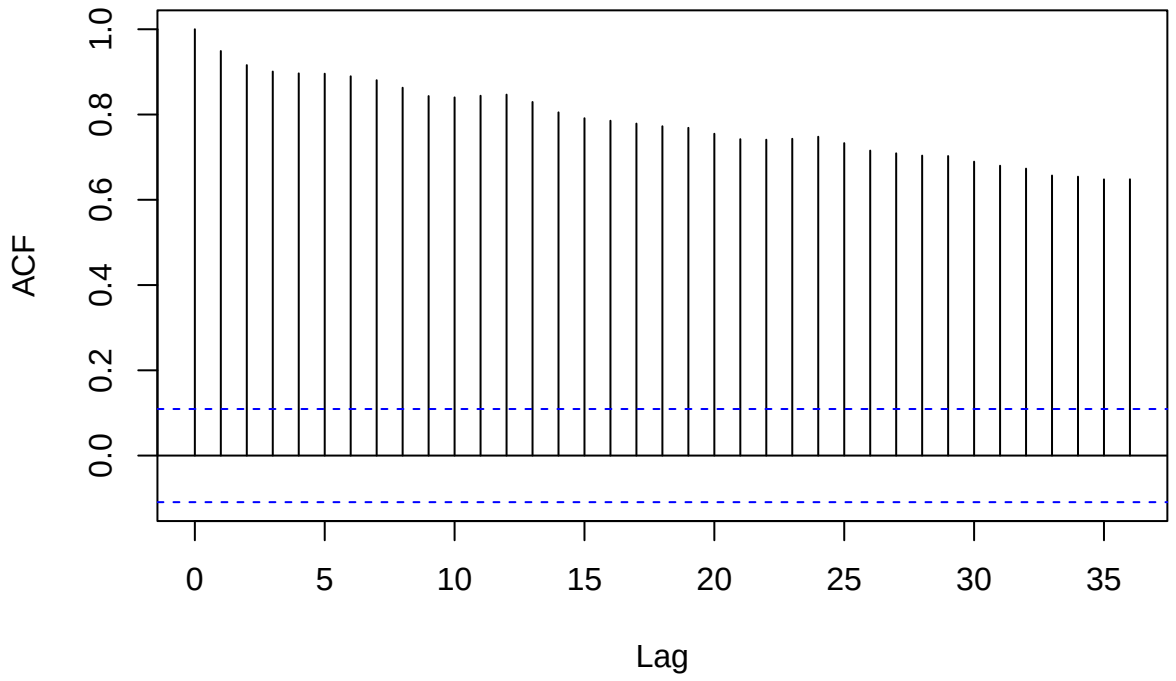
Funcao de Autocorrelacao

```
rho = acf(dados_energia$consumo, lag = length(dados_energia$consumo), plot = FALSE)
```



Nos podemos dar um “zoom” no grafico é observar que:

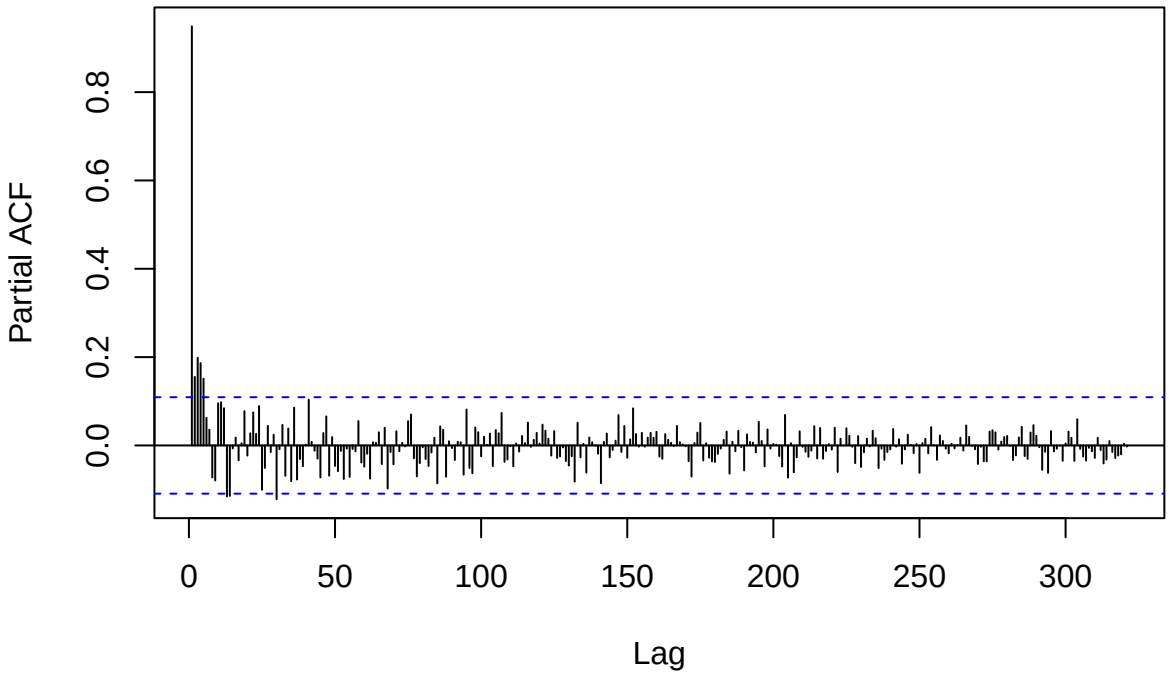
```
rho = acf(dados_energia$consumo, lag = 36, plot = FALSE)
```



