Modulo 4

Heitor Pereira Nunes F. Cunha - 12111EBI027

Uberlândia - UFU

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
GRADUAÇÃO EM ENGENHARIA BIOMÉDICA

Heitor Pereira Nunes Fernandes Cunha

Processamento de Sinais Biomédicos: Módulo 4

Uberlândia, MG
2025

Questão 1

A avaliação de estatísticas de um sinal biomédico é de grande relevância para a caracterização do sinal e para o desenvolvimento de dispositivos. Por exemplo, ao monitorarmos a atividade eletromiográfica ao longo do tempo e estimarmos estatísticas (e.g., a média do valor absoluto) do sinal é possível realizar o controle de dispositivos miolétricos. De forma genérica, o sinal eletromiográfico pode ser simulado a partir de amostras de uma distribuição normal, que possui uma média zero e desvio padrão unitário. Considerando essas informações você deve:

Simular um sinal eletromiográfico, EMG1, amostrado a 1000 Hz, de duração, t, de 30 segundos, cuja a média é 0.8 e o devio padrão é 1.3. Deve-se plotar o gráfico mostrando o sinal gerado.

```
library(pracma)
```

```
## Warning: package 'pracma' was built under R version 4.3.3
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

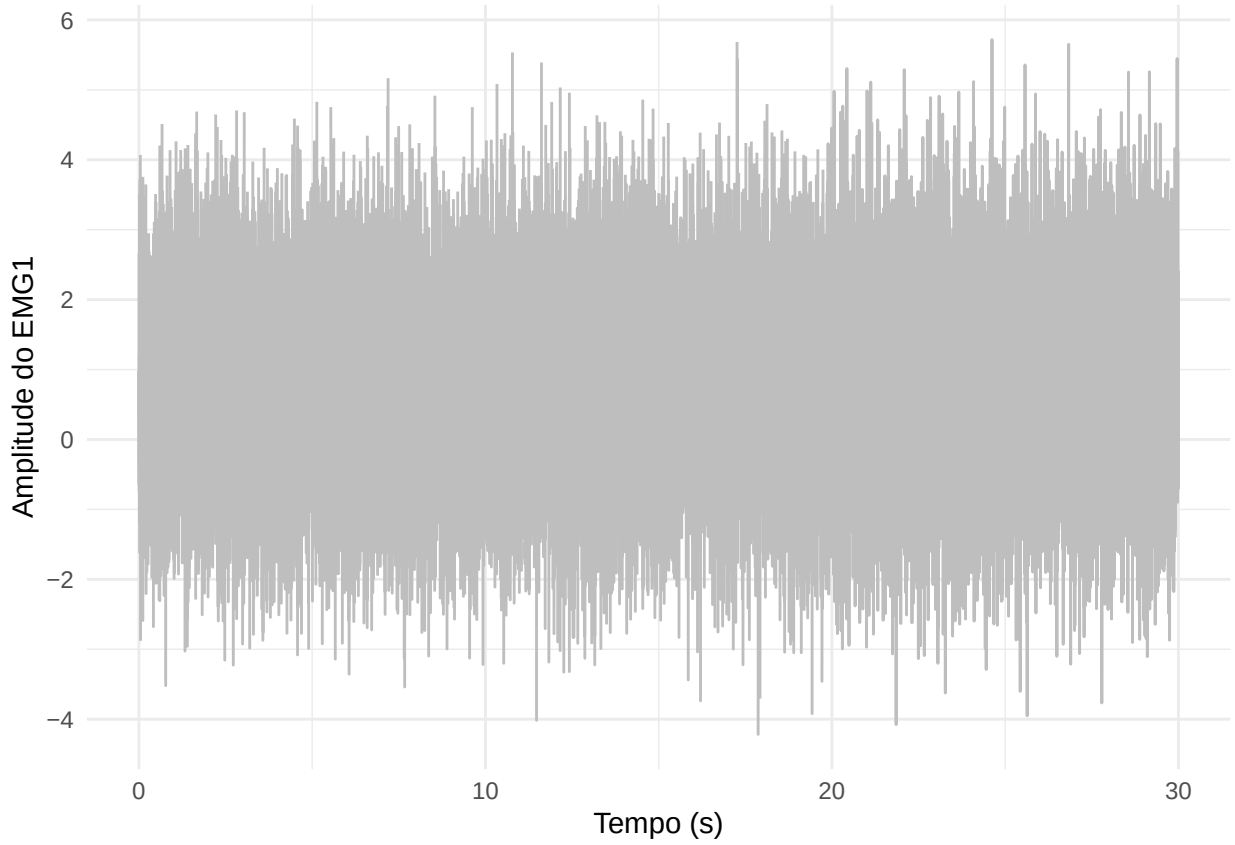
```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
fs1 <- 1000
dt1 <- 1/fs1
tempo <- seq(0, 30, by = dt1)

media1 <- 0.8
desvio_padrao1 <- 1.3

# Gerando ruído com distribuição normal
ruido1 <- randn(length(tempo), 1)
EMG1 <- media1 + desvio_padrao1 * ruido1

dados_EMG1 <- data.frame(Tempo = tempo, Amplitude = EMG1)

grafico_EMG1 <- ggplot(dados_EMG1, aes(x = Tempo, y = Amplitude)) +
  geom_line(color = "grey") +
```

```
  labs(x = "Tempo (s)", y = "Amplitude do EMG1") +
  theme_minimal()

# Exibe o gráfico do EMG1
grafico_EMG1
```



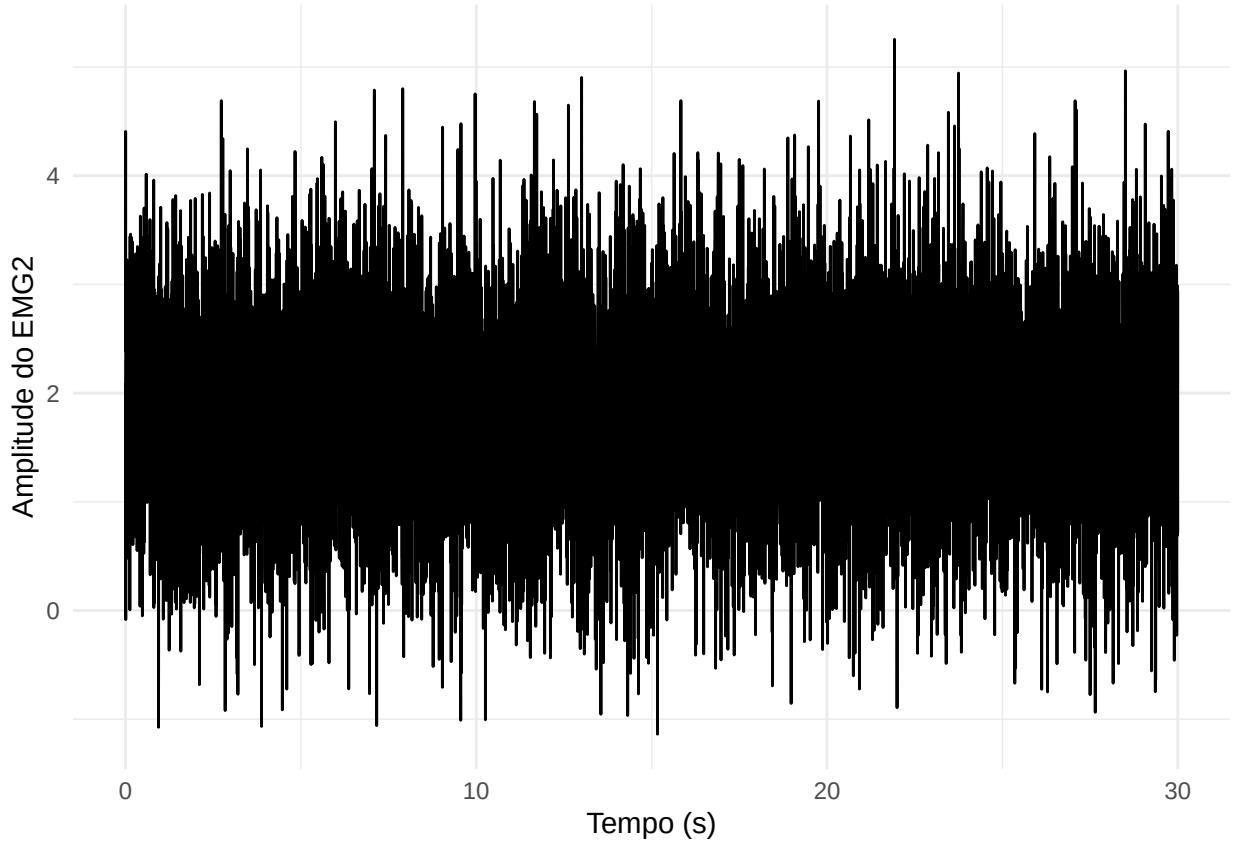
Simular um sinal eletromiográfico, EMG2, amostrado a 300 Hz, de duração, t, de 30 segundos, cuja a média é 1.8 e o devio padrão é 0.9. Deve-se plotar o gráfico mostrando o sinal gerado.

```
library(pracma)
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3

## Warning: package 'tibble' was built under R version 4.3.3

## Warning: package 'tidyr' was built under R version 4.3.3

## Warning: package 'readr' was built under R version 4.3.3

## Warning: package 'purrr' was built under R version 4.3.3

## Warning: package 'stringr' was built under R version 4.3.3
```

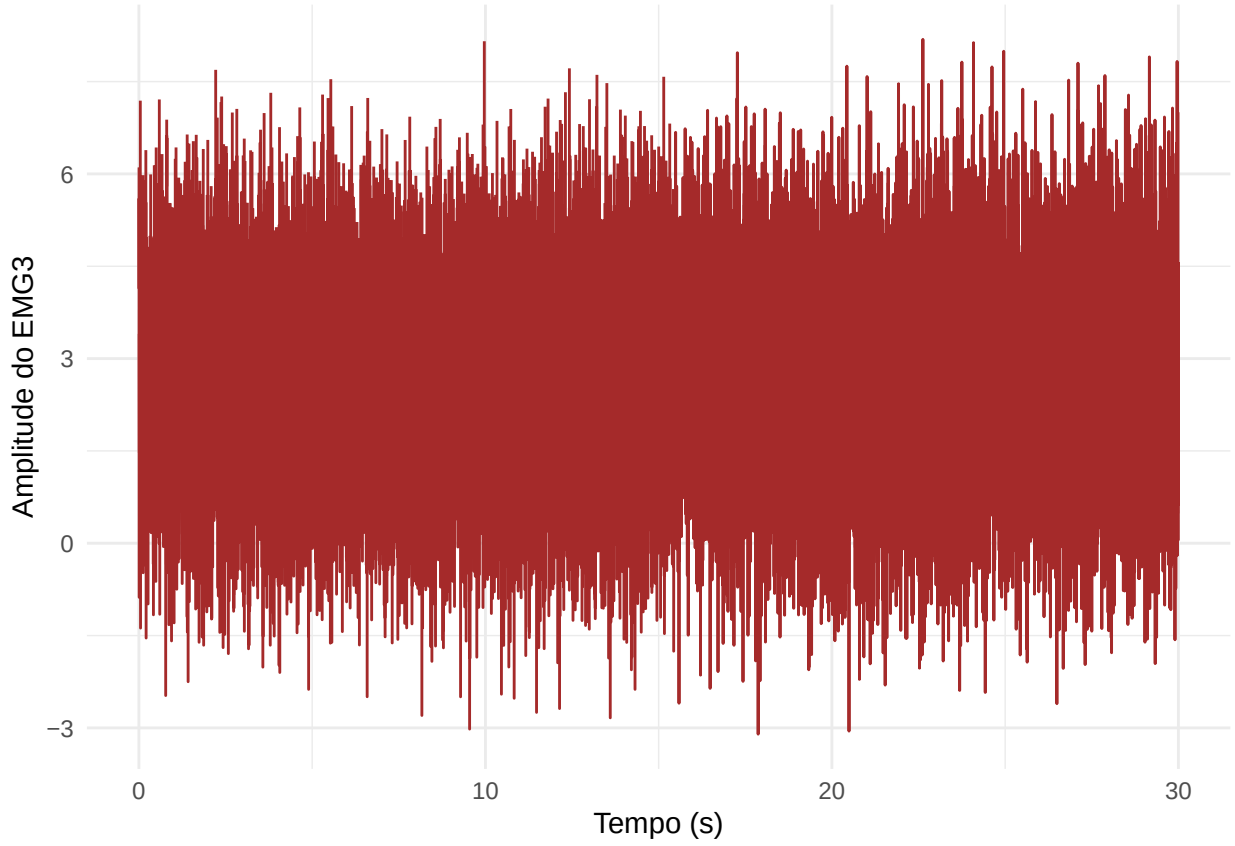
```
## Warning: package 'forcats' was built under R version 4.3.3

## Warning: package 'lubridate' was built under R version 4.3.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
## v readr     2.1.5
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x purrr::cross()  masks pracma::cross()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
fs2 <- 300
dt2 <- 1/fs2
tempo2 <- seq(0, 30, by = dt2)

media2 <- 1.8
desvio_padrao2 <- 0.9

# Gerando ruído para o EMG2
ruido2 <- randn(length(tempo2), 1)
EMG2 <- media2 + desvio_padrao2 * ruido2

dados_EMG2 <- data.frame(Tempo = tempo2, Amplitude = EMG2)

grafico_EMG2 <- ggplot(dados_EMG2, aes(x = Tempo, y = Amplitude)) +
  geom_line(color = "black") +
  labs(x = "Tempo (s)", y = "Amplitude do EMG2") +
  theme_minimal()

# Exibe o gráfico do EMG2
grafico_EMG2
```



Estimar o sinal EMG3 = EMG1 + EMG2, amostrado na mesma frequência de amostragem que EMG1. Deve-se plotar o gráfico mostrando o sinal gerado