Existe um decaimento lento da Funcao de Autocorrelacao com $\rho(1)$ proximo de 1 o que é indicativo de nao estacionariedade

Funcao de Autocorrelacao Parcial (FACP)

```
phi = pacf(dados_energia$consumo, lag = length(dados_energia$consumo), plot = FALSE)
```



O corte abrupto da FACP no lag 1 com ϕ_{11} proximo de 1 é indicativo de nao estacionariedade

Novamente podemos apoximar o grafico apra ficar mais evidente:

```
phi = pacf(dados_energia$consumo, lag = 36, plot = FALSE)
```

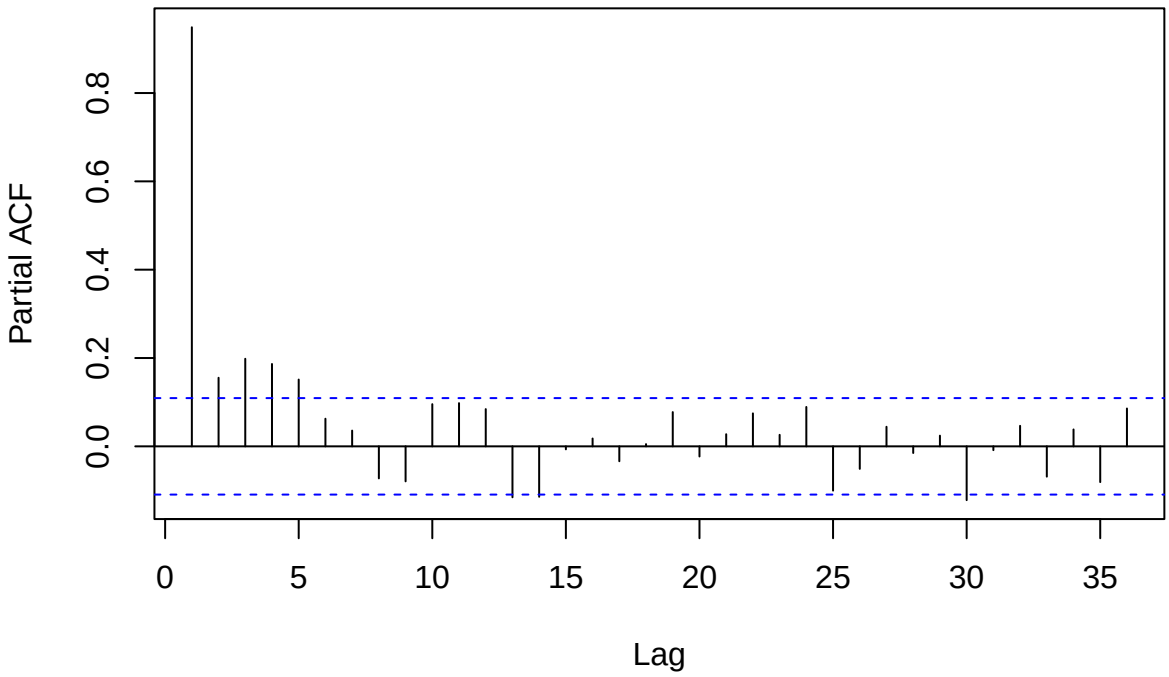


Diagrama de fase entre Y_t e Y_{t+1}

Sabemos que nosso Y_t é nosso consumo (Energia/Dia) e queremos analisar Y_{t+1} para isso dentro do R teremos que deslocar a serie de valores em uma unidade. Dentro do R vamos organizar da seguinte forma:

- Y_t = Consumo = Energia/Dias = `consumo`
- Y_{t+1} = Consumo deslocado em 1 = `consumo_1`

```
n = length(dados_energia$consumo)

dados_energia$consumo_1 = c(NA, (dados_energia$consumo[1:(n-1)]))

head((cbind(dados_energia$consumo,dados_energia$consumo_1)))
```

```
##          [,1]     [,2]
## [1,] 1.000000       NA
## [2,] 1.033333 1.000000
## [3,] 2.212121 1.033333
## [4,] 5.689655 2.212121
## [5,] 6.000000 5.689655
## [6,] 6.030303 6.000000
```

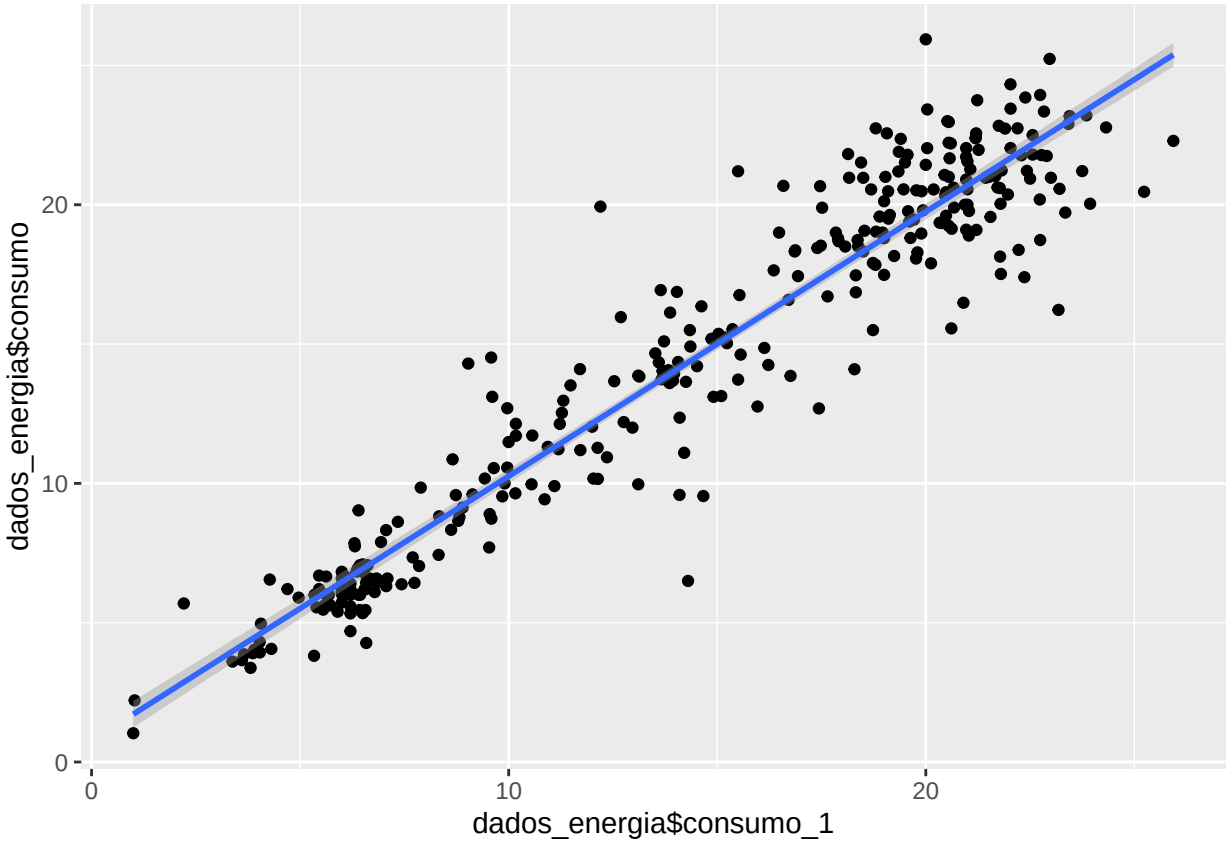
```
head(c(NA,diff(dados_energia$consumo))) #diferenca entre as obs
```

```
## [1]         NA 0.03333333 1.17878788 3.47753396 0.31034483 0.03030303
```