```
# Realizando interpolação para EMG2
df_c <- data.frame(spline(x = tempo2, y = EMG2, xout = tempo))
amplitude_interpolado <- df_c$y

# Calculando o EMG3
EMG3 <- EMG1 + amplitude_interpolado

dados_EMG3 <- data.frame(Tempo = tempo, Amplitude = EMG3)

grafico_EMG3 <- ggplot(dados_EMG3, aes(x = Tempo, y = Amplitude)) +
  geom_line(color = "brown") +
  labs(x = "Tempo (s)", y = "Amplitude do EMG3") +
  theme_minimal()

# Exibe o gráfico do EMG3
grafico_EMG3
```



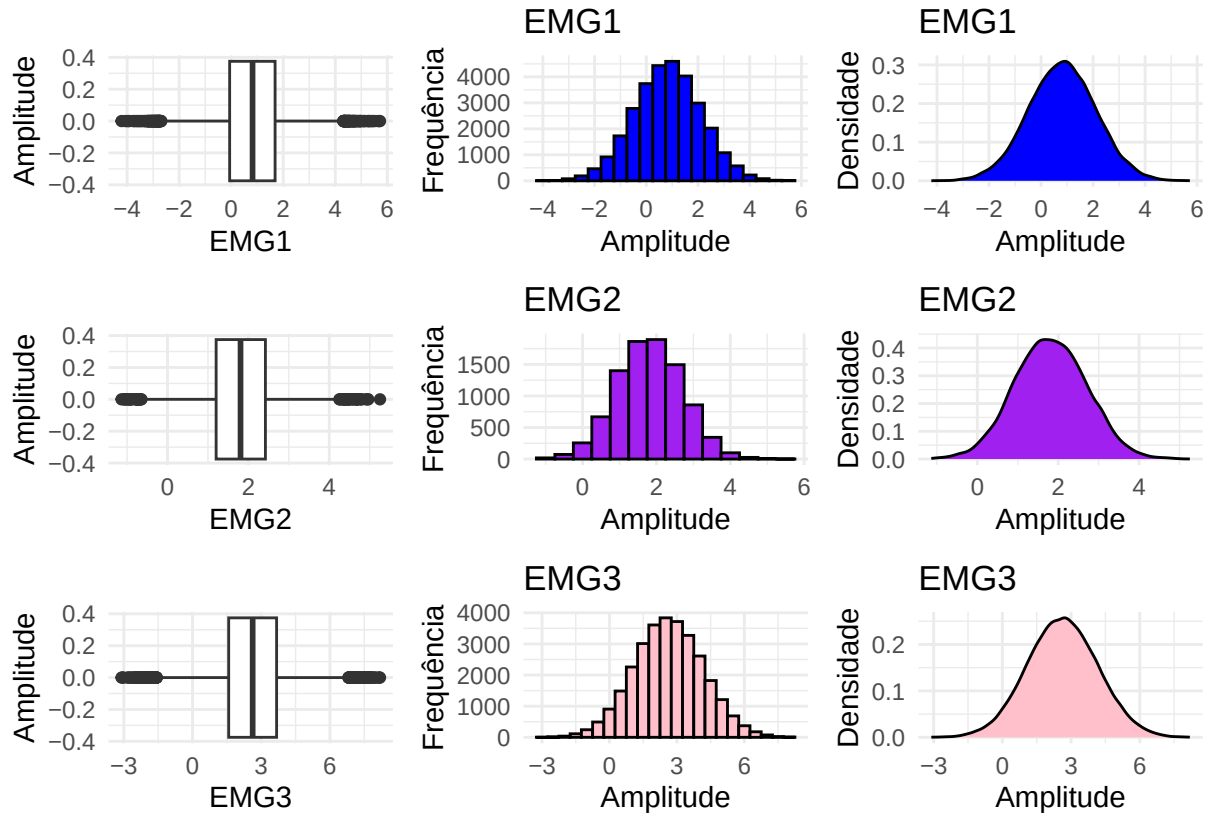
Gerar o boxplot, histograma e gráfico da densidade de cada um dos sinais, EMG1, EMG2 e EMG3. Os resultados devem gerar uma única imagem, contendo um painel dividido em três linhas e três colunas. A coluna 1 deve ser destinada ao boxplot, a dois ao histograma e a três à densidade.

```
library(ggplot2)
library(patchwork)
```

```
## Warning: package 'patchwork' was built under R version 4.3.3
```

```
# Boxplot do sinal EMG1
box_emg1 <- ggplot(dados_EMG1, aes(x = Amplitude)) +
  geom_boxplot() +
  labs(x = "EMG1", y = "Amplitude") +
  theme_minimal()

# Histograma do sinal EMG1
hist_emg1 <- ggplot(dados_EMG1, aes(x = Amplitude)) +
  geom_histogram(binwidth = 0.5, fill = "blue", color = "black") +
  labs(title = "EMG1", x = "Amplitude", y = "Frequência") +
  theme_minimal()

# Densidade do sinal EMG1
densid_emg1 <- ggplot(dados_EMG1, aes(x = Amplitude)) +
  geom_density(fill = "blue", color = "black") +
```

```
  labs(title = "EMG1", x = "Amplitude", y = "Densidade") +
  theme_minimal()

# Boxplot do sinal EMG2
box_emg2 <- ggplot(dados_EMG2, aes(x = Amplitude)) +
  geom_boxplot() +
  labs(x = "EMG2", y = "Amplitude") +
  theme_minimal()

# Histograma do sinal EMG2
hist_emg2 <- ggplot(dados_EMG2, aes(x = Amplitude)) +
  geom_histogram(binwidth = 0.5, fill = "purple", color = "black") +
  labs(title = "EMG2", x = "Amplitude", y = "Frequência") +
  theme_minimal()

# Densidade do sinal EMG2
densid_emg2 <- ggplot(dados_EMG2, aes(x = Amplitude)) +
  geom_density(fill = "purple", color = "black") +
  labs(title = "EMG2", x = "Amplitude", y = "Densidade") +
  theme_minimal()

# Boxplot do sinal EMG3
box_emg3 <- ggplot(dados_EMG3, aes(x = Amplitude)) +
  geom_boxplot() +
  labs(x = "EMG3", y = "Amplitude") +
  theme_minimal()

# Histograma do sinal EMG3
hist_emg3 <- ggplot(dados_EMG3, aes(x = Amplitude)) +
  geom_histogram(binwidth = 0.5, fill = "pink", color = "black") +
  labs(title = "EMG3", x = "Amplitude", y = "Frequência") +
  theme_minimal()

# Densidade do sinal EMG3
densid_emg3 <- ggplot(dados_EMG3, aes(x = Amplitude)) +
  geom_density(fill = "pink", color = "black") +
  labs(title = "EMG3", x = "Amplitude", y = "Densidade") +
  theme_minimal()

# Combinando os gráficos em um painel
(box_emg1 | hist_emg1 | densid_emg1) /
(box_emg2 | hist_emg2 | densid_emg2) /
(box_emg3 | hist_emg3 | densid_emg3)
```



Calcular a média, desvio padrão, variância, coeficiente de assimetria e curtose dos sinais EMG1, EMG2 e EMG3. Os resultados devem ser apresentados com três casas decimais.