Agora vamos adicionar essas colunas no banco de dados

```
dados_energia = dados_energia %>%
  mutate(consumo_1 = c(NA, (consumo[1:(n-1)])),
         diff = c(NA,diff(consumo)))
```

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


dados_energia$consumo
0
10
20
0
10
20
dados_energia$consumo_1

Variacao de energia (Diferenciacao)

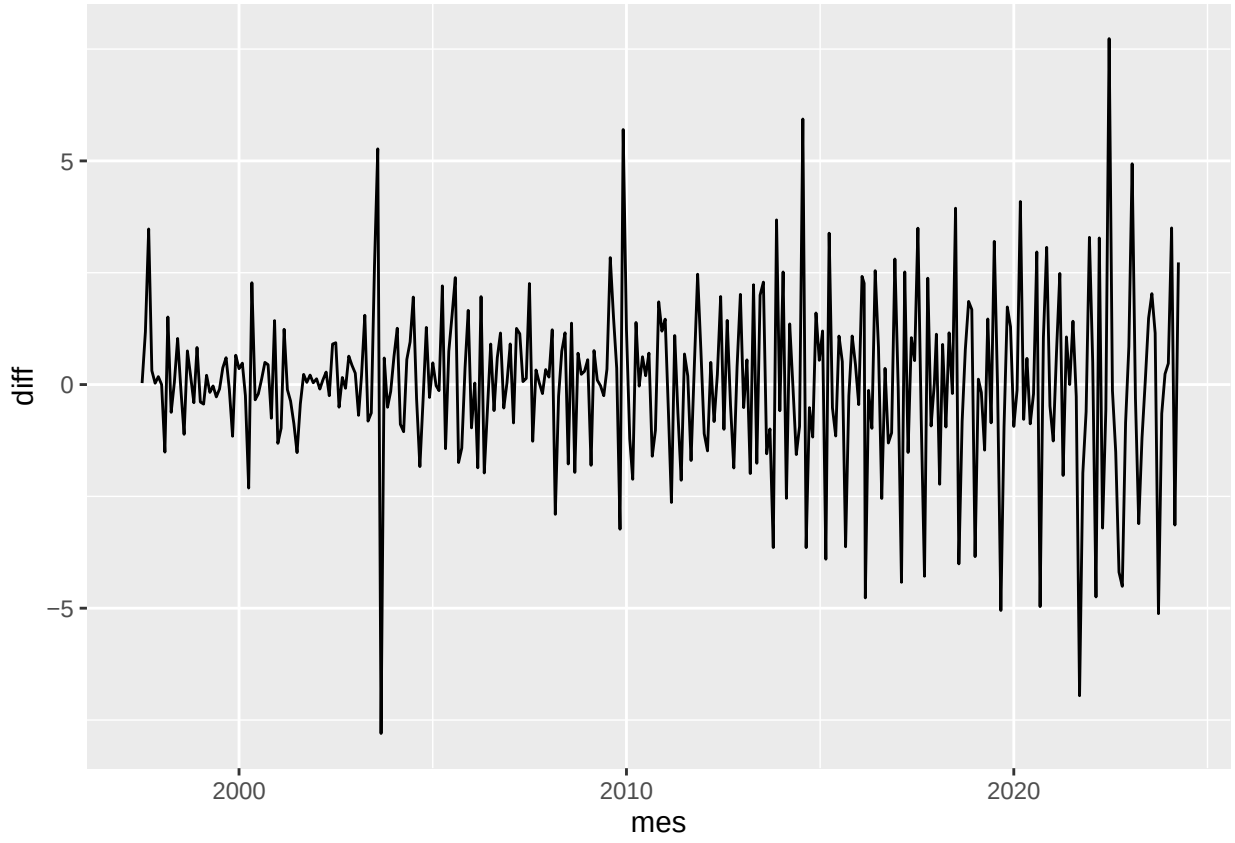
Nessa etapa vamos repetira glumas analises mas no lugar da variavel `consumo` vamo utilizar a variavel `diff` que representa a variacao do consumo de um mes para o outro.

```
head(dados_energia$diff)
```

```
## [1]         NA 0.03333333 1.17878788 3.47753396 0.31034483 0.03030303
```