```
library(kableExtra)
```

```
## Warning: package 'kableExtra' was built under R version 4.3.3
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
library(e1071)
```

```
##
## Attaching package: 'e1071'
```

```
## The following object is masked from 'package:pracma':
##
##     sigmoid
```

```r
library(dplyr)

# Calculando a média com a função mean()
media_EMG1 <- mean(dados_EMG1$Amplitude)
media_EMG2 <- mean(dados_EMG2$Amplitude)
media_EMG3 <- mean(dados_EMG3$Amplitude)

# Arredondando o resultado para 3 casas decimais
media_EMG1_3 <- round(media_EMG1, 3)
media_EMG2_3 <- round(media_EMG2, 3)
media_EMG3_3 <- round(media_EMG3, 3)

# Calculando o desvio padrão
dp_EMG1 <- sd(dados_EMG1$Amplitude)
dp_EMG2 <- sd(dados_EMG2$Amplitude)
dp_EMG3 <- sd(dados_EMG3$Amplitude)

# Arredondando o resultado para 3 casas decimais
dp_EMG1_3 <- round(dp_EMG1, 3)
dp_EMG2_3 <- round(dp_EMG2, 3)
dp_EMG3_3 <- round(dp_EMG3, 3)

# Calculando a variância
v_EMG1 <- var(dados_EMG1$Amplitude)
v_EMG2 <- var(dados_EMG2$Amplitude)
v_EMG3 <- var(dados_EMG3$Amplitude)

# Arredondando o resultado para 3 casas decimais
v_EMG1_3 <- round(v_EMG1, 3)
v_EMG2_3 <- round(v_EMG2, 3)
v_EMG3_3 <- round(v_EMG3, 3)

# Calculando o coeficiente de assimetria
coefassim_EMG1 <- skewness(dados_EMG1$Amplitude)
coefassim_EMG2 <- skewness(dados_EMG2$Amplitude)
coefassim_EMG3 <- skewness(dados_EMG3$Amplitude)

# Arredondando resultado para 3 casas decimais
coefassim_EMG1_3 <- round(coefassim_EMG1, 3)
coefassim_EMG2_3 <- round(coefassim_EMG2, 3)
coefassim_EMG3_3 <- round(coefassim_EMG3, 3)

# Calculando a curtose
curt_EMG1 <- kurtosis(dados_EMG1$Amplitude)
curt_EMG2 <- kurtosis(dados_EMG2$Amplitude)
curt_EMG3 <- kurtosis(dados_EMG3$Amplitude)

# Arredondando resultado para 3 casas decimais
curt_EMG1_3 <- round(curt_EMG1, 3)
curt_EMG2_3 <- round(curt_EMG2, 3)
curt_EMG3_3 <- round(curt_EMG3, 3)

# Criando um dataframe com os resultados
```

sinais	mean	dp	var	ca	curtose
EMG1	0.814	1.296	1.679	-0.011	-0.006
EMG2	1.814	0.892	0.795	0.037	-0.011
EMG3	2.628	1.539	2.368	0.012	-0.041

```
df <- data.frame(sinais = c("EMG1", "EMG2", "EMG3"),
                 mean = c(media_EMG1_3, media_EMG2_3, media_EMG3_3),
                 dp = c(dp_EMG1_3, dp_EMG2_3, dp_EMG3_3),
                 var = c(v_EMG1_3, v_EMG2_3, v_EMG3_3),
                 ca = c(coefassim_EMG1_3, coefassim_EMG2_3, coefassim_EMG3_3),
                 curtose = c(curt_EMG1_3, curt_EMG2_3, curt_EMG3_3))

# Exibindo o dataframe formatado com kableExtra
df %>% kbl() %>% kable_styling()
```

Questão 2

Considerando o conceito de precisão e acurácia disponível, você deverá:

Propor uma questão que simule a geração de sinais com (i) alta precisão e alta exatidão, (ii) baixa precisão e alta exatidão, (iii) alta precisão e baixa exatidão e (iv) baixa precisão e baixa exatidão.

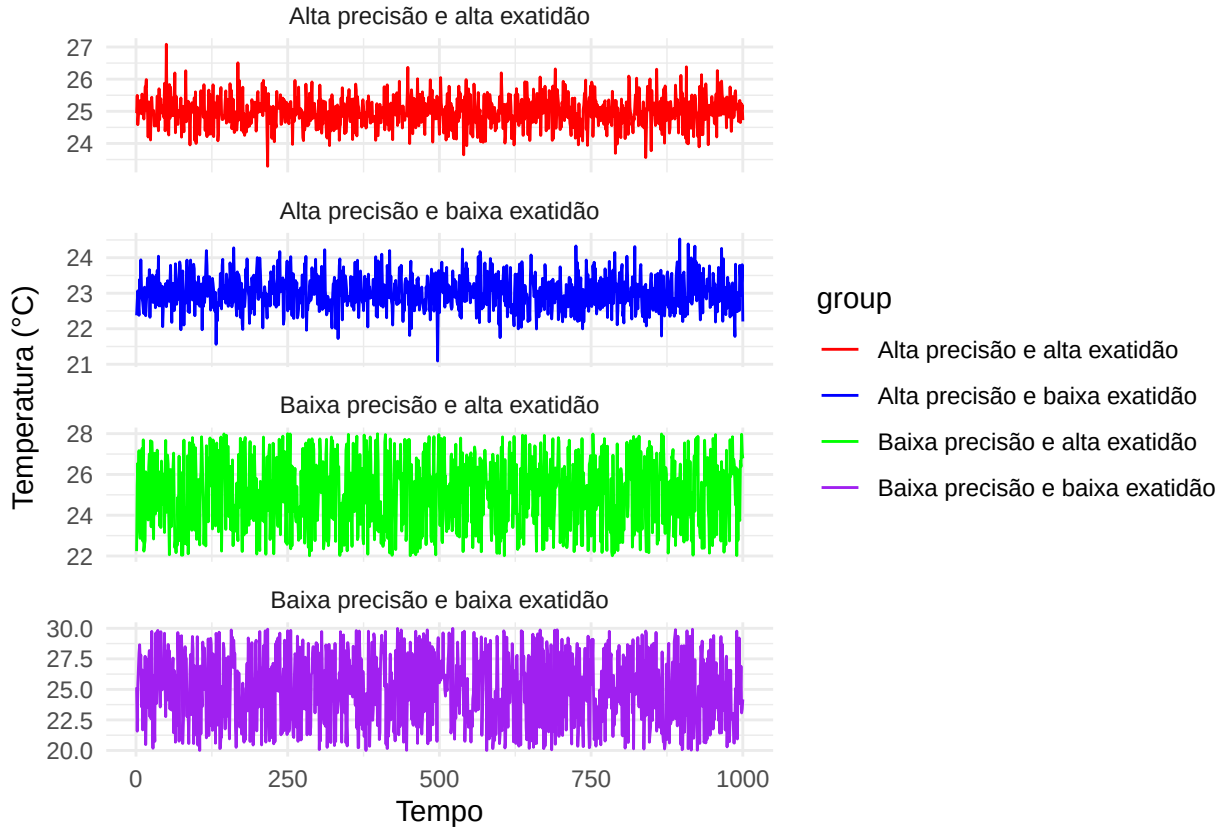
Suponha que você seja um pesquisador monitorando a temperatura dentro de uma estufa agrícola. Para garantir o crescimento ideal das plantas, a temperatura deve ser mantida em torno de 25°C. Você coletará dados de temperatura em quatro cenários distintos:

1. Alta precisão e alta exatidão: A temperatura varia pouco em torno do valor desejado, por exemplo, distribuição normal com média de 25°C e desvio padrão de 0,5°C.
2. Baixa precisão e alta exatidão: A temperatura varia bastante, mas ainda mantém a média em torno do valor ideal, por exemplo, distribuição uniforme entre 22°C e 28°C.
3. Alta precisão e baixa exatidão: A temperatura se mantém estável, mas abaixo do ideal, por exemplo, distribuição normal com média de 23°C e desvio padrão de 0,5°C.
4. Baixa precisão e baixa exatidão: A temperatura varia muito e não se concentra no valor desejado, por exemplo, distribuição uniforme entre 20°C e 30°C.

Desenvolver um programa em R que solucione o problema proposto. Os sinais gerados devem ser plotados em uma única imagem, dividida em uma coluna e quatro linhas. Deve-se incluir unidades e legenda para cada gráfico.

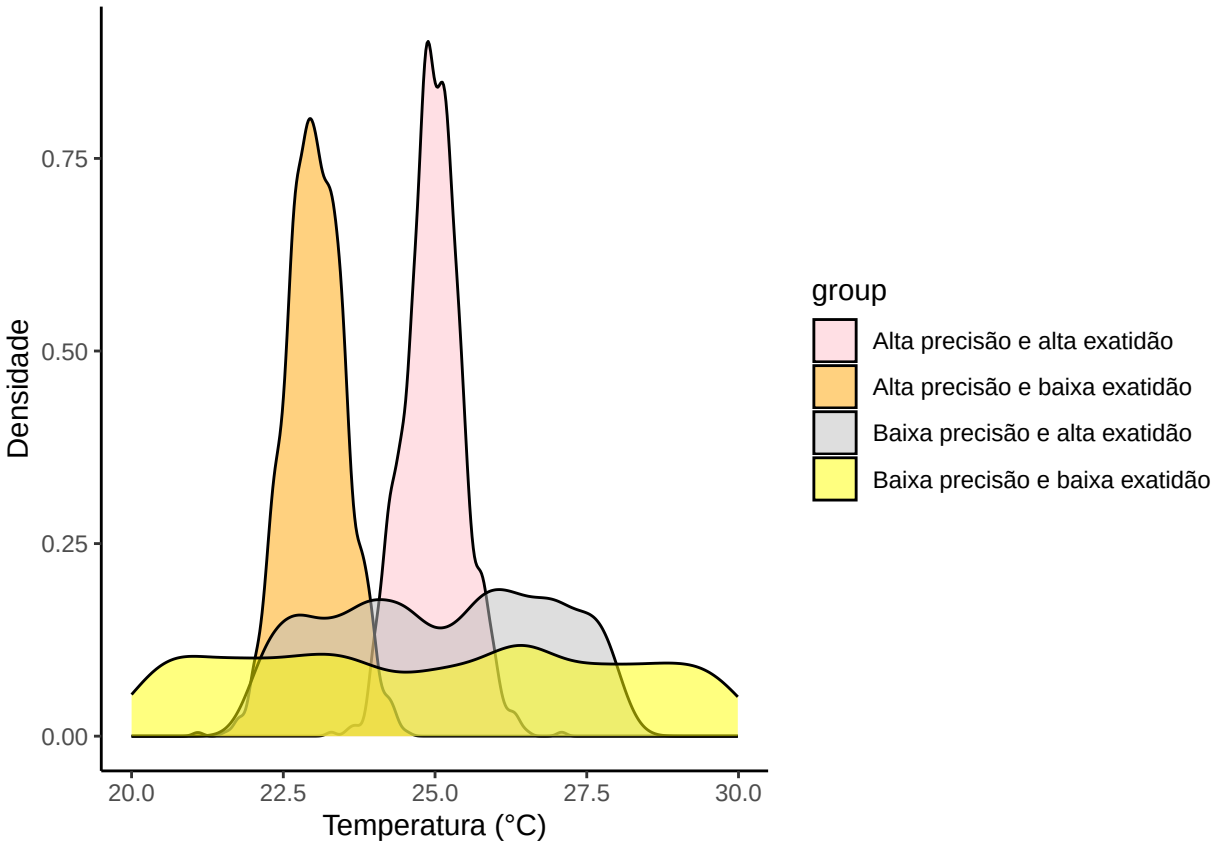
```r
library(tidyverse)

# Amplitudes do Sinal (1) - Alta precisão e alta exatidão
temp1 <- rnorm(1000, mean = 25, sd = 0.5)
temp1 <- pmax(pmin(temp1, 30), 20)

# Amplitudes do Sinal (2) - Baixa precisão e alta exatidão
temp2 <- runif(1000, min = 22, max = 28)
temp2 <- pmax(pmin(temp2, 30), 20)

# Amplitudes do Sinal (3) - Alta precisão e baixa exatidão
temp3 <- rnorm(1000, mean = 23, sd = 0.5)
temp3 <- pmax(pmin(temp3, 30), 20)

# Amplitudes do Sinal (4) - Baixa precisão e baixa exatidão
temp4 <- runif(1000, min = 20, max = 30)
temp4 <- pmax(pmin(temp4, 30), 20)

# Criando vetor de tempo
t <- rep(1:1000, 4)

# Criando data frames para os sinais
df1 <- data.frame(temperatura = temp1, t = t, group = 'Alta precisão e alta exatidão')
df2 <- data.frame(temperatura = temp2, t = t, group = 'Baixa precisão e alta exatidão')
df3 <- data.frame(temperatura = temp3, t = t, group = 'Alta precisão e baixa exatidão')
```

```r
df4 <- data.frame(temperatura = temp4, t = t, group = 'Baixa precisão e baixa exatidão')

# Unindo os data frames
df <- rbind(df1, df2, df3, df4)

# Plotando os sinais ao longo do tempo
ggplot(df, aes(x = t, y = temperatura, color = group)) +
  geom_line() +
  scale_color_manual(values = c("red", "blue", "green", "purple")) +
  facet_wrap(~ group, ncol = 1, scales = "free_y") +
  xlab("Tempo") +
  ylab("Temperatura (°C)") +
  theme_minimal()
```



Plota um gráfico, usando o ggplot, reportando os resultados como exemplificado na Figura 1