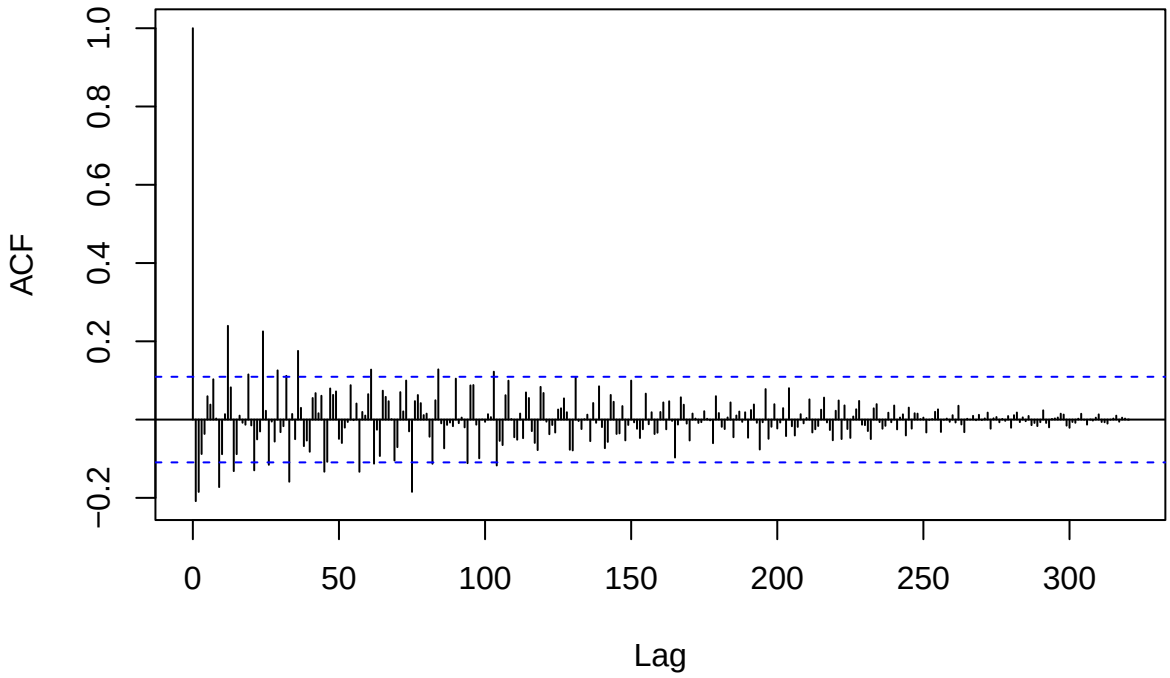
```
ggplot(data = dados_energia, aes(x = mes, y = diff))+
  geom_line()
```

```
## Warning: Removed 1 row containing missing values (`geom_line()`).
```