```r
# Plotando as distribuições das temperaturas
ggplot(df, aes(x = temperatura, fill = group)) +
  geom_density(alpha = 0.5) +
  scale_fill_manual(values = c("pink", "orange", "gray", "yellow")) +
  xlab("Temperatura (°C)") +
  ylab("Densidade") +
  theme_classic()
```


Densidade
0.75
0.50
0.25
0.00
20.0
22.5
25.0
27.5
30.0
Temperatura (°C)
group
Alta precisão e alta exatidão
Alta precisão e baixa exatidão
Baixa precisão e alta exatidão
Baixa precisão e baixa exatidão

Questão 3

O conjunto de sinais eletromiográficos, DadosM4-1.txt, disponível na plataforma moodle, foi originado a partir de uma coleta simultânea de dados eletromiográficos e inerciais. Os dados foram coletados seguindo o seguinte protocolo experimental:

1. Os sensores de eletromiografia foram posicionados no músculo tibial anterior e nos músculos do tríceps sural. O acelerômetro foi posicionado nos dois terços proximais da parte lateral da perna, com o eixo y contra a gravidade.
2. Com o sujeito na posição ortostática realizou-se o movimento de dorsiflexão e flexão plantar. No retorno da flexão realizou-se um contato brusco do calcanhar com o solo. 3 Foram realizadas 60 repetições da tarefa, sem descanso.

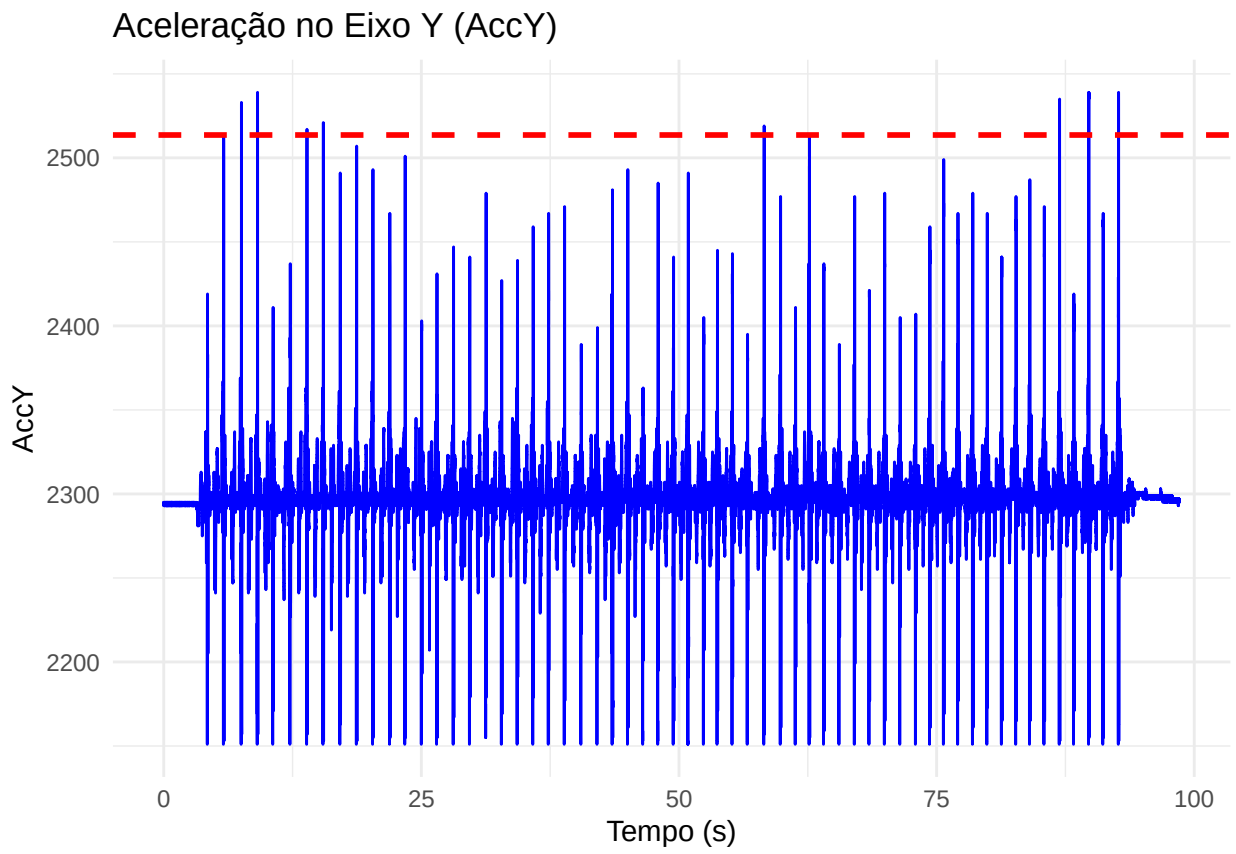
Faça a estimativa de parâmetros estatísticos dos sinais eletromiográficos disponíveis no arquivo DadosM4-1.txt. Os seguintes passos devem ser executados:

```
library(dygraphs)
```

```
## Warning: package 'dygraphs' was built under R version 4.3.3
```

```
library(tidyverse)

# Carregar os dados a partir do arquivo de texto
dados <- read.table("E:/GitHub/PSB/Módulo 4 - PSB/DadosM4-1.txt", header = FALSE, sep = " ", skip = 6)

# Atribuindo nomes significativos às colunas
colnames(dados) <- c("AccX", "AccY", "MuscAnterior", "MuscPosterior")

# Definindo a frequência de amostragem e o intervalo de tempo
fs <- 500
dt <- 1 / fs
tempo <- seq(0, dt * (nrow(dados) - 1), by = dt)

# Adicionando o vetor de tempo ao data frame
dados <- cbind(time = tempo, dados)
limite_90 <- 0.9 * max(dados$AccY, na.rm = TRUE) * 1.1

# Plotando o gráfico de AccY
ggplot(dados, aes(x = time, y = AccY)) +
  geom_line(color = "blue") +
  geom_hline(yintercept = limite_90, color = "red", linetype = "dashed", size = 1) +
  labs(title = "Aceleração no Eixo Y (AccY)",
       x = "Tempo (s)",
       y = "AccY") +
  theme_minimal()
```

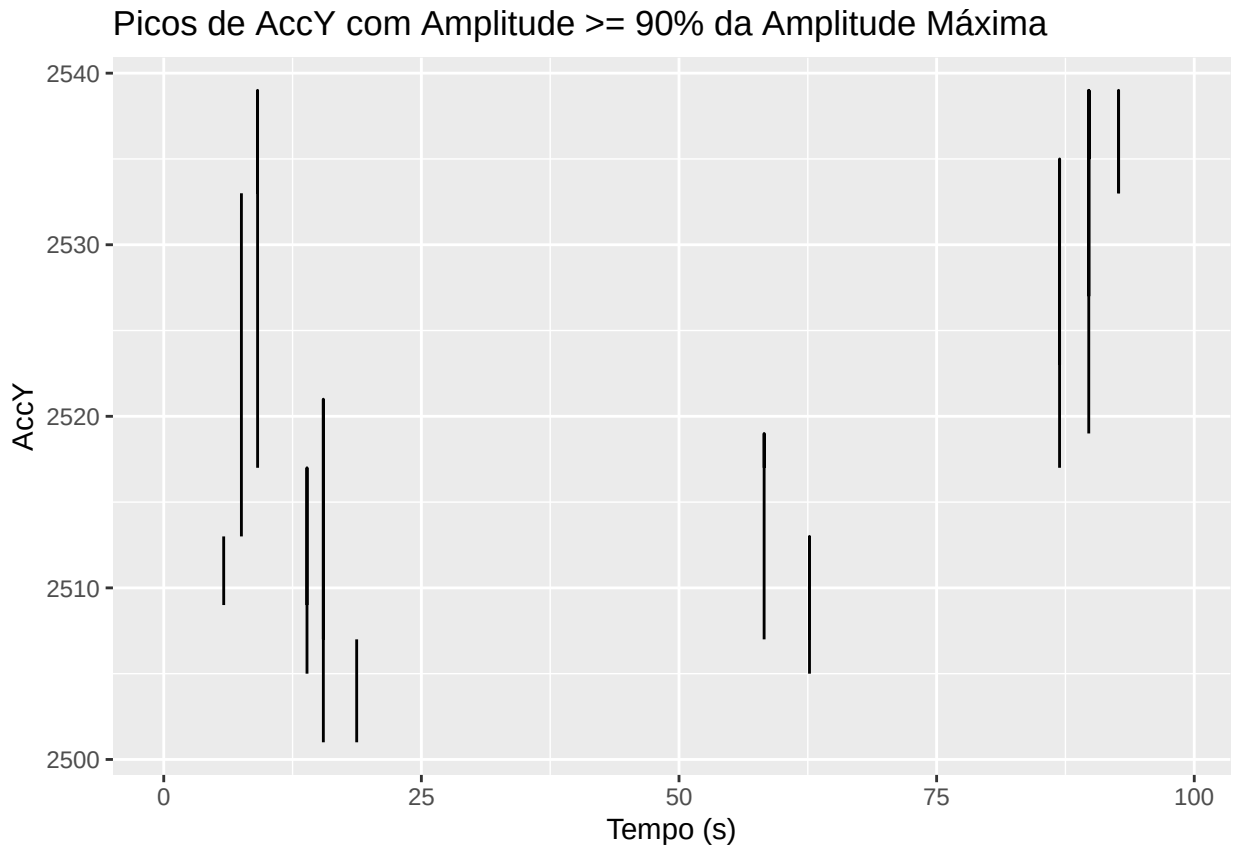
```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```




```r
# Amplitude Máxima
AmpMAX_AccY <- max(dados$AccY) - min(dados$AccY)

lim_min <- 0.9 * AmpMAX_AccY

# Encontrando os picos e armazenando os valores

dados <- dados %>%
  mutate(pico = ifelse(dados$AccY - min(dados$AccY) >= lim_min, dados$AccY, NA))
```

Gere um gráfico ocm os picos encontrados na letra a).

```r
# Plotando o gráfico com os picos encontrados
ggplot(dados, aes(x = time, y = pico)) +
  geom_line(na.rm = TRUE) +
  labs(title = "Picos de AccY com Amplitude >= 90% da Amplitude Máxima", x = "Tempo (s)", y = "AccY")
```



Calcule as estatísticas Média, Mediana, Moda, Amplitude, Variância, Coeficiente de Variação, Distância Interquartil para cada janela de sinal do músculo tibial anterior. O tamanho da janela deve ser de 500 ms, e o início da mesma deve ser a partir de cada um dos picos detectados na letra a).

```r
# Selecionando o sinal do músculo tibial anterior (MuscAnterior)
sinal <- dados$MuscAnterior

# Definindo o tamanho da janela de 500 ms
janela <- 500

# Localizando os índices dos picos
picos_indice <- which(!is.na(dados$pico))

# Criando um data frame para armazenar os resultados
resultados <- data.frame()

# Loop para calcular as estatísticas para cada janela
for (i in picos_indice) {
  # Determinando os índices de início e fim da janela
  indx_inicio <- i - floor(janela / 2)
  indx_fim <- i + ceiling(janela / 2) - 1

  # Verificando se a janela está dentro dos limites dos dados
  if (indx_inicio > 0 & indx_fim <= length(sinal)) {
```