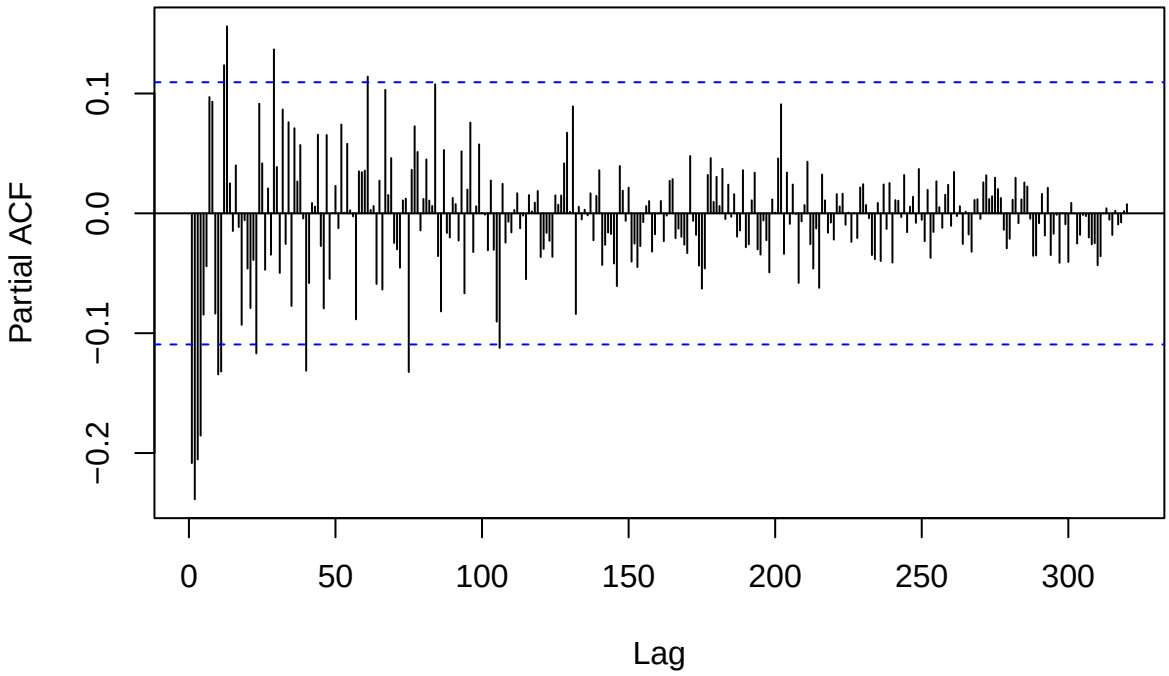
FAC Variacao de energia

```
x = na.omit(dados_energia$diff)
rho = acf(x, lag = length(x), plot = FALSE)
```



FACP Variacao de energia

```
phi = pacf(x, lag = length(x),plot = FALSE)
```



A forma da FAC parcial mostra uma forma de dependência autorregressiva com vários termos

Predicao

- Vamos separar a série temporal de duas partes: treinamento e teste.
  - Por exemplo, vamos usar n = 160 para treinamento para prever a observação 161. Depois vamos usar n = 161 para treinamento para prever a observação 162, e assim por diante.
- À medida que uma nova observação torna-se disponível, mantendo-se a estrutura do modelo, todos os parâmetros do modelo são reestimados novamente.
  - Por exemplo, se o tamanho da série de treinamento for n e se $h = 1$, para iniciar o teste dispondo-se de m observações, a próxima série de treinamento será de tamanho $n + 1$, e assim sucessivamente até a inclusão de todas as observações reservadas para o teste, perfazendo uma amostra final de tamanho $n + m$.

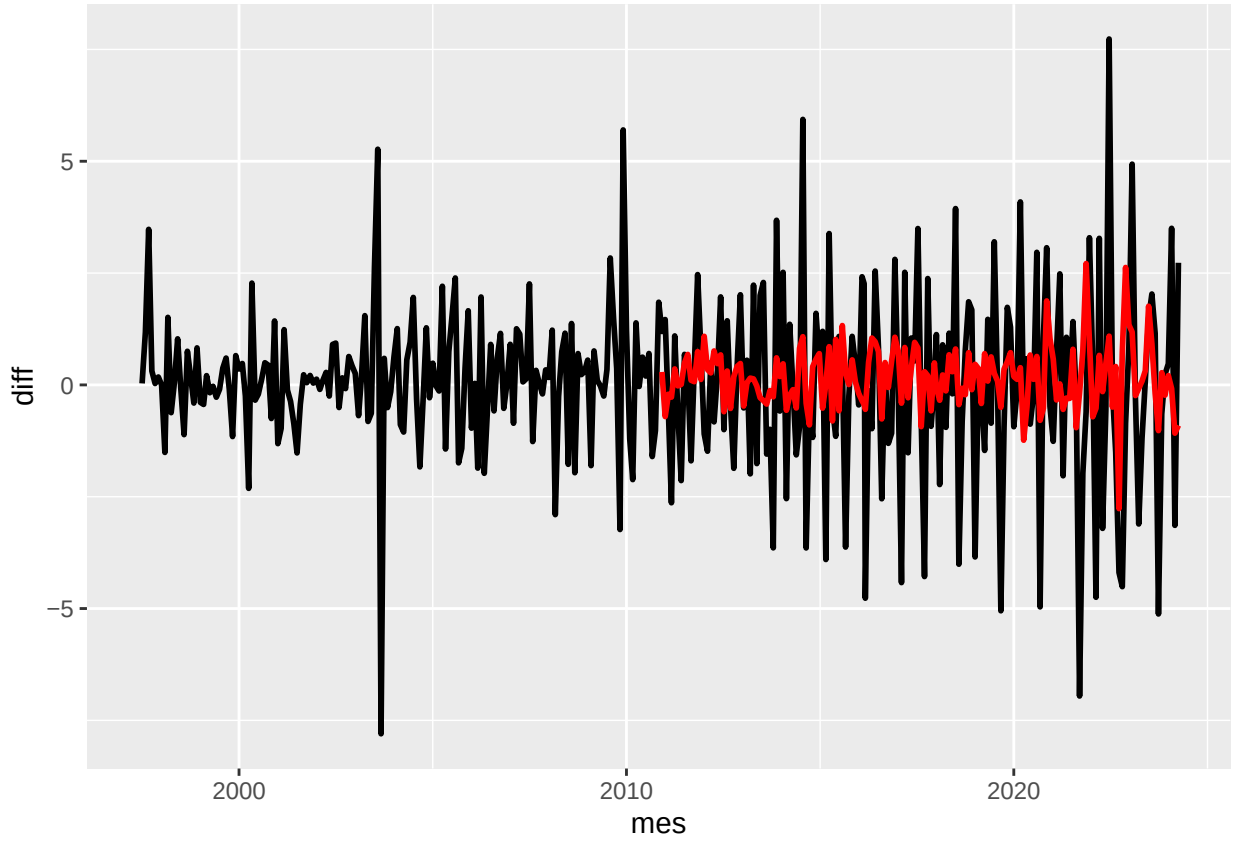
```
# Remover valores ausentes e preparar os dados
x = na.omit(dados_energia$diff) # Removendo os NA
n.size = length(x)  # Obtém o tamanho da amostra após a remoção de NA
n.training = ceiling(n.size/2)  # Calcula o tamanho do conjunto de treinamento
observed = NULL  # Inicializa a lista para armazenar as observações reais
predicted = NULL  # Inicializa a lista para armazenar as previsões

# Loop para previsão de observações
for (t in (n.training+1):n.size) {
  # Selecionar dados de treinamento até o ponto anterior ao ponto de teste atual
  x.training = x[1:(t-1)]

  # Calcular a função de autocorrelação até o lag correspondente ao ponto de teste
  rho = acf(x.training, lag = (t-1), plot = FALSE)
  last.lag = length(rho$acf)  # Obtém o índice do último valor na função de autocorrelação
  Rho = rho$acf  # Obtém a função de autocorrelação

  # Construir a matriz de covariância a partir da função de autocorrelação
  Omega = toeplitz(Rho[-last.lag])  # Constrói a matriz de Toeplitz a partir dos valores da FAC
  beta = inv(Omega) %*% Rho[-1]  # Estima os coeficientes do modelo de previsão

  # Calcular o termo constante do modelo de previsão
  beta.0 = mean(x.training) * (1 - sum(beta))  # Calcula o termo constante do modelo

  # Fazer a previsão para o ponto de teste atual usando o modelo construído
  predicted[t] = beta.0 + sum(rev(beta) * x.training[-1])  # Calcula a previsão para o ponto

  # Armazenar a observação real correspondente ao ponto de teste atual
  observed[t] = x[t]  # Armazena a observação real para fins de comparação posterior
}
```

Apos rodar esse codigo vamos ter interesse em analisar a nova variavel que criamos a `predicted` e podemos fazer uma comparacao direta com o que foi observado usando a variavel `observed`

```
dados_energia = dados_energia %>%
  mutate(observed = c(NA,observed),
         predicted = c(NA,predicted))
```

```
## Warning: Removed 1 row containing missing values (`geom_line()`).
```

```
## Warning: Removed 162 rows containing missing values (`geom_line()`).
```