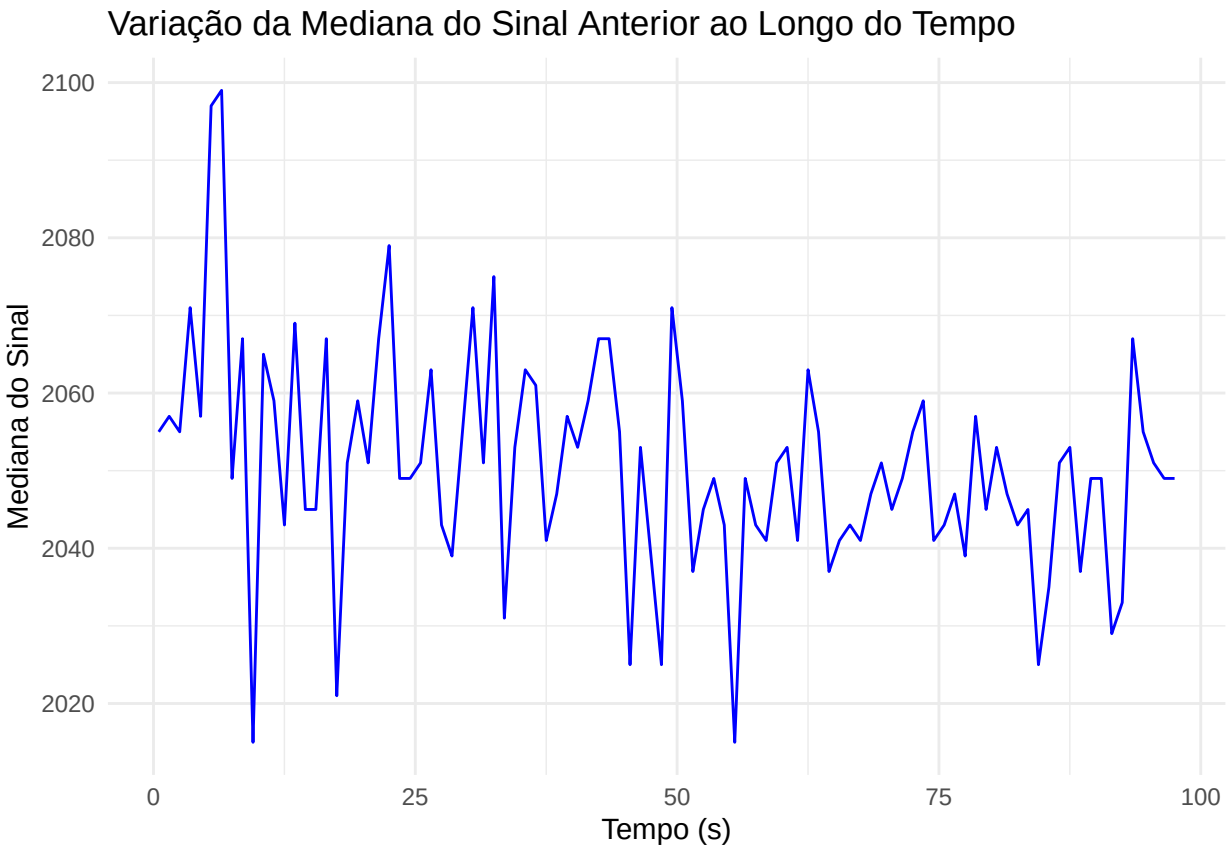
```r
    # Calculando as estatísticas para a janela
    media <- mean(sinal[indx_inicio:indx_fim])
    mediana <- median(sinal[indx_inicio:indx_fim])
    moda <- as.numeric(names(table(sinal[indx_inicio:indx_fim]))[which.max(table(sinal[indx_inicio:indx
    amplitude <- max(sinal[indx_inicio:indx_fim]) - min(sinal[indx_inicio:indx_fim])
    variancia <- var(sinal[indx_inicio:indx_fim])
    cv <- sd(sinal[indx_inicio:indx_fim]) / mean(sinal[indx_inicio:indx_fim])
    iqr <- IQR(sinal[indx_inicio:indx_fim])

    # Armazenando os resultados em um novo data frame
    resultados <- rbind(resultados, data.frame(pico = i, media = media, mediana = mediana, moda = moda,
                                               amplitude = amplitude, variancia = variancia, cv = cv, i
  }
}

# Exibindo os resultados
resultados
```

```
##     pico    media mediana moda amplitude variancia        cv    iqr
## 1   2910 2082.654    2078 2019      4095  653798.5 0.3882441  904.0
## 2   2911 2084.130    2078 2019      4095  650443.1 0.3869723  904.0
## 3   3768 2064.892    2050    0      4095  770473.4 0.4250906 1039.0
## 4   3769 2067.294    2050    0      4095  764776.1 0.4230239 1039.0
## 5   3770 2069.798    2054    0      4095  766845.8 0.4230835 1040.5
## 6   4552 2080.624    2049 4095      4095  808160.6 0.4320711 1062.0
## 7   4553 2079.676    2049 4095      4095  807913.9 0.4322021 1062.0
## 8   4554 2081.454    2049 4095      4095  805832.8 0.4312764 1055.0
## 9   4555 2084.892    2049 4095      4095  797760.2 0.4284031 1047.0
## 10  6950 2059.990    2063    0      4095  794359.6 0.4326567 1046.5
## 11  6951 2060.652    2063    0      4095  793403.3 0.4322573 1046.5
## 12  6952 2056.918    2062    0      4095  792963.9 0.4329221 1055.5
## 13  7742 2081.624    2043    0      4095  690765.3 0.3992666  829.0
## 14  7743 2083.950    2044    0      4095  690144.0 0.3986416  820.0
## 15  7744 2084.610    2044    0      4095  689601.4 0.3983586  816.0
## 16  7745 2086.066    2045    0      4095  690113.9 0.3982285  819.0
## 17  9362 2056.284    2050    0      4095  718787.8 0.4123037  869.0
## 18  9363 2051.844    2048    0      4095  710504.5 0.4108082  851.5
## 19 11714 2068.334    2047    0      4095  716179.4 0.4091572  895.5
## 20 29131 2040.168    2047 4095      4095  880356.6 0.4599000 1102.0
## 21 29132 2039.480    2047 4095      4095  881746.1 0.4604180 1102.0
## 22 29133 2039.484    2047 4095      4095  881737.7 0.4604149 1102.0
## 23 31331 2086.052    2059    0      4095  764100.9 0.4190349  819.5
## 24 31332 2084.212    2059    0      4095  763548.3 0.4192532  805.5
## 25 31333 2080.362    2058    0      4095  758508.5 0.4186406  801.5
## 26 43466 2070.006    2045 4095      4095  590145.5 0.3711145  579.0
## 27 43467 2071.158    2045 4095      4095  589869.4 0.3708213  574.0
## 28 43468 2069.746    2045 4095      4095  588772.8 0.3707292  563.5
## 29 43469 2067.486    2045 4095      4095  586592.6 0.3704467  556.5
## 30 44883 2085.334    2047 2043      4095  576484.5 0.3640979  608.5
## 31 44884 2084.706    2046 2043      4095  576395.1 0.3641794  608.5
## 32 44885 2082.722    2045 2043      4095  574712.2 0.3639937  604.5
## 33 44886 2079.804    2045 2043      4095  570611.5 0.3632017  599.0
## 34 44887 2077.346    2045 2043      4095  567661.3 0.3626902  587.0
## 35 44888 2073.322    2045 2043      4095  559487.0 0.3607682  579.0
```

```
## 36 44889 2076.140    2046 2043      4095  555555.2 0.3590103  577.5
## 37 44890 2073.106    2046 2043      4095  550892.4 0.3580238  568.5
## 38 46328 2063.944    2028 4095      4095  610599.6 0.3785998  649.0
## 39 46329 2065.778    2028 4095      4095  608040.2 0.3774701  648.0
## 40 46330 2063.826    2026 4095      4095  607452.7 0.3776445  646.0
## 41 46331 2066.700    2026 4095      4095  600894.0 0.3750779  646.0
## 42 46332 2066.812    2026 4095      4095  600899.8 0.3750594  646.0
## 43 46333 2066.512    2026 4095      4095  600579.2 0.3750138  646.0
```

Faça um gráfico que descreva a variação da mediana em função do tempo

```r
# Seleção do sinal de interesse (músculo anterior)
sinal <- dados$MuscAnterior

# Definindo o tamanho da janela (em número de amostras)
janela_size <- 500

# Gerando os índices iniciais de cada janela
inicio_janela <- seq(from = 1, to = length(sinal), by = janela_size)
num_janelas <- length(inicio_janela)  # Número de janelas

# Criando um data frame vazio para armazenar os resultados
mediana_resultados <- data.frame(
  time = rep(NA, num_janelas - 1),
  mediana = NA
)

# Loop para calcular a mediana em cada janela
for (i in 1:(num_janelas - 1)) {
  # Calcula o tempo centralizado para cada janela
  mediana_resultados$time[i] <- (inicio_janela[i] + (inicio_janela[i + 1] - inicio_janela[i]) / 2) * dt
  # Calcula a mediana do sinal na janela atual
  mediana_resultados$mediana[i] <- median(sinal[inicio_janela[i]:inicio_janela[i + 1]])
}

# Plotando os resultados usando dygraphs
ggplot(mediana_resultados, aes(x = time, y = mediana)) +
  geom_line(color = "blue") +
  labs(title = "Variação da Mediana do Sinal Anterior ao Longo do Tempo",
       x = "Tempo (s)",
       y = "Mediana do Sinal") +
  theme_minimal()
```


Variação da Mediana do Sinal Anterior ao Longo do Tempo
Mediana do Sinal
2100
2080
2060
2040
2020
0
25
50
75
100
Tempo (s)