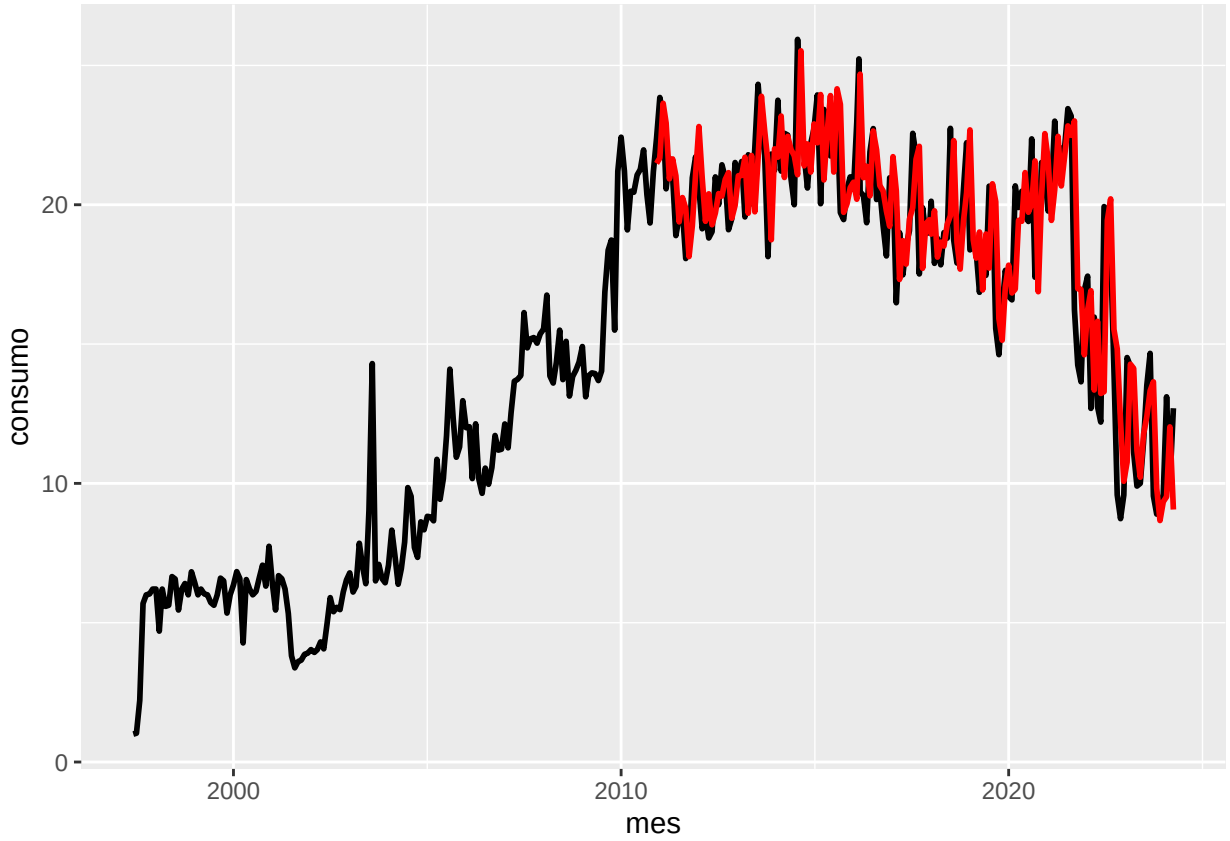
Com isso temos nossa funcao de autocorrelacao preditiva. Com isso podemos calcular nosso Y usando os valores preditos e comparar com a serie original e com os valores reais observados:

```
Y.hat = NULL  # Inicializa uma lista para armazenar as previsões da série original
Y.t = NULL  # Inicializa uma lista para armazenar os valores da série original

for (h in (n.training+1):n.size) {
  # Previsão do próximo valor na série original com base nas diferenças entre os valores
  Y.t[h+1] = dados_energia$consumo[h] + dados_energia$diff[h+1]

  # Previsão do próximo valor na série original com base nas previsões feitas anteriormente
  Y.hat[h+1] = dados_energia$consumo[h] + dados_energia$predicted[h+1]
}
```

Podemos adicioanr esse vetores no DF

```
dados_energia = dados_energia %>%
  mutate(Y.hat = Y.hat,
         Y.t = Y.t)
```

```
## Warning: Removed 162 rows containing missing values (`geom_line()`).
```



Resultado de Abril 2024

```
tail(dados_energia,n = c(1,2))
```

```
## # A tibble: 1 x 2
##   Y.hat   Y.t
##   <dbl> <dbl>
## 1  9.06  12.7
